BEA WebLogic Server™

WebLogic Tuxedo Connector Administration Guide

Release 7.0
Date: April 8, 2004

Copyright

Restricted Rights Legend

Trademarks or Service Marks

WebLogic Tuxedo Connector Administration Guide

Contents

About This Document

1. Introduction to WebLogic Tuxedo Connector

2. Configuring WebLogic Tuxedo Connector

3. WebLogic Tuxedo Connector Administration

7. Using FML with WebLogic Tuxedo Connector

8. Connecting WebLogic Process Integrator and Tuxedo Applications

9. Troubleshooting The WebLogic Tuxedo Connector

10. WebLogic Tuxedo Connector MBean Attributes

About This Document

This document introduces the BEA WebLogic Tuxedo Connector™ application development environment. This document provides information on how to configure and administer the WebLogic Tuxedo Connector to interoperate between WebLogic Server and Tuxedo.

The document is organized as follows:

- Chapter 1, "Introduction to WebLogic Tuxedo Connector," is an overview of the WebLogic Tuxedo Connector.
- Chapter 2, "Configuring WebLogic Tuxedo Connector," describes how to configure the WebLogic Tuxedo Connector.
- Chapter 3, "WebLogic Tuxedo Connector Administration," provides configuration information about the WebLogic Tuxedo Connector.
- Chapter 4, "How to Manage WebLogic Tuxedo Connector in a Clustered Environment," provides information on how to use WebLogic Tuxedo Connector in a clustered environment.
- Chapter 5, "Administration of CORBA Applications," provides information on how to administer CORBA applications.
- Chapter 6, "How to Configure the tBridge Message Interface," provides information on tBridge functionality and configuration.
- Chapter 7, "Using FML with WebLogic Tuxedo Connector," discusses the Field Manipulation Language (FML) and describes how the WebLogic Tuxedo Connector uses FML.
- Chapter 8, "Connecting WebLogic Process Integrator and Tuxedo Applications," discusses WebLogic Process Integrator - Tuxedo integration using the WebLogic Tuxedo Connector.

- Chapter 9, "Troubleshooting The WebLogic Tuxedo Connector," provides WebLogic Tuxedo Connector troubleshooting information.
- Chapter 10, "WebLogic Tuxedo Connector MBean Attributes," provides reference information on WebLogic Tuxedo Connector MBeans and attributes.

Audience

This document is intended for system administrators and application developers who are interested in building distributed Java applications that interoperate between WebLogic Server and Tuxedo environments. It assumes a familiarity with the WebLogic Server, Tuxedo, and Java programming.

e-docs Web Site

BEA product documentation is available on the BEA corporate Web site. From the BEA Home page, click on Product Documentation or go directly to the WebLogic Server 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 main topic at a time, by using the File—Print option on your Web browser.

A PDF version of this document is available on the WebLogic Server 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 WebLogic Server documentation Home page, click Download Documentation, and select the document you want to print.

Adobe Acrobat Reader is available at no charge from the Adobe Web site at http://www.adobe.com.

Related Information

The BEA corporate Web site provides all documentation for WebLogic Server and Tuxedo.

For more information about Java and Java CORBA applications, refer to the following sources:

- The OMG Web Site at http://www.omg.org/
- The Sun Microsystems, Inc. Java site at http://java.sun.com/

Contact Us!

Your feedback on BEA 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 documentation.

In your e-mail message, please indicate the software name and version you are using, as well as the title and document date of your documentation. If you have any questions about this version of BEA WebLogic Server, or if you have problems installing and running BEA WebLogic Server, contact BEA Customer Support through BEA WebSupport at http://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	Usage
Ctrl+Tab	Keys you press simultaneously.
italics	Emphasis and book titles.

`monospace text`	Code samples, commands and their options, Java classes, data types, directories, and file names and their extensions. Monospace text also indicates text that you enter from the keyboard.
	Examples:
	`import java.util.Enumeration;`
	`chmod u+w *`
	`config/examples/applications`
	`.java`
	`config.xml`
	`float`

`monospace italic text`	Variables in code. *Example*: `String CustomerName;`
UPPERCASE TEXT	Device names, environment variables, and logical operators. *Example*s: LPT1 BEA_HOME OR
`{ }`	A set of choices in a syntax line.

Convention	Usage
`[ ]`	Optional items in a syntax line. *Example*: `java utils.MulticastTest -n name -a address [-p portnumber] [-t timeout] [-s send]`
`|`	Separates mutually exclusive choices in a syntax line. *Example*: `java weblogic.deploy [list|deploy|undeploy|update] password {application} {source}`
`...`	Indicates one of the following in a command line: ■ An argument can be repeated several times in the command line. ■ The statement omits additional optional arguments. ■ You can enter additional parameters, values, or other information
. . .	Indicates the omission of items from a code example or from a syntax line.

CHAPTER

1 Introduction to WebLogic Tuxedo Connector

The following sections summarize the concepts and functionality of WebLogic Tuxedo Connector for WebLogic Server Release 7.0:

WebLogic Tuxedo Connector Overview

The WebLogic Tuxedo Connector provides interoperability between WebLogic Server applications and Tuxedo services. The connector allows WebLogic Server clients to invoke Tuxedo services and Tuxedo clients to invoke WebLogic Server Enterprise Java Beans (EJBs) in response to a service request.

Key Functionality and Administrative Features

The WebLogic Tuxedo Connector enables you to develop and support applications interoperating WebLogic Server and Tuxedo by using a Java Application-to-Transaction Monitor Interface (JATMI) similar to the Tuxedo ATMI. The WebLogic Tuxedo Connector tBridge functionality provides Tuxedo /Q and JMS advanced messaging services.

The WebLogic Tuxedo Connector provides the following bi-directional interoperability:

- Ability to call WebLogic Server applications from Tuxedo applications and vice versa or to call EJBs.
- Ability to integrate WebLogic Server applications into existing Tuxedo environments.
- Transaction support.
- Ability to provide interoperability between CORBA Java and CORBA C++ server applications.
- Ability to use WebLogic Process Integrator to manage workflow across Tuxedo ATMI services, including eLink 1.2 adapters.
- Ability to define multiple connections between WebLogic Server and Tuxedo.

The WebLogic Tuxedo Connector includes the following key administration features:

- Simple implementation. The WebLogic Tuxedo Connector does not require modification of existing Tuxedo application code.
  - Existing Tuxedo clients call WebLogic Server EJBs through the WebLogic Tuxedo Connector.
  - New or modified WebLogic Server clients call Tuxedo services through WebLogic Tuxedo Connector.
- Bi-directional security propagation, including domain and ACL security.
- Domain-level failover and fallback.
- Advanced messaging services provided by Tuxedo /Q and JMS.
- Interoperability with mainframes and other legacy applications using eLink.

Known Limitations

WebLogic Tuxedo Connector has the following limitations:

- Does not support dynamic configuration changes to the WebLogic Tuxedo Connector gateway.
- Does not support inbound RMI/IIOP or outbound CORBA in clustered environments.
- Does not support Tuxedo 6.5 running on VMS and OS/390 platforms.
- Does not support outbound CORBA applications from WebLogic Server running on IBM AIX platforms. WebLogic Server uses a Sun specific extension for creating sockets that is not compatible with IBM JDK.

How WebLogic Tuxedo Connector Differs from Jolt

The WebLogic Tuxedo Connector is not a replacement for Jolt. WebLogic Tuxedo Connector differs from Jolt in the following ways:

- WebLogic Tuxedo Connector offers a similar but different API than Jolt.
- Jolt enables the development of generic Java clients and other Web server applications that the WebLogic Tuxedo Connector does not.
- Jolt does not provide a mechanism for an integrated WebLogic Server-Tuxedo transaction.

Users should use Jolt as a solution instead of the WebLogic Tuxedo Connector when a generic Java client or other Web server application is required and WebLogic Server is not part of the solution.

Documentation

You can download the WebLogic Tuxedo Connector documentation from the following location:

- On the BEA corporate Web Site. From the BEA Home page at http://www.bea.com, click on Product Documentation.
- Go directly to the WebLogic Server "e-docs" product documentation page at http://e-docs.bea.com/index.html. Follow the links from the WebLogic Server 7.0 Documentation Center.

Platform Support

See our Platforms Support page at http://www.weblogic.com/platforms/index.html for the most accurate and current information regarding platform support.

Licensing

Note: For more information on WebLogic Server licensing information, see Installing a WebLogic Server License at http://e-docs.bea.com/wls/docs70/install/instlic.html.

This section provides licensing information for the WebLogic Tuxedo Connector:

- There is no license requirement for using the connector without encryption.
- An appropriate Tuxedo LLE license and an appropriate *WebLogic Server* SSL license is required to use encryption.

Updating WebLogic Tuxedo Connector 6.x Applications

You must make some changes in your WebLogic Tuxedo Connector 6.x applications (including WebLogic Tuxedo Connector 1.0) to use them with WebLogic Server 7.0. For detailed information on the administration and programming changes required to upgrade to WebLogic Tuxedo Connector in WebLogic Server 7.0, see Upgrading WebLogic Server 6.x to Version 7.0 at http://e-docs.bea.com/wls/docs70/upgrade/upgrade6xto70.html.

2 Configuring WebLogic Tuxedo Connector

The following sections describe how to configure the WebLogic Tuxedo Connector.

Summary of Environment Changes and Considerations

This section provides an overview of the changes you must make to the Tuxedo and WebLogic Server environments before you can start using the WebLogic Tuxedo Connector.

Tuxedo Changes

Note: For more information on Tuxedo domains, see the *BEA TUXEDO Domains Guide*.

Tuxedo users need to make the following environment changes:

- If an existing Tuxedo application is already using Tuxedo /T DOMAINS, then a new domain must be added to the domains configuration file for each connection to a WebLogic Tuxedo Connector instantiation.
- If the existing Tuxedo application does not use domains, then the domain servers must be added to the *TUXCONFIG* of the application. A new *DMCONFIG* must be created with a Tuxedo /T Domain entry corresponding to the WebLogic Tuxedo Connector instantiation.
- WebLogic Tuxedo Connector requires that the Tuxedo domain always have encoding turned on. `MTYPE` should always be unset or set to NULL in the *DMCONFIG* file.

WebLogic Server Changes

The following sections describe WebLogic Server changes required to use the WebLogic Tuxedo Connector:

Administration and Programming

WebLogic Server users need to make the following environment changes:

- Create Java clients or servers. For more information on creating WebLogic Tuxedo Connector clients or servers, see the *WebLogic Tuxedo Connector Programmer's Guide*.
- Configure the WebLogic Tuxedo Connector using the WebLogic Server console or command-line interface. For more information on how to configure the WebLogic Tuxedo Connector, see "Configuring WebLogic Tuxedo Connector for Your Applications" on page 2-3.
- If the Local WLS Domain ACL Policy is set to `Local`, the Tuxedo remote domain `DOMAINID` must be authenticated as a local user. For more information, see "User Authentication" on page 3-11.

WebLogic Server Threads

Note: For more information on WebLogic Server performance and tuning, see BEA WebLogic Server Performance and Tuning at http://e-docs.bea.com/wls/docs70/perform/index.html.

The number of client threads available when dispatching services from the gateway may limit the number of concurrent services running. For this release of WebLogic Tuxedo Connector, there is no WebLogic Tuxedo Connector attribute to increase the number of available threads. Use a reasonable thread model when invoking service EJBs.You may need to increase the number of WebLogic Server threads available to a larger value.

Configuring WebLogic Tuxedo Connector for Your Applications

Note: This release of the WebLogic Tuxedo Connector provides only static configuration. If you need to change any parameters used to configure the WebLogic Tuxedo Connector, the WebLogic Server must be restarted for the changes to take effect. For example, you can not add or remove domain network links, change network addresses, or import or export new services.

This section provides information on how to configure the WebLogic Tuxedo Connector to allow WebLogic Server applications and Tuxedo applications to interoperate.

- WebLogic Tuxedo Connector MBean Classes
- Configuring WebLogic Tuxedo Connector Using the Administration Console
- Configuring WebLogic Tuxedo Connector Using the Command-Line Interface
- Set the WebLogic Server Environment
- How to Set WebLogic Tuxedo Connector Properties
- WebLogic Tuxedo Connector Configuration Guidelines

WebLogic Tuxedo Connector MBean Classes

Note: For more information on MBean parameters, see "WebLogic Tuxedo Connector MBean Attributes" on page 10-1.

The WebLogic Tuxedo Connector uses MBeans to describe connectivity information and security protocols to process service requests between WebLogic Server and Tuxedo. These configuration parameters are analogous to the interoperability attributes required for communication between Tuxedo domains. The configuration parameters are stored in the WebLogic Server `config.xml file`. The following table lists the MBean types used to configure WebLogic Tuxedo Connector:

MBean Type	Description
WTCServerMBean	Parent MBean containing the interoperability attributes required for a connection between WebLogic Server and Tuxedo.
WTCLocalTuxDomMBean	Provides a view of local domains as they appears to other domains. You must configure at least one local domain.
WTCRemoteTuxDomMBean	Provides a view of remote domains as they appears to a local domain. You may configure multiple remote domains.
WTCExportMBean	Provides information on services exported by a local domain.
WTCImportMbean	Provides information on services imported and available on remote domains.
WTCResourcesMBean	Specifies global field table classes, view table classes, and application passwords for domains
WTCPasswordMbean	Specifies the configuration information for inter-domain authentication.
WTCtBridgeGlobalMBean	Specifies global configuration information for the transfer of messages between WebLogic Server and Tuxedo.
WTCtBridgeRedirectMBean	Specifies the source, target, direction, and transport of messages between WebLogic Server and Tuxedo.

Configuring WebLogic Tuxedo Connector Using the Administration Console

The Administration Console allows you to configure, manage, and monitor WebLogic Tuxedo Connector connectivity. To display the tabs that you use to perform these tasks, complete the following procedure:

1. Start the Administration Console.
2. Locate the Services node in the left pane, then expand the WebLogic Tuxedo Connector node.
3. Create or modify the node in the tree specific to the component you want to configure.
4. Follow the instructions in the Online Help. For links to the Online Help, see Table 2-1.

The following table shows the connectivity tasks, listed in typical order in which you perform them. You may change the order; just remember you must configure an object before associating or assigning it.

Table 2-1 WebLogic Tuxedo Connector Configuration Tasks

Task #	Task	Description
1	Creating a WTCServer	On the General tab in the right pane, you set the attributes for Name and Deployment Order.
2	Creating a Local WLS Domain	Set the attributes that describe your local domain in the General, Connections, and Security tabs. You must configure at least one local domain.
3	Creating a Remote Tuxedo Domain	Set the attributes that describe your remote Tuxedo domains in the General, Connections, and Security tabs.
4	Creating an Exported Service	Set the attributes that describe your exported WebLogic Server services in the General tab.
5	Creating Imported Services	Set the attributes that describe your imported Tuxedo services in the General tab.

Table 2-1 WebLogic Tuxedo Connector Configuration Tasks

Task #	Task	Description
6	Creating a Password Configuration	Set the attributes that describe your passwords in the Configuration tab.
7	Creating a Resource	Set the attributes that describe your WebLogic Tuxedo Connector resources in the Configuration tab.
8	Creating a tBridge Connection	Set the global configuration information for the transfer of messages between WebLogic Server and Tuxedo.
9	Creating a tBridge Redirection	Sets the attributes used to specify the source, target, direction, and transport of a message between WebLogic Server and Tuxedo
10	Assign a WTCServer to a Server	Select a target server for your WTCServer MBean.

Configuring WebLogic Tuxedo Connector Using the Command-Line Interface

The command-line interface provides a way to create and manage WebLogic Tuxedo Connector connections. For information on how to use the command-line interface, see the Mbean Management Command Reference at http://e-docs.bea.com/wls/docs70/adminguide/cli.html#1155716.

Set the WebLogic Server Environment

You need to set the environment of your WebLogic Server application by running the `setEnv` script.

- NT/2000 users: run `setEnv.cmd`
- UNIX users: run `setEnv.sh`

If you are setting the environment for the first time, you will need to review the settings in the script. If necessary, use the following steps to modify the settings for your application environment:

1. From the command line, change directories to the location of the WebLogic Server application.
2. Edit the `setEnv` script with a text editor, such as `vi`.
   - NT/2000 users: edit `setEnv.cmd`
   - UNIX users: edit `setEnv.sh`
3. Save the file.

Note: The `setExamplesEnv` file is used to set the environment for the WebLogic Server examples provided with your distribution.

How to Set WebLogic Tuxedo Connector Properties

Note: For more information about setting WebLogic Server properties, see Starting and Stopping WebLogic Servers at http://e-docs.bea.com/wls/docs70/adminguide/startstop.html.

`TraceLevel`, `PasswordKey` and `encoding` are WebLogic Server Properties. If you need to set these properties, update the `JAVA_OPTIONS` variable in your server start script. Example:

```
JAVA_OPTIONS=-Dweblogic.wtc.TraceLevel=100000
```

Set TraceLevel

Note: For more information on `TraceLevel`, see "Monitoring the WebLogic Tuxedo Connector" on page 9-1.

Use `TraceLevel` to specify the level of message tracing WebLogic Tuxedo Connector will send to your log files:

```
JAVA_OPTIONS= -Dweblogic.wtc.TraceLevel=tracelevel
```

where *tracelevel* is a number between 10,000 and 100,000 that specifies the level of WebLogic Tuxedo Connector tracing.

Set PasswordKey

Note: For more information on `PasswordKey`, see "Configuring a WTCPassword MBean" on page 3-8.

Use `PasswordKey` to specify the key used by the `weblogic.wtc.gwt.genpassword` utility to encrypt passwords:

`JAVA_OPTIONS=-Dweblogic.wtc.PasswordKey=`*mykey*

where *mykey* is the key value.

Set encoding

To transfer non-ascii (multibyte) strings between WebLogic Server and Tuxedo applications, you must configure WebLogic Tuxedo Connector to provide characer set translation. WebLogic Tuxedo Connector uses a WebLogic Server property to match the encoding used by all the Tuxedo remote domains specified in a WebLogic Tuxedo Connector service. If you require more than one coding set running simultaneously, you will require WebLogic Tuxedo Connector services running in separate WebLogic Server instances.

To enable character set translation, update the JAVA_OPTIONS variable in your server start script. Example:

`JAVA_OPTIONS=-Dweblogic.wtc.encoding=`*codesetname*

where *codesetname* is the name of a supported codeset used by a remote Tuxedo domain. See Supported Encodings at http://java.sun.com/j2se/1.3/docs/guide/intl/encoding.doc.html for list of supported supported base and extended coding sets.

You may not be able to select the exact encoding name to match the encoding used by the remote domain. In this situation, you should select an encoding name that is equivalent to the remote domain.

Example:

- The Supported Encoding list includes `EUC_JP`
- The remote domain is supported by a Solaris operating system using `eucJP`

Although the names don't match exactly, `EUC_JP` and `eucJP` are equivalent encoding sets and provide the correct string translation between WebLogic Server and your remote domain. You should set the encoding property to `EUC_JP`:

```
JAVA_OPTIONS=-Dweblogic.wtc.encoding=EUC_JP
```

WebLogic Tuxedo Connector Configuration Guidelines

Use the following guidelines when configuring WebLogic Tuxedo Connector:

- You must have at least one local TDOM in your configuration.
- You may create 1 or more WTCServer MBeans in your `config.xml` file.
- You can not target 2 or more WTCServer MBeans to the same server. A server can only be a target for 1 WTCServer MBean.
- Any configuration changes implemented in a WTCServer MBean after a target server is selected will not be updated in the target server instance. You must remove the WTCServer MBean from the server and then add the updated WTCServer Mbean add to the target server. For more information on selecting a target server, see Assign a WTCServer to a Server.

3 WebLogic Tuxedo Connector Administration

Note: For more detailed reference information on the WebLogic Tuxedo Connector MBean attributes, see Chapter 10, "WebLogic Tuxedo Connector MBean Attributes."

The following sections describe how to establish connectivity and provide security between domains in the WebLogic Server and Tuxedo environments. The WebLogic Tuxedo Connector MBeans attributes are analogous to the interoperability attributes required for the communication between Tuxedo domains.

The following sections provide WebLogic Tuxedo Connector configuration information:

- Configuring the Connections Between Domains
- How ConnectionPolicy Affects Dynamic Status
- Configuring Domains-level Failover and Failback
- Authentication of Remote Domains
- User Authentication
- How to Configure WebLogic Tuxedo Connector to Provide Security between Tuxedo and WebLogic Server
- Example ACL Policy for Simpapp and Simpserv Examples

- Link-Level Encryption

Configuring the Connections Between Domains

Note: For more information on Dynamic Status, see "How ConnectionPolicy Affects Dynamic Status" on page 3-5.

Several options can specify the conditions under which a local domain gateway tries to establish a connection with a remote domain. Specify these conditions using the `ConnectionPolicy` attribute in the `WTCLocalTuxDom` MBean and `WTCRemoteTuxDom` MBean. You can select any of the following connection policies:

- How to Request a Connection at Boot Time (ON_STARTUP)
- How to Request Connections for Client Demands (ON_DEMAND)
- Accepting Incoming Connections (INCOMING_ONLY)
- How to use LOCAL Connection Policy

For connection policies of `ON_STARTUP` and `INCOMING_ONLY`, Dynamic Status is invoked. Dynamic Status checks and reports on the status of imported services associated with each remote domain.

How to Request a Connection at Boot Time (ON_STARTUP)

A policy of `ON_STARTUP` means that a domain gateway attempts to establish a connection with its remote domain access points at gateway server initialization time. The connection policy retries failed connections at regular intervals determined by the `RetryInterval` parameter and the `MaxRetries` parameter. To request a connection at boot time, set the `WTCLocalTuxDom` MBean connection policy to `ON_STARTUP`.

How to Configure RetryInterval

You can control the frequency of automatic connection attempts by specifying the interval (in seconds) during which the gateway should wait before trying to establish a connection again. The minimum value is 0; the default value is 60, and maximum value is 2147483647.

How to Configure MaxRetries

Note: Use only when `ConnectionPolicy` is set to `ON_STARTUP`. For other connection policies, retry processing is disabled.

You indicate the number of times a domain gateway tries to establish connections to remote domain access points before quitting by assigning a value to the `MaxRetries` parameter: the minimum value is 0; the default and maximum value is 2147483647.

- If you set `MaxRetries` to 0, automatic connection retry processing is turned off. The server does not attempt to connect to the remote gateway automatically.
- If you set `MaxRetries` to a number, the gateway tries to establish a connection the specified number of times before quitting.
- If you set `MaxRetries` to 2147483647, retry processing is repeated indefinitely or until a connection is established.

Table 3-1 Example Settings of MaxRetries and RetryInterval Parameters

If you set ...	Then ...

`ConnectionPolicy: ON_STARTUP`
`RetryInterval: 30`
`MaxRetries: 3`
	The gateway makes 3 attempts to establish a connection, at 30 seconds intervals, before quitting.

`ConnectionPolicy: ON_STARTUP` `MaxRetries: 0`	The gateway attempts to establish a connection at initialization time but does not retry if the first attempt fails.
`ConnectionPolicy: ON_STARTUP` `RetryInterval: 30`	The gateway attempts to establish a connection every 30 seconds until a connection is established.

How to Request Connections for Client Demands (ON_DEMAND)

Note: If the `ConnectionPolicy` is not specified, the WebLogic Tuxedo Connector uses a `ConnectionPolicy` of `ON_DEMAND`.

A connection policy of `ON_DEMAND` means that a connection is attempted only when requested by either a client request to a remote service or an administrative connect command.

Accepting Incoming Connections (INCOMING_ONLY)

A connection policy of `INCOMING_ONLY` means that a domain gateway does not establish a connection to remote domains upon starting. The domain gateway is available for incoming connections from remote domain access points and remote services are advertised when the domain gateway receives an incoming connection.

How to use LOCAL Connection Policy

Note: A `ConnectionPolicy` of `LOCAL` is not valid for local domains.

A connection policy of `LOCAL` indicates that a remote domain connection policy is explicitly defaulted to the local domain `ConnectionPolicy` attribute value. If the remote domain `ConnectionPolicy` is not defined, the system uses the setting specified by the associated local domain using the `LocalAccessPoint`.

How ConnectionPolicy Affects Dynamic Status

Dynamic Status determines the availability of remote services. The connection policy used in the WebLogic Tuxedo Connector configuration file determines whether the Dynamic Status feature is available for a service. The following table describes how `ConnectionPolicy` affects Dynamic Status capability.

ON_STARTUP	Dynamic Status is on. Services imported from a remote domain are advertised while a connection to that remote domain exists.
ON_DEMAND	Dynamic Status is off. Services imported from remote domains are always advertised.
INCOMING_ONLY	Dynamic Status is on. Remote services are initially suspended. The domain gateway is available for incoming connections from remote domains. Remote services are advertised when the local domain gateway receives an incoming connection.

Configuring Domains-level Failover and Failback

Note: In the Tuxedo T/ Domain, there is a limit of 2 backup remote domains. The WebLogic Tuxedo Connector has no limit to the number of backup domains allowed to be configured for a service.

WebLogic Tuxedo Connector provides a domains-level failover mechanism that transfers requests to alternate remote domains when a failure is detected with a primary remote domain. It also provides failback to the primary remote domain when that domain is restored.

This level of failover/failback depends on Dynamic Status. The domain must be configured with a CONNECTION_POLICY of ON_STARTUP or INCOMING_ONLY to enable Domains-level failover/failback.

Domains-level failover/failback defines a remote domain as available when a network connection to the remote domain exists, and unavailable when a network connection to the remote domain does not exist.

Prerequisite to Using Domains-level Failover and Failback

To use Domains-level failback, you must specify ON_STARTUP or INCOMING_ONLY as the value of the CONNECTION_POLICY parameter.

A connection policy of ON_DEMAND is unsuitable for Domains-level failback as it operates on the assumption that the remote domain is always available. If you do not specify ON_STARTUP or INCOMING_ONLY as your connection policy, your servers cannot fail over to the alternate remote domains that you have specified with the Tuxedo RDOM parameter.

Note: A remote domain is *available* if a network connection to it exists; a remote domain is *unavailable* if a network connection to it does not exist.

How to Configure Domains to Support Failover

To support failover, you must specify the remote domains responsible for executing a particular service. You must specify the following in your WTCServer MBean:

- Create a WTCRemoteTuxDom MBean for each remote domain.
- Create a WTCImport MBean that specifies the service provided by each remote domain.

Suppose a service, TOUPPER, is available from two remote domains: TDOM1 and TDOM3. Your WTCServer Mbean would include two WTCRemoteTuxDom MBeans and two WTCImport MBeans. The WTCServer Mbean in your config.xml file would contain the following:

```
<WTCServer Name="WTCsimpapp"
     <WTCExport EJBName="tuxedo.services.TOLOWERHome"
           LocalAccessPoint="TDOM2" Name="myExportedResources"
          ResourceName="TOLOWER"/>
     <WTCImport LocalAccessPoint="TDOM2" Name="myImportedResources"
          RemoteAccessPointList="TDOM1" ResourceName="TOUPPER"/>
     <WTCImport LocalAccessPoint="TDOM2" Name="2ndImportedResources"
          RemoteAccessPointList="TDOM3" ResourceName="TOUPPER"/>
     <WTCLocalTuxDom AccessPoint="TDOM2" AccessPointId="TDOM2"
          ConnectionPolicy="ON_DEMAND" Interoperate="no"
          NWAddr="//123.123.123.123:5678" Name="myLoclTuxDom" Security="NONE"/>
     <WTCRemoteTuxDom AccessPoint="TDOM1" AccessPointId="TDOM1"
           LocalAccessPoint="TDOM2" NWAddr="//123.123.123.123:1234"
          Name="myRTuxDom"/>
     <WTCRemoteTuxDom AccessPoint="TDOM3" AccessPointId="TDOM3"
           LocalAccessPoint="TDOM2" NWAddr="//234.234.234.234:5555"
          Name="2ndRemoteTuxDom"/>
</WTCServer>
```

How to Configure Domains to Support Failback

Failback occurs when a network connection to the primary remote domain is reestablished for any of the following reasons:

- Automatic retries (`ON_STARTUP` only)
- Incoming connections

Authentication of Remote Domains

Note: Tuxedo 6.5 users should set the `Interoperate` parameter to `Yes`.

Domain gateways can be made to authenticate incoming connections requested by remote domains and outgoing connections requested by local domains. Application administrators can define when security should be enforced for incoming connections from remote domains. You can specify the level of security used by a particular local domain by setting the `SECURITY` parameter in the `WTCLocalTuxDom` MBean of your `WTCServer` Mbean. There are three levels of password security:

- *No Security* (using the `NONE` option)—incoming connections from remote domains are not authenticated.

- *Application Password* (using the `APP_PW` option)— incoming connections from remote domains are authenticated using the application password defined in the `WTCPassword` MBean of your `WTCServer` Mbean. You use the `weblogic.wtc.gwt.genpasswd` utility to create encrypted application passwords.
- *Remote Domains Password* (using the `DM_PW` option*)*—this feature enforces security between two or more domains. Connections between the local and remote domains are authenticated using password pairs defined in the `WTCPassword` MBean of your `WTCServer` Mbean. You use the `weblogic.wtc.gwt.genpasswd` utility to create encrypted local and remote passwords.

The `SECURITY` parameter in the `WTCLocalTuxDom` MBean of your `WTCServer` Mbean must match the SECURITY parameter of the *DM_LOCAL_DOMAINS section of the Tuxedo domain configuration file.

- If authentication is required, it is done every time a connection is established between the local domain and a remote domain.
- If the security type of the `WTCLocalTuxDom` MBean does not match the security type of the *DM_LOCAL_DOMAINS or if the passwords do not match, the connection fails.

Configuring a WTCPassword MBean

Note: For more information on how to assign a `PasswordKey`, see "How to Set WebLogic Tuxedo Connector Properties" on page 2-7.

Use `weblogic.wtc.gwt.genpasswd` to generate encrypted passwords for LocalPassword, RemotePassword, and AppPassword elements. The utility uses a key to encrypt a password that is copied into the `WTCPassword` MBean of your `WTCServer` Mbean.

- The `WTCPassword` MBean does not store clear text passwords.
- The key value is a WebLogic Server property.
  - `PasswordKey` is the attribute used to assign the key.

    `-Dweblogic.wtc.PasswordKey=`*`mykey`*

where: *mykey* is the key value

- The `PasswordKey` attribute can only be assigned one key value. This key value is used for all Weblogic Tuxedo Connector passwords generated (local, remote, and application) for use with a specific WebLogic Server.

Usage

Call the utility without any arguments to display the command line options.

Example:

```
$ java weblogic.wtc.gwt.genpasswd

Usage: genpasswd Key <LocalPassword|RemotePassword|AppPassword>
<local|remote|application>
```

The utility will respond with an XML element containing the encoded password and password IV. Cut and paste the results into the appropriate fields in your WTCServer MBean.

```
<LocalPassword IV="my_passwordIV">my_password</LocalPassword>
```

where

- Cut and paste the string of characters represented by *my_passwordIV* into the PasswordIV field. Do not include the quotation marks.
- Cut and paste the string of characters represented by *my_password* into the Password field. Do not include > or <.

Examples

This section provides examples of each of the password element types.

LocalPasswords

The following example uses *key1* to encrypt "LocalPassword1" as the password of the local domain.

```
$ java weblogic.wtc.gwt.genpasswd Key1 LocalPassword1 local
<LocalPassword IV=”I#^Da0efo1”>!djK*87$klbJJ</LocalPassword>
```

Your Password MBean attributes are:

Local Password IV: `I#^Da0efo1`

Local Password: `!djK*87$klbJJ`

RemotePasswords

The following example uses *mykey* to encrypt “RemotePassword1” as the password for the remote domain.

```
$ java weblogic.wtc.gwt.genpasswd mykey RemotePassword1 remote
<RemotePassword IV=”Rq$45%%kK”>McFrd3#f41Kl</RemotePassword>
```

Your Password MBean attributes are:

Remote Password IV: `Rq$45%%kK`

Remote Password: `McFrd3#f41Kl`

AppPasswords

The following example uses *key1* to encrypt “test123” as the application password.

```
$ weblogic.wtc.gwt.genpasswd mykey test123 application
<AppPassword IV="gx8aSkAgLFg=">c98Y/P94HY3rCAVmkF=</AppPassword>
```

Your Resources MBean attributes are:

Application Password IV: `gx8aSkAgLFg=`

Application Password: `c98Y/P94HY3rCAVmkF=`

User Authentication

Access Control Lists (ACLs) limit the access to local services within a Local WLS Domain by restricting the remote Tuxedo domains that can execute these services. Inbound policy from a remote Tuxedo domain is specified using the `AclPolicy` element. Outbound policy towards a remote Tuxedo domain is specified using the `CredentialPolicy` element. This allows WebLogic Server and Tuxedo applications to share the same set of users and the users are able to propagate their credentials from one system to the other.

The valid values for `AclPolicy` and `CredentialPolicy` are:

- `LOCAL`
- `GLOBAL`

ACL Policy is LOCAL

If the WebLogic Tuxedo Connector ACL Policy is set to `Local`, access to local services does not depend on the `CredentialPolicy`. The Tuxedo remote domain `DOMAINID` is authenticated as a local WebLogic Server user. To allow WebLogic Tuxedo Connector to authenticate a `DOMAINID` as a local user, use the WebLogic Server Console to complete the following steps:

1. Click on the Security node.
2. Click on Realms.
3. Select your default security Realm.
4. Click on Users.
5. Click the Configure a new User text link.
6. Click DefaultAuthenticator
7. In the General tab, do the following:
   a. Add the Tuxedo `DOMAINID` in the Name field.
   b. Enter and validate a password.

c. Click apply.

ACL Policy is GLobal

If the WebLogic Tuxedo Connector ACL Policy is `GLOBAL`, access to local services depends on the `CredentialPolicy`.

Credential Policy is Global

If a remote domain is running with the `CredentialPolicy` set to `GLOBAL`, the request has the credentials of the caller.

CredentialPolicy is Local

If a remote domain is running with the `CredentialPolicy` set to `LOCAL`, the result depends on the user configuration that initiated the call.

User Authenticaion for Tuxedo 6.5

Tuxedo 6.5 users should set the `Interoperate` parameter to `Yes`. The `AclPolicy` and `CredentialPolicy` elements are ignored and the Tuxedo remote domain `DOMAINID` is authenticated as a local WebLogic Server user. If you require User Security features and use the WebLogic Tuxedo Connector, you will need to upgrade to Tuxedo 7.1 or higher.

How to Configure WebLogic Tuxedo Connector to Provide Security between Tuxedo and WebLogic Server

Use the following steps to configure WebLogic Tuxedo Connector to provide security between Tuxedo and WebLogic Server applications:

- Configure a WTCLocalTuxDom MBean
- Configure a WTCRemoteTuxDom MBean

Configure a WTCLocalTuxDom MBean

Set the `SECURITY` parameter in the `WTCLocalTuxDom` MBean of your `WTCServer` MBean to match the SECURITY parameter of the *DM_LOCAL_DOMAINS section of the Tuxedo domain configuration file.

Example WTCLocalTuxDom MBean Configuration

```
<WTCLocalTuxDom AccessPoint="TDOM2" AccessPointId="TDOM2"
    ConnectionPolicy="ON_DEMAND" Interoperate="no"
     NWAddr="//123.123.123.123:5678" Name="myLoclTuxDom"
     Security="DM_PW"/>
```

Example Tuxedo *DM_LOCAL_DOMAINS Configuration

```
.
.
.
*DM_LOCAL_DOMAINS
     domain1      DOMAINID = "domain1"
     GWGRP = "GWGRP"
     CONNECTION_POLICY=ON_DEMAND
     DMTLOGDEV = "/nfs/home1/dkumar/lcsol5/tmp.100/DMTLOG"
     DMTLOGNAME = "DMTLG1"
     SECURITY=DM_PW
```

```
        BLOCKTIME=10
.
.
.
```

Configure a WTCRemoteTuxDom MBean

Configure the `WTCRemoteTuxDom` MBean to establish an inbound and outbound Access Control List (ACL) policy.

Perform the following steps to prepare the WebLogic Server environment:

1. Set the AclPolicy attribute of the `WTCRemoteTuxDom` MBean of your `WTCServer` MBean to `GLOBAL`.
2. Set the CredentialPolicy attribute of the `WTCRemoteTuxDom` MBean of your `WTCServer` MBean to `GLOBAL`.
3. If the `CredentialPolicy` is set to `GLOBAL`, you must have a copy of the Tuxedo `tpusr` file in your WebLogic Server environment. Use the following steps to add a `TpUserFile` to your environment:
   a. Copy the `tpusr` file from TUXEDO to the WebLogic Server application environment or generate your own `tpusr` file.
   b. Set the `WTCRemoteTuxDom` MBean `TpUserFile` attribute of your `WTCServer` MBean to the fully qualified path and name of the `tpusr` file.

Note: For more information on how to create a Tuxedo tpusr file, see How to Enable User-Level Authentication at http://e-docs.bea.com/tuxedo/tux80/atmi/secadm19.htm.

Example WTCRemoteTuxDom MBean

```
<WTCRemoteTuxDom AccessPoint="TDOM3" AccessPointId="TDOM3"
     AclPolicy="GLOBAL" CredentialPolicy="GLOBAL"
     LocalAccessPoint="TDOM2" NWAddr="//234.234.234.234:5555"
     Name="WTCRemoteTuxDom-1012347174947"
     TpUsrFile="C:\runs\tmp.100\config\wldom1\tpusr"/>
```

Example Tuxedo *DM_LOCAL_DOMAINS Configuration

```
.
.
.
*DM_REMOTE_DOMAINS
     wldom1 DOMAINID = "wldom1"
     ACCESSPOINTID="wldom1"
     ACL_POLICY=”GLOBAL”
     CREDENTIAL_POLICY="GLOBAL"
.
.
.
```

Example ACL Policy for Simpapp and Simpserv Examples

This section provides an example of how to set up ACL control using the `simpapp` and `simpserv` examples.

- Only John and Bob have access to Toupper.
- Only Dan and John to access Tolower.
- Toupper is used for accessing remote the Tuxedo service TOUPPER.
- Tolower provides the actual service for remote Tuxedo user.

Use the following steps to establish ACL control:

1. Add user John, Bob, and Dan to WebLogic Security using the WebLogic Server Console.
2. Modify the `ejb-jar.xml` to add the `security-role` and `method_permission` elements for the Tuxedo TOUPPER service. The bolded statements indicate the changes added to support the security implementation.

Listing 3-1 TOUPPER ejb-jar.xml Security Example Code

```
<?xml version="1.0"?>
<!--
Copyright (c) 2000 BEA Systems, Inc. All rights reserved

THIS IS UNPUBLISHED PROPRIETARY SOURCE CODE OF BEA Systems, Inc. The
copyright notice above does not evidence any actual or intended publication
of such source code.

-->

<!DOCTYPE ejb-jar PUBLIC '-//Sun Microsystems, Inc.//DTD Enterprise JavaBeans
2.0//EN' 'http://java.sun.com/j2ee/dtds/ejb-jar_2_0.dtd'>

<ejb-jar>
     <enterprise-beans>
          <session>
               <ejb-name>Toupper</ejb-name>
     <home>weblogic.wtc.examples.simpapp.ToupperHome</home>
     <remote>weblogic.wtc.examples.simpapp.Toupper</remote>
     <ejb-class>weblogic.wtc.examples.simpapp.ToupperBean</ejb-class>
               <session-type>Stateful</session-type>
               <transaction-type>Container</transaction-type>
          </session>
     </enterprise-beans>
     <assembly-descriptor>
          <security-role>
               <role-name>dom2</role-name>
          </security-role>
          <method-permission>
               <role-name>dom2</role-name>
               <method>
                     <ejb-name>Toupper</ejb-name>
                     <method-name>Toupper</method-name>
               </method>
          </method-permission>
          <container-transaction>
               <method>
                    <ejb-name>Toupper</ejb-name>
                    <method-intf>Remote</method-intf>
                    <method-name>*</method-name>
               </method>
               <trans-attribute>Supports</trans-attribute>
           </container-transaction>
     </assembly-descriptor>
</ejb-jar>
```

3. Modify the `Weblogic-ejb-jar.xml` to add the `security-role-assignment` element for the Tuxedo TOUPPER service. The bolded statements indicate the changes added to support the security implementation.

Listing 3-2 TOUPPER Weblogic-ejb-jar.xml Security Example Code

```
<?xml version="1.0"?>
<!--

Copyright (c) 2000 BEA Systems, Inc. All rights reserved

THIS IS UNPUBLISHED PROPRIETARY SOURCE CODE OF BEA Systems, Inc.
The copyright notice above does not evidence any actual or intended
publication of such source code.

-->

<!DOCTYPE weblogic-ejb-jar PUBLIC '-//BEA Systems, Inc.//DTD WebLogic 6.0.0
EJB//EN' 'http://www.bea.com/servers/wls600/dtd/weblogic-ejb-jar.dtd'>

<weblogic-ejb-jar>
     <weblogic-enterprise-bean>
          <ejb-name>Toupper</ejb-name>
          <stateful-session-descriptor>
               <stateful-session-cache>
          <max-beans-in-cache>100</max-beans-in-cache>
               </stateful-session-cache>
          </stateful-session-descriptor>
          <jndi-name>tuxedo.services.ToupperHome</jndi-name>
     </weblogic-enterprise-bean>
     <security-role-assignment>
          <role-name>dom2</role-name>
          <principal-name>john</principal-name>
          <principal-name>bob</principal-name>
     </security-role-assignment>
</weblogic-ejb-jar>
```

4. Modify the `ejb-jar.xml` to add the `security-role` and `method-permission` elements for the Tolower service. The bolded statements indicate the changes added to support the security implementation.

Listing 3-3 Tolower ejb-jar.xml Security Example Code

```
<?xml version="1.0"?>
<!--

Copyright (c) 2000 BEA Systems, Inc. All rights reserved
THIS IS UNPUBLISHED PROPRIETARY SOURCE CODE OF BEA Systems, Inc.
The copyright notice above does not evidence any actual or intended
publication of such source code.

-->
<!DOCTYPE ejb-jar PUBLIC '-//Sun Microsystems, Inc.//DTD Enterprise JavaBeans
2.0//EN' 'http://java.sun.com/j2ee/dtds/ejb-jar_2_0.dtd'>

<ejb-jar>
     <enterprise-beans>
          <session>
               <ejb-name>Tolower</ejb-name>
            <home>weblogic.wtc.jatmi.TuxedoServiceHome</home>
             <remote>weblogic.wtc.jatmi.TuxedoService</remote>
            <ejb-class>weblogic.wtc.examples.simpserv.TolowerBean</ejb-class>
            <session-type>Stateless</session-type>
            <transaction-type>Container</transaction-type>
          </session>
     </enterprise-beans>

     <assembly-descriptor>
          <security-role>
               <role-name>rdom2</role-name>
          </security-role>
          <method-permission>
               <role-name>rdom2</role-name>
               <method>
                    <ejb-name>Tolower</ejb-name>
                    <method-name>service</method-name>
               </method>
          </method-permission>
          <container-transaction>
               <method>
                    <ejb-name>Tolower</ejb-name>
                    <method-intf>Remote</method-intf>
                    <method-name>*</method-name>
               </method>
               <trans-attribute>Supports</trans-attribute>
          </container-transaction>
     </assembly-descriptor>
</ejb-jar>
```

5. Modify the `Weblogic-ejb-jar.xml` to add the `security-role-assignment` element for the Tolower service.The bolded statements indicate the changes added to support the security implementation.

Listing 3-4 Tolower Weblogic-ejb-jar.xml Security Example Code

```
<?xml version="1.0"?>
<!--

Copyright (c) 2000 BEA Systems, Inc. All rights reserved

THIS IS UNPUBLISHED PROPRIETARY SOURCE CODE OF BEA Systems, Inc.
The copyright notice above does not evidence any actual or intended
publication of such source code.

-->

<!DOCTYPE weblogic-ejb-jar PUBLIC '-//BEA Systems, Inc.//DTD WebLogic 6.0.0
EJB//EN' 'http://www.bea.com/servers/wls600/dtd/weblogic-ejb-jar.dtd'>

<weblogic-ejb-jar>
     <weblogic-enterprise-bean>
          <ejb-name>Tolower</ejb-name>
     <stateless-session-descriptor>
          <pool>
               <max-beans-in-free-pool>100</max-beans-in-free-pool>
          </pool>
     </stateless-session-descriptor>
     <jndi-name>tuxedo.services.TOLOWERHome</jndi-name>
     </weblogic-enterprise-bean>
     <security-role-assignment>
          <role-name>rdom2</role-name>
          <principal-name>john</principal-name>
          <principal-name>dan</principal-name>
     </security-role-assignment>
</weblogic-ejb-jar>
```

6. Perform the following steps to prepare the Tuxedo environment for outbound requests:

- If needed, add the group using `tpgrpadd`.
- If needed, add John, Bob, and Dan as users using `tpusradd`.
- Add TOUPPER service to be protected by TUXEDO ACL using `tpacladd`.
- Set the BDMCONFIG for the remote domain (the WebLogic Server domain) with the ACL_POLICY="GLOBAL".

7. Perform the following steps to prepare the Tuxedo environment for inbound requests:

- If needed, add the group using `tpgrpadd`.
- If needed, add John, Bob, and Dan as users using `tpusradd`.
- Add TOUPPER service to be protected by TUXEDO ACL using `tpacladd`.
- Set the BDMCONFIG for the remote domain (the WebLogic Server domain) with the ACL_POLICY="GLOBAL". If ACL_POLICY="LOCAL", configure the remote DOMAINID or the AccessPointID as a user using `tpusradd`.

8. Perform the following steps to prepare the WebLogic Server environment:

- Copy the `tpusr` file from TUXEDO or generate your own `tpusr` file.
- Set the `WTCRemoteTuxDom` MBean `TpUserFile` attribute of your `WTCServer` MBean to the fully qualified path and name of the `tpusr` file.

9. Set the `WTCRemoteTuxDom` MBean CredentialPolicy attribute of your `WTCServer` MBean to `GLOBAL`.

Link-Level Encryption

You can use encryption across domains to ensure data privacy. In this way, a network-based eavesdropper cannot learn the content of messages or application-generated messages flowing from one domain gateway to another. You configure this security mechanism by setting the `MINENCRYPTBITS` and `MAXENCRYPTBITS` parameters of the WTCLocalTuxDom and the WTCRemoteTuxDom MBeans of your WTCServer MBean.

Note: Encryption requires appropriate licensing. For more information on license requirements, see "Licensing" on page 1-5.

CHAPTER

4 How to Manage WebLogic Tuxedo Connector in a Clustered Environment

Note: For more information on WebLogic Server Clusters, see Using WebLogic Server Clusters at http://e-docs.bea.com/wls/docs70/cluster/index.html.

The following sections provide information on how to administer and configure the WebLogic Tuxedo Connector for use in a clustered environment:

- WebLogic Tuxedo Connector Guidelines for Clustered Environments
- How to Configure OutBound Requests to Tuxedo Domains
- How to Configure Inbound Requests from Tuxedo Domains

WebLogic Tuxedo Connector Guidelines for Clustered Environments

Use the following guidelines when deploying WebLogic Tuxedo Connector in a clustered environment:

- Because the binding is not replicated in other servers in a cluster, all the WebLogic Servers in the cluster must have a configured WebLogic Tuxedo Connector. If one server in the cluster does not have a WebLogic Tuxedo Connector deployed, the EJB or MDB won't be able to find a Tuxedo Connection Factory for that connection.
- You can not shutdown a WebLogic Tuxedo Connector instance in a cluster without shutting down the server.
- All the WebLogic Tuxedo Connector instances in a WebLogic Server cluster must have the same services defined in the `WTCImport` MBean.
- The EJB or MDB that uses the WebLogic Tuxedo Connector must be deployed on all the WebLogic Servers in the cluster.
- The administrator is responsible for the correct configuration of the TUXEDO DMCONFIG to allow proper load balancing and fail over among clustered nodes.
- ATMI connections use the advertised Tuxedo service named to identify services in the `WTCImport` MBean.
- WebLogic Tuxedo Connector does not support inbound RMI/IIOP or outbound Corba in clustered environments.

How to Configure OutBound Requests to Tuxedo Domains

Note: For more information on WebLogic Server Clusters, see Cluster Features and Infrastructure at http://e-docs.bea.com/wls/docs70/cluster/features.html. WebLogic Tuxedo Connector also provides domain-level failover and failback capabilities. For more information, see "Configuring Domains-level Failover and Failback" on page 3-5.

The load balancing and failover of the outbound requests from WebLogic Server depend on the WebLogic Server EJB and MDB.

Example Clustered WebLogic Tuxedo Connector Configuration

The following configuration provides an example of WebLogic Tuxedo Connector in a clustered environment. The cluster consists of an administration server (`wtcAServer`) and three managed servers (`wtcMServer1`, `wtcMServer2`, `wtcMServer3`). Each managed server has a configured `WTCServer` MBean that contains the same service (TOUPPER) in the `WTCImport` MBean.

Listing 4-1 Example Clustered WebLogic Tuxedo Connector Configuration

```
<Domain Name="wtcDomain" >
<Security Name="wtcDomain" Realm="mysecurity"/>
<Realm Name="mysecurity" FileRealm="myrealm"/>
<FileRealm Name="myrealm"/>

<Cluster Name="wtcCluster" MulticastAddress="239.0.0.20"
     MulticastPort="7700" MulticastTTL="1"/>


<Security GuestDisabled="false"/>

<Server Name="wtcAServer" NativeIOEnabled="true" ListenPort="3472"
```

```
     ListenAddress="mymachine" TunnelingEnabled="true" >
<SystemDataStore ListenPort="7555" ListenAddress="mymachine" />
</Server>

<Server Name="wtcMServer1" Cluster="wtcCluster"
NativeIOEnabled="true" ListenPort="7701" ListenAddress="mymachine"
TunnelingEnabled="true" > </Server>

<Server Name="wtcMServer2" Cluster="wtcCluster"
NativeIOEnabled="true" ListenPort="7702" ListenAddress="mymachine"
TunnelingEnabled="true" > </Server>

<Server Name="wtcMServer3" Cluster="wtcCluster"
NativeIOEnabled="true" ListenPort="7703" ListenAddress="mymachine"
TunnelingEnabled="true" > </Server>


<WTCServer Name="WTCServer1" Targets="wtcMServer1">
<WTCExport EJBName="tuxedo.services.TOLOWERHome"
     LocalAccessPoint="WDOM1" Name="exp0" ResourceName="TOLOWER"/>
<WTCImport LocalAccessPoint="WDOM1" Name="imp0"
     RemoteAccessPointList="TDOM2,TDOM1" ResourceName="TOUPPER"/>
<WTCLocalTuxDom AccessPoint="WDOM1" AccessPointId="WDOM1"
     BlockTime="30000" ConnectionPolicy="ON_DEMAND"
     NWAddr="//mymachine:20401" Name="ltd0" Security="NONE"/>
<WTCRemoteTuxDom AccessPoint="TDOM1" AccessPointId="TDOM1"
     LocalAccessPoint="WDOM1" NWAddr="//123.123.123.123:20301"
     Name="rtd0"/>
<WTCRemoteTuxDom AccessPoint="TDOM2" AccessPointId="TDOM2"
     LocalAccessPoint="WDOM1" NWAddr="//123.123.123.123:20302"
     Name="rtd1"/>
</WTCServer>

<WTCServer Name="WTCServer2" Targets="wtcMServer2" >
<WTCExport EJBName="tuxedo.services.TOLOWERHome"
     LocalAccessPoint="WDOM2" Name="exp1" ResourceName="TOLOWER"/>
<WTCImport LocalAccessPoint="WDOM2" Name="imp1"
     RemoteAccessPointList="TDOM1,TDOM2" ResourceName="TOUPPER"/>
<WTCLocalTuxDom AccessPoint="WDOM2" AccessPointId="WDOM2"
     BlockTime="30000" ConnectionPolicy="ON_DEMAND"
     NWAddr="//mymachine:20402" Name="ltd1" Security="NONE"/>
<WTCRemoteTuxDom AccessPoint="TDOM1" AccessPointId="TDOM1"
     LocalAccessPoint="WDOM2" NWAddr="//123.123.123.123:20301"
     Name="rtd2"/>
<WTCRemoteTuxDom AccessPoint="TDOM2" AccessPointId="TDOM2"
     LocalAccessPoint="WDOM2" NWAddr="//123.123.123.123:20302"
```

```
     Name="rtd3"/>
</WTCServer>

<WTCServer Name="WTCServer3" Targets="wtcMServer3" >
<WTCExport EJBName="tuxedo.services.TOLOWERHome"
     LocalAccessPoint="WDOM3" Name="exp2" ResourceName="TOLOWER"/>
<WTCImport LocalAccessPoint="WDOM3" Name="imp2"
     RemoteAccessPointList="TDOM1,TDOM2" ResourceName="TOUPPER"/>
<WTCLocalTuxDom AccessPoint="WDOM3" AccessPointId="WDOM3"
     BlockTime="30000" ConnectionPolicy="ON_DEMAND"
     NWAddr="//mymachine:20403" Name="ltd2" Security="NONE"/>
<WTCRemoteTuxDom AccessPoint="TDOM1" AccessPointId="TDOM1"
     LocalAccessPoint="WDOM3" NWAddr="//123.123.123.123:20301"
     Name="rtd4"/>
<WTCRemoteTuxDom AccessPoint="TDOM2" AccessPointId="TDOM2"
     LocalAccessPoint="WDOM3" NWAddr="//123.123.123.123:20302"
     Name="rtd5"/>
</WTCServer>

</Domain>
```

How to Configure Inbound Requests from Tuxedo Domains

Load balancing and failover of inbound requests from Tuxedo depend on the Tuxedo domain DMCONFIG configuration.

Load Balancing

Note: For more information on Tuxedo Load Balancing http://e-docs.bea.com/tuxedo/tux80/atmi/intatm24.htm.

The following is a sample Tuxedo DMCONFIG that load balances from Tuxedo to clustered WTC. This configuration has three nodes in a WebLogic Server cluster. Each node has a single properly configured WebLogic Tuxedo Connector instance that provides an exported service that is accessible to the Tuxedo client.

```
*DM_IMPORT

TOUPPER LDOM=tuxedo_dom RDOM=WDOM1 LOAD=50

TOUPPER LDOM=tuxedo_dom RDOM=WDOM2 LOAD=50

TOUPPER LDOM=tuxedo_dom RDOM=WDOM3 LOAD=50
```

Fail Over

Notes: For more information on Tuxedo Configuring Failover and Failback in a Domains Environment at http://e-docs.bea.com/tuxedo/tux80/atmi/addomc23.htm.

The following is a sample Tuxedo DMCONFIG that uses a more sophisticated configuration that load balances between the WebLogic Server nodes as well as illustrate Tuxedo failover capability. The Tuxedo domain must be configured with a `CONNECTION_POLICY` of `ON_STARTUP` or `INCOMING_ONLY` to enable Domains-level failover/failback.

```
*DM_IMPORT

TOUPPER LDOM=tuxedo_dom RDOM=WDOM1,WDOM2,WDOM3 LOAD=50

TOUPPER LDOM=tuxedo_dom RDOM=WDOM2,WDOM3,WDOM1 LOAD=50

TOUPPER LDOM=tuxedo_dom RDOM=WDOM3,WDOM1,WDOM2 LOAD=50
```

5 Administration of CORBA Applications

Note: For more information on CORBA applications, see Tuxedo CORBA at http://e-docs.bea.com/tuxedo/tux80/interm/corba.htm.

The following sections provide information on how to administer and configure the WebLogic Tuxedo Connector to support Tuxedo CORBA clients and services.

- How to Configure WebLogic Tuxedo Connector for CORBA Service Applications
- How to Administer and Configure WebLogic Tuxedo Connector for Inbound RMI-IIOP
- How to Configure WebLogic Tuxedo Connector for Outbound RMI-IIOP

How to Configure WebLogic Tuxedo Connector for CORBA Service Applications

Note: For more information on how to configure your WebLogic Tuxedo Connector MBeans, see "Configuring WebLogic Tuxedo Connector for Your Applications" on page 2-3.

This section provides information on how to configure a `WTCServer` MBean to support a call to a Tuxedo CORBA server from a WebLogic Server EJB. Use the following steps to configure your `WTCServer` MBean:

1. Configure a `WTCLocalTuxDom` MBean for your WebLogic Server domain.
2. Configure a `WTCRemoteTuxDom` MBean for your Tuxedo CORBA domain.
3. Configure a `WTCImport` MBean.
   - Set `ResourceName` to *"//domain_id"* where *domain_id* is `DOMAINID` specified in the Tuxedo `UBBCONFIG` file of the remote Tuxedo domain where the object is deployed. The maximum length of this unique identifier for CORBA domains is 15 characters including the //.
   - `Set LocalAccessPoint` to the value of the `LocalAccessPoint` attribute of the `WTCRemoteTuxDom` MBean.
   - `RemoteAccessPointList` to the value of the `AccessPointId` attribute of the `WTCRemoteTuxDom` MBean.

For information on how to develop client applications that call a Tuxedo CORBA service using a WebLogic Server EJB, see the *WebLogic Tuxedo Connector Programmer's Guide* at {DOCROTT}/wtc_atmi/index.html.

Example WTCServer MBean and Tuxedo UBB Files

The following `WTCServer` MBean provides an example of how to configure the `WTCImport` MBean for a TUXEDO CORBA server.

Listing 5-1 Example WTCServer MBean for a CORBA Server Application

```
<WTCServer Name="WTCsimpappCNS"
     <WTCImport LocalAccessPoint="examples"
          Name="myImportedResources" RemoteAccessPointList="TUXDOM"
          ResourceName="//simpapp"/>
     <WTCLocalTuxDom AccessPoint="examples" AccessPointId="examples"
          ConnectionPolicy="ON_DEMAND" NWAddr="//123.123.123.123:5678"
          Name="myLoclTuxDom" Security="NONE"/>
     <WTCRemoteTuxDom AccessPoint="TUXDOM" AccessPointId="TUXDOM"
          LocalAccessPoint="examples" NWAddr="//123.123.123.123:1234"
          Name="myRTuxDom"/>
</WTCServer>
```

The following example Tuxedo UBB configuration file has a DOMAINID name of simpapp. The DOMAINID name is used in the ResourceName attribute of the WTCImport MBean.

Listing 5-2 Example Tuxedo UBB File for a CORBA Server Application

```
*RESOURCES
     IPCKEY     55432
     DOMAINID   simpapp
     MASTER     SITE1
     MODEL      SHM
     LDBAL      N
*MACHINES
     "YODA"
     LMID=SITE1
     APPDIR="your APPDIR"
     TUXCONFIG="APPDIR\tuxconfig"
     TUXDIR="your TUXDIR"
     MAXWSCLIENTS=10
*GROUPS
     SYS_GRP
          LMID=SITE1
          GRPNO=1
     APP_GRP
          LMID=SITE1
          GRPNO=2
*SERVERS
     DEFAULT:
          RESTART=Y
          MAXGEN=5
     TMSYSEVT
          SRVGRP=SYS_GRP
          SRVID=1
     TMFFNAME
          SRVGRP=SYS_GRP
          SRVID=2
          CLOPT="-A -- -N -M"
     TMFFNAME
          SRVGRP=SYS_GRP
          SRVID=3
          CLOPT= "-A -- -N"
     TMFFNAME
          SRVGRP=SYS_GRP
          SRVID=4
          CLOPT="-A -- -F"
     ISL
```

```
            SRVGRP=SYS_GRP
            SRVID=5
            CLOPT="-A -- -n <//your tux machine:2468>"
      cns
            SRVGRP=SYS_GRP
            SRVID=6
            CLOPT="-A --"
      DMADM SRVGRP=SYS_GRP SRVID=7
      GWADM SRVGRP=SYS_GRP SRVID=8
      GWTDOMAIN SRVGRP=SYS_GRP SRVID=9
      simple_server
            SRVGRP=APP_GRP
            SRVID=1
            RESTART = N
*SERVICES
```

How to Administer and Configure WebLogic Tuxedo Connector for Inbound RMI-IIOP

This section provides information on how to administer your application environment and configure WTCServer MBeans to enable Tuxedo CORBA objects to invoke upon EJBs deployed in WebLogic Server using the RMI-IIOP API.

- Configuring the WTCServer Mbean
- Administering the Tuxedo Application Environment

Configuring the WTCServer Mbean

Note: For more information on how to configure your WebLogic Tuxedo Connector MBeans, see "Configuring WebLogic Tuxedo Connector for Your Applications" on page 2-3.

Configure `WTCLocalTuxDom` and `WTCRemoteTuxDom` MBeans as needed for your environment. No special administration steps are required to enable Tuxedo CORBA objects to invoke upon EJBs deployed in WebLogic Server using the RMI-IIOP API.

Administering the Tuxedo Application Environment

Note: For more information on how to configure your Tuxedo application environment, see Tuxedo Administration Topics at http://e-docs.bea.com/tuxedo/tux80/interm/admin.htm.

You must perform some additional steps when configuring your Tuxedo application environment.

1. Set the TOBJADDR for your environment.

   Example: `//<hostname>:2468`

2. Register WebLogic Server (WLS) Naming Service in the Tuxedo domain's CosNaming namespace by entering the following command:

   ```
   cnsbind -o ior.txt your_bind_name
   ```

where

your_bind_name is the CosNaming service object name from your Tuxedo application.

The `ior.txt` file contains the URL of the WebLogic Server's domain Naming Service.

Listing 5-3 ior.txt File for iiop.ejb.stateless.server.tux Tuxedo Client Example

```
corbaloc:tgiop:myServer/NameService
```

where

myServer is your server name.

3. Modify the *DM_REMOTE_SERVICES of your Tuxedo domain configuration file. Replace your WebLogic Server service name, formerly the DOMAINID, with the name of your WebLogic Server.

Listing 5-4 Domain Configuration File

```
*DM_RESOURCES
     VERSION=U22

*DM_LOCAL_DOMAINS
     TDOM1 GWGRP=SYS_GRP
     TYPE=TDOMAIN
     DOMAINID="TDOM1"
     BLOCKTIME=20
     MAXDATALEN=56
     MAXRDOM=89

*DM_REMOTE_DOMAINS
     TDOM2 TYPE=TDOMAIN
     DOMAINID="TDOM2"

*DM_TDOMAIN
     TDOM1 NWADDR="<network address of Tuxedo domain:port>"
     TDOM2 NWADDR="<network address of WTC domain:port>"

*DM_REMOTE_SERVICES
"//myServer"
```

where

myServer is your server name

4. Load your modified domain configuration file using dmloadcf.

Guidelines About Using Your Server Name as an Object Reference

This section provides guidelines you need to remember when creating server names that are used as object references.

- The maximum field length Tuxedo accepts in the *DM_REMOTE_SERVICES section is 15 characters including the //. For example: If your server name is

examplesServer, your *DM_REMOTE_SERVICES object reference is //examplesServe.

- If you require muliple servers, the server names must be unique in the first 13 charactcers.
- You can use the complete name of your server name in the ior.txt file if it exceeds 13 characters. For example: corbaloc:tgiop:examplesServer/NameService

How to Configure WebLogic Tuxedo Connector for Outbound RMI-IIOP

Note: For more information on how to configure your WebLogic Tuxedo Connector MBeans, see "Configuring WebLogic Tuxedo Connector for Your Applications" on page 2-3.

This section provides information on how to enable WebLogic Server EJBs to invoke upon Tuxedo CORBA objects using the RMI-IIOP API. Use the following steps to modify your WTCServer MBean:

1. Configure a WTCLocalTuxDom MBean for your Local WLS Domain.
2. Configure a WTCRemoteTuxDom MBean for your Remote Tuxedo Domain. Outbound RMI-IIOP requires two additional elements: FederationURL and FederationName.
   - Set FederationURL to the URL for a foreign name service that is federated into the JNDI. This must be the same URL used by the EJB to obtain the initial context used to access the remote Tuxedo CORBA object.
   - Set FederationName to the symbolic name of the federation point.
3. Configure a WTCImport MBean.
   - Set ResourceName to *"//domain_id"* where *domain_id* is DOMAINID specified in the Tuxedo UBBCONFIG file of the remote Tuxedo domain where the object is deployed. The maximum length of this unique identifier for CORBA domains is 15 characters including the //.

- Set `LocalAccessPoint` to the value of the `LocalAccessPoint` element of the WTCRemoteTuxDom MBean.
- `RemoteAccessPointList` to the value of the `AccessPointId` element of the WTCRemoteTuxDom MBean.

For information on how to develop applications that use RMI-IIOP to call a Tuxedo service using a WebLogic Server EJB, see the *WebLogic Tuxedo Connector Programmer's Guide* at http://e-docs.bea.com/wls/docs61/wtc_atmi/index.html.

Example Outbound RMI-IIOP Configuration

The following `WTCServer` MBean provides an example of how to configure the WebLogic Tuxedo Connector for outbound RMI-IIOP.

Listing 5-5 Example WTCServer MBean for Outbound RMI-IIOP

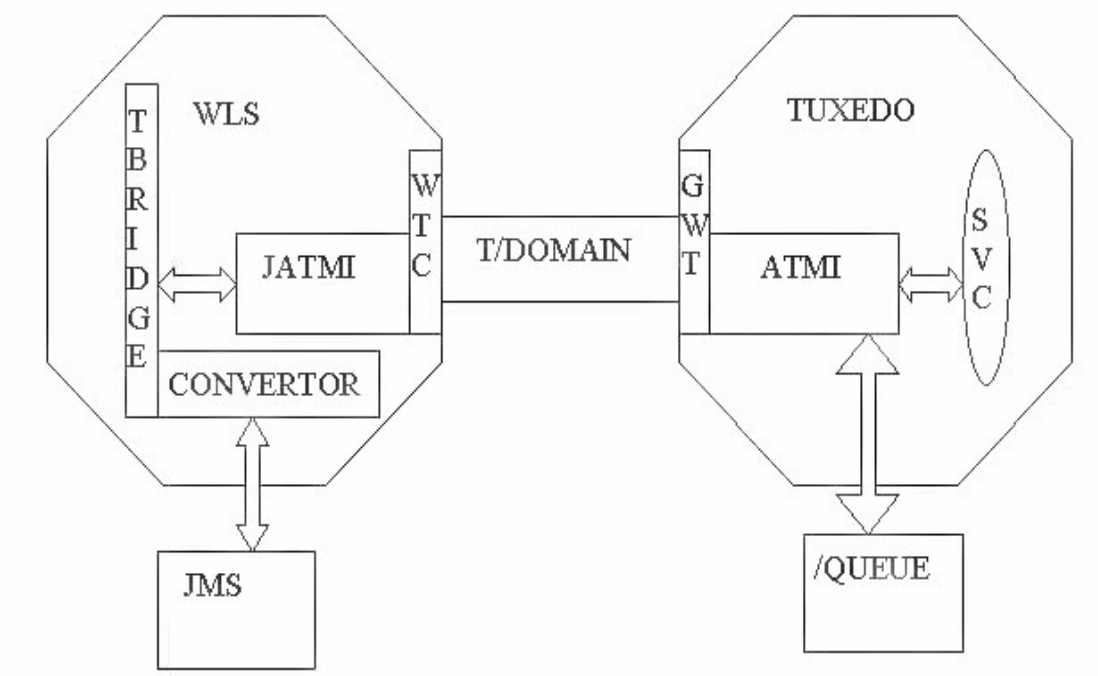
```
<WTCServer Name="WTCtrader">
   <WTCImport LocalAccessPoint="TDOM2" Name="myImportedResources"
   RemoteAccessPointList="TDOM1" ResourceName="//simpapp"/>
   <WTCLocalTuxDom AccessPoint="TDOM2" AccessPointId="TDOM2"
   ConnectionPolicy="ON_DEMAND" NWAddr="//123.123.123.123:5678"
   Name="myLoclTuxDom" Security="NONE"/>
   <WTCRemoteTuxDom AccessPoint="TDOM1" AccessPointId="TDOM1"
   FederationName="tuxedo.corba.remote"
   FederationURL="corbaloc:tgiop:simpapp/NameService"
   LocalAccessPoint="TDOM2" NWAddr="//123.123.123.123:1234"
   Name="myRTuxDom"/>
</WTCServer>
```

6 How to Configure the tBridge Message Interface

The following sections provide information on tBridge functionality and configuration.

Overview of the tBridge

The tBridge is a part of the WebLogic Tuxedo Connector that provides a bi-directional JMS interface for your WebLogic Server applications communicate to Tuxedo application environments. The transfer of messaging between the environments consists of JMS based messages containing text, Byte, or XML data streams used to invoke services on behalf of the client application.



The following features determine the functionality of the tBridge:

- Connectivity is determined by the configuration of the attributes in the `WTCtBridgeGlobal` and `WTCtBridgeRedirect` MBeans of a `WTCServer` MBean
- The tBridge uses Java Messaging Service (JMS) to provide an interface to a Tuxedo /Q or a Tuxedo service.
- The tBridge provides simple translation between XML and FML32 to provide connectivity to existing Tuxedo systems.

How tBridge connects JMS with Tuxedo

Note: All messages remain on the JMS queue until they have been acknowledged.

This section provides information on how JMS messages flow through the tBridge to Tuxedo queues and services.

1. A JMS client, such as a web enabled WLPI application, places a message to be processed by Tuxedo on a JMS Queue. If this message was part of a transaction, the transaction commits.
2. The message is removed from the JMS queue to be processed by the tBridge Converter.
3. The tBridge Converter checks the message type and converts supported JMS types to JATMI buffer types.
   - BytesMessage, TextMessage, XML are converted respectively to TypedCArray, TypedString, and TypedFML32. XML/FML translation is performed according to the `TranslateFML` attribute.
   - Translation errors are sent to the `wlsServerErrorDestination` queue and the message is acknowledged in the JMS session.
   - If an unrecognized JMS message is received: an appropriate error message is logged, the message is acknowledged, and then is discarded. This is considered a configuration error and the tBridge does not redirect the message to the error queue.
4. The converted message is sent to Tuxedo using the T/Domain gateway.
   - Messages with a redirect set to `JmsQ2TuxQ` use JATMI `tpenqueue` to deliver the message to a Tuxedo queue.
   - Messages with a redirect set to `JmsQ2TuxS` use JATMI `tpcall` to deliver the message to a Tuxedo service.
5. The `tpenqueue` is successful or `tpcall` is successful and the return results are placed in the replyQ. The message is acknowledged in the JMS session.
   - If the `tpenqueue` or `tpcall` fails, tBridge delivers the message to the `wlsServerErrorDestination` queue and the message is acknowledged in the JMS session. If a `wlsServerErrorDestination` queue is not configured, the message is discarded and the tBridge processes the next available unacknowledged message.

How tBridge connects Tuxedo to JMS

Note: tBridge uses a transaction to prevent the loss of messages while transferring messages from Tuxedo /Q to a JMS queue.

This section provides information on how Tuxedo messages flow through the tBridge to a JMS queue using the `TuxQ2JmsQ` redirect.

1. tBridge polls the Tuxedo queue for available messages.
2. A Tuxedo service places a message on a Tuxedo queue.
3. tBridge uses JATMI `tpdequeue` to forward the message from Tuxedo and places the message in the JMS queue.
   - If a message cannot be redirected to a JMS queue for any reason after the specified retries have been exhausted, the message is put into the `tuxErrorDestination` queue within the same queue space as the Tuxedo queue.
   - If the tBridge is not able to put the message into the `tuxErrorDestination` queue for any reason, an error is logged and the message is lost.
   - If the `tuxErrorDestination` queue is not specified, the message is lost.

tBridge Limitations

The tBridge has the following limitations:

- Transactions are not used when retrieving messages from the JMS location and placing them on the Tuxedo queue or invoking a Tuxedo service.
- tBridge is thread intensive. A thread is used to transport each message from JMS queue to Tuxedo. A polling thread is required to monitor the configured Tuxedo queue.
- The XML/FML translator is intended to construct simple message structures. For more information on XML to FML conversion see, "FML32 Considerations" on page 7-7.

WTCServer MBean Configuration for tBridge

The WebLogic Tuxedo Connector tBridge connectivity is determined by configuring the `WTCtBridgeGlobal` and `WTCtBridgeRedirect` MBeans which contain the necessary information to establish a connection to Tuxedo.

Starting the tBridge

The tBridge is started as part of the WebLogic Server application environment if the the `WTCtBridgeGlobal` and `WTCtBridgeRedirect` MBeans of a WTCServer MBean are configured and the WTCServer MBean is deployed to a target server. Any configuration condition that prevents the tBridge from starting results in an error being logged.

Error Logging

WebLogic Tuxedo Connector errors are logged to the WebLogic Server error log.

tBridge Connectivity

Note: JMS message types: MapMessage, ObjectMessage, StreamMessage are not valid in WebLogic Tuxedo Connector. If one of these message types is received by the tBridge, a log entry is generated indicating this is an unsupported type and the message is discarded.

The tBridge establishes a one-way data connection between instances of a JMS queue and a Tuxedo /Q or a JMS queue and a Tuxedo service. This connection is represented by a WTCtBridgeRedirect MBean and provides a one-to-one connection between the identified points. Three types of connections can be configured. The following is a description of each of the connection types:

- `JmsQ2TuxQ`: Reads from a given JMS queue and transports the messages to the specified Tuxedo /Q.
- `TuxQ2JmsQ`: Reads from a Tuxedo /Q and transports the messages to JMS.
- `JmsQ2TuxS`: Reads from a given JMS queue, synchronously calls the specified Tuxedo service, and places the reply back onto a specified JMS queue.

Example Connection Type Configurations

The following sections provide example configurations for each connection type.

Example JmsQ2TuxQ Configuration

The following section provides example code for reading from a JMS queue and sending to Tuxedo /Q.

```
<WTCtBridgeRedirect
     Direction="JmsQ2TuxQ"
     Name="redir0"
     ReplyQ="RPLYQ"
     SourceName="weblogic.jms.Jms2TuxQueue"
     TargetAccessPoint="TDOM2"
     TargetName="STRING"
     TargetQspace="QSPACE"
     TranslateFML="NO"/>
```

The following section describes the components of the `JmsQ2TuxQ` configuration:

- The `Direction` connection type is *`JmsQ2TuxQ`*.
- `SourceName` specifies the name of the JMS queue to read is *`weblogic.jms.Jms2TuxQueue`*. The tBridge establishes a JMS client session to this queue using CLIENT_ACKNOWLEDGE semantics.
- `TargetAccessPoint` specifies the name of the access point is *`TDOM2`*.
- `TargetQspace` specifies the name of the Qspace is *`Qspace`*.

- `TargetName` specifies the name of the queue is *`STRING`*.
- `ReplyQ` specifies the name of a JMS reply queue is *`RPLYQ`*. Use of this queue causes `tpenqueue` to provide `TMFORWARD` functionality.
- `TranslateFML` set to NO specifies that no data translation is provided by the tBridge.

The following table provides information on JmsQtoTuxQ message mapping:

From: JMS Message Type	To: WebLogic Tuxedo Connector JATMI (Tuxedo)
BytesMessage	TypedCArray
TextMessage (translateFML = NONE)	TypedString
TextMessage (translateFML = FLAT)	TypedFML32

Example TuxQ2JmsQ Configuration

The following section provides example code for reading from a Tuxedo /Q and sending to a JMS queue.

```
<WTCtBridgeRedirect
     Direction="TuxQ2JmsQ"
     Name="redir1"
     SourceAccessPoint="TDOM2"
     SourceName="STRING"
     SourceQspace="QSPACE"
     TargetName="weblogic.jms.Tux2JmsQueue"
     TranslateFML="NO"/>
```

The following section describes the components of the `TuxQ2JmsQ` configuration:

- The `Direction` connection type is *`TuxQ2JmsQ`*.
- `TargetName` specifies the name of the JMS queue to read is *`weblogic.jms.Tux2JmsQueue`*.
- `SourceAccessPoint` specifies the name of the access point is *`TDOM2`*.
- `SourceQspace` specifies the name of the Qspace is *`Qspace`*.

- `SourceName` specifies the name of the queue is *`STRING`*.
- `TranslateFML` set to NO specifies that no data translation is provided by the tBridge.
- `TranslateFML` set to Flat specifies that the data is translated from FML to XML by the tBridge.

The following table provides information on TuxQ2JmsQ message mapping:

From: WebLogic Tuxedo Connector JATMI (Tuxedo)	To: JMS Message Type
TypedCArray	BytesMessage
TypedString (translateFML = NO)	TextMessage
TypedFML32 (translateFML = FLAT)	TextMessage
TypedFML (translateFML = FLAT)	TextMessage
TypedXML	TextMessage

Example JmsQ2TuxS Configuration

Note: For more information on XML/FML conversion, see Chapter 7, "Using FML with WebLogic Tuxedo Connector."

The following section provides example code for reading from a JMS queue, calling a Tuxedo service, and then writing the results back to a JMS queue.

```
<WTCtBridgeRedirect
     Direction="JmsQ2TuxS"
     Name="redir0"
     ReplyQ="weblogic.jms.Tux2JmsQueue"
     SourceName="weblogic.jms.Jms2TuxQueue"
     TargetAccessPoint="TDOM2"
     TargetName="TOUPPER"
     TranslateFML="FLAT"/>
```

The following section describes the components of the `JmsQ2TuxS` configuration:

- The `Direction` connection type is *`JmsQ2TuxS`*.

- SourceName specifies the name of the JMS queue to read is *weblogic.jms.Jms2TuxQueue*.
- TargetAccessPoint specifies the name of the access point is *TDOM2*.
- TargetName specifies the name of the queue is *TOUPPER*.
- ReplyQ specifies the name of the JMS reply queue is *weblogic.jms.Tux2JmsQueue*.
- TranslateFML set to *FLAT* specifies that when a JMS message is received, the message is in XML format and is converted into the corresponding FML32 data buffer. The message is then placed in a tpcall with arguments *TDOM2* and *TOUPPER*. The resulting message is then translated from FML32 into XML and placed on the weblogic.jms.Tux2JmsQueue.

The following table provides information on the JMSQ2TuxX message mapping:

JMS Message Type	WebLogic Tuxedo Connector JATMI (Tuxedo)	JMS Message Type
BytesMessage	TypedCArray	BytesMessage
TextMessage (translateFML = NONE)	TypedString	TextMessage
TextMessage (translateFML = FLAT)	TypedFML32	TextMessage

Priority Mapping

WebLogic Tuxedo Connector supports multiple tBridge redirect instances. In many environments, using multiple redirect instances significantly improves application scalability and performance. However, it does randomizes the order in which messages are processed. Although priority mapping does not guarantee ordering, it does provides a mechanism to react to messages based on an assigned importance. If the order of delivery must be guaranteed, use a single tBridge redirect instance.

Use priorityMapping to map priorities between the JMS and Tuxedo.

- JMS has ten priorities (0 - 9).
- Tuxedo/Q has 100 priorities (1 - 100).

This section provides a mechanism to map the priorities between the Tuxedo and JMS subsystems. There are two mapping directions:

- `JmstoTux`
- `TuxtoJms`

Defaults are provided for all values, shown below in pairs of `value:range`.

- The `value` specifies the given input priority.
- The `range` specifies a sequential group of resulting output priorities.

`JmstoTux`- 0:1 | 1:12 | 2:23 | 3:34 | 4:45 | 5:56 | 6:67 | 7:78 | 8:89 | 9:100

`TuxtoJms`- 1-10:0 | 11-20:1 | 21-30:2 | 31-40:3 | 41-50:4| 51-60:5 | 61-70:6 | 71-80:7 | 81-90:8 | 91-100:9

For this configuration, a JMS message of priority 7 is assigned a priority of 78 in the Tuxedo /Q. A Tuxedo /Q with a priority of 47 is assigned a JMS priority of 4.

Error Queues

When tBridge encounters a problem retrieving messages from Tuxedo Queue or JMS Queue after the retry interval:

- The information is logged.
- The message is saved in the error queue if it is configured.

wlsServerErrorDestination

The `wlsErrorDestination` queue is used if a JMS message cannot be properly delivered due to Tuxedo failure or a translation error.

Unsupported Message Types

If an unrecognized JMS message is received, an appropriate error message is logged and the message is discarded. This is considered a configuration error and the tBridge does not redirect the message to the error queue.

tuxErrorQueue

The `tuxErrorQueue` is the failure queue for the JATMI primitive `tpdequeue` during a `TuxQ2JmsQ` redirect.

Limitations

The tBridge error queues have the following limitations:

- `TuxErrorDestination` can be specified only once. Any error queue name associated with the `ErrorDestination` implies that all the QSPACEs have the same error queue name available.
- When there is an error, the message is put back in the source QSPACE. Assuming the QSPACE is corrupted or full, subsequent messages would be lost.
- There is no way to specify to drop messages on error. All messages are received or none are received.
- Information about the error is only available in the server log.

CHAPTER

7 Using FML with WebLogic Tuxedo Connector

The following sections discuss the Field Manipulation Language (FML) and describe how the WebLogic Tuxedo Connector uses FML.

- Overview of FML
- The WebLogic Tuxedo Connector FML API
- FML Field Table Administration
- tBridge XML/FML32 Translation

Overview of FML

Note: For more information about using FML, see Programming a BEA Tuxedo Application Using FML at http://e-docs.bea.com/tuxedo/tux80/atmi/fml01.htm.

FML is a set of java language functions for defining and manipulating storage structures called fielded buffers. Each fielded buffer contains attribute-value pairs in fields. For each field:

- The attribute is the field's identifier.

- The associated value represents the field's data content.
- An occurrence number.

There are two types of FML:

- FML16 based on 16-bit values for field lengths and identifiers. It is limited to 8191 unique fields, individual field lengths of 64K bytes, and a total fielded buffer size of 64K bytes.
- FML32 based on 32-bit values for the field lengths and identifiers. It allows for about 30 million fields, and field and buffer lengths of about 2 billion bytes.

The WebLogic Tuxedo Connector FML API

Note: The WebLogic Tuxedo Connector implements a subset of FML functionality. For example, `views` are not supported.

The FML application program interface (API) is documented in the `weblogic.wtc.jatmi` package included in the Javadocs for WebLogic Server Classes.

FML Field Table Administration

Field tables are generated in a manner similar to Tuxedo field tables. The field tables are text files that provide the field name definitions, field types, and identification numbers that are common between the two systems. To interoperate with a Tuxedo system using FML, the following steps are required:

1. Copy the field tables from the Tuxedo system to WebLogic Tuxedo Connector environment.

   For example: Your Tuxedo distribution contains a bank application example called `bankapp`. It contains a file called `bankflds` that has the following structure:

```
#Copyright (c) 1990 Unix System Laboratories, Inc.
#All rights reserved
#ident  "@(#) apps/bankapp/bankflds     $Revision: 1.3 $"
# Fields for database bankdb

# name                          number  type    flags   comments
ACCOUNT_ID                      110     long    -       -
ACCT_TYPE                       112     char    -       -
ADDRESS                         109     string  -       -
.

.

.
```

2. Convert the field table definition into Java source files. Use the `mkfldclass` utility supplied in the `weblogic.wtc.jatmi` package. This class is a utility function that reads a FML32 Field Table and produces a Java file which implements the FldTbl interface. There are two instances of this utility:

   - `mkfldclass`
   - `mkfldclass32`

   Use the correct instance of the command to convert the `bankflds` field table into FML32 java source. The following example uses mkfldclass.

   ```
   java weblogic.wtc.jatmi.mkfldclass bankflds
   ```

   The resulting file is called `bankflds.java` and has the following structure:

```
import java.io.*;
import java.lang.*;
import java.util.*;
import weblogic.wtc.jatmi.*;

public final class bankflds
        implements weblogic.wtc.jatmi.FldTbl
{
        /** number: 110  type: long */
        public final static int ACCOUNT_ID = 33554542;
        /** number: 112  type: char */
        public final static int ACCT_TYPE = 67108976;
        /** number: 109  type: string */
        public final static int ADDRESS = 167772269;
        /** number: 117  type: float */

.

.
```

.

3. Compile the resulting `bankflds.java` file using the following command:

   ```
   javac bankflds.java
   ```

   The result is a `bankflds.class` file. When loaded, the WebLogic Tuxedo Connector uses the class file to add, retrieve and delete field entries from an FML32 field.

4. Add the field table class file to your application CLASSPATH.

5. Update your `WTCServer` MBean.

- Update the `WTCResources` MBean to reflect the fully qualified location of the field table class file.
- Use the keywords required to describe the FML buffer type: fml16 or fml32.
- You can enter multiple field table classes in a comma separated list.

For example:

```
<WTCResources
     FldTbl16Classes="my.bankflds,your.bankflds,more.bankflds"
     Name="BankappResources"/>
```

6. Restart your WebLogic Server to load the field table class definitions.

Using the DynRdHdr Property for mkfldclass32 Class

WebLogic Tuxedo Connector provides a property that provides an alternate method to compile FML tables. You may need to use the `DynRdHdr` utility if:

- You are using very large FML tables and the `.java` method created by the mkfldclass32 class exceeds the internal Java Virtual Machine limit on the total complexity of a single class or interface.
- You are using very large FML tables and are unable to load the class created when compiling the `.java` method.

Use the following steps to use the `DynRdHdr` property when compiling your FML tables:

1. Convert the field table definition into Java source files.

```
java -DDynRdHdr=Path_to_Your_FML_Table weblogic.wtc.jatmi.mkfld
class32 userTable
```

The arguments for this command are defined as follows:

Attribute	Description
`-DDynRdHdr`	WebLogic Tuxedo Connector property used to compile an FML table.
`Path_to_Your_FML_Table`	Fully qualified path and the file name of your FML table.
`weblogic.wtc.jatmi.mkfldclass32`	This class is a utility function that reads an FML32 Field Table and produces a Java file which implements the FldTbl interface.
`userTable`	Name of the `.java` method created by the mkfldclass32 class.

2. Compile the *`userTable`* file using the following command:

   ```
   javac userTable.java
   ```

3. Add the *`userTable`*`.class` file to your application CLASSPATH.
4. Update the `WTCResources` MBean to reflect the fully qualified location of the *`userTable`*`.class` file.
5. Target your WTCServer. The *`userTable`*`.class` is loaded when the WTCServer service starts.

Once you have created the *`userTable`*`.class` file, you can modify the FML table and deploy the changes without having to manually create an updated *`userTable`*`.class`. When the WTCServer is started, Weblogic Tuxedo Connector will load the updated FML table using the location specified by the `WTCResources` MBean. If the *`Path_to_Your_FML_Table`* attribute changes, you will need to use the preceding procedure to update your *`userTable`*`.java` and *`userTable`*`.class` files.

tBridge XML/FML32 Translation

Note: The data type specified must be FLAT or NO. If any other data type is specified, the redirection fails.

The `TranslateFML` element of the WTCtBridgeRedirect MBean is used to indicate if FML32 translation is performed on the message payload. There are two types of FML32 translation: FLAT and NO.

FLAT

The message payload is translated using the WebLogic Tuxedo Connector internal FML32/XML translator. Fields are converted field-by-field values without knowledge of the message structure (hierarchy) and repeated grouping.

In order to convert an FML32 buffer to XML, the tBridge pulls each instance of each field in the FML32 buffer, converts it to a string, and places it within a tag consisting of the field name. All of these fields are placed within a tag consisting of the service name. For example, an FML32 buffer consisting of the following fields:

```
NAME          JOE
ADDRESS       CENTRAL CITY
PRODUCTNAME   BOLT
PRICE         1.95
PRODUCTNAME   SCREW
PRICE         2.50
```

The resulting XML buffer would be:

```
<FML32>
  <NAME>JOE</NAME>
  <ADDRESS>CENTRAL CITY</ADDRESS>
  <PRODUCTNAME>BOLT</PRODUCTNAME>
  <PRODUCTNAME>SCREW</PRODUCTNAME>
  <PRICE>1.95</PRICE>
  <PRICE>2.50</PRICE>
</FML32>
```

NO

No translation is used. The tBridge maps a JMS TextMessage into a Tuxedo TypedBuffer (TypedString) and vice versa depending on the direction of the redirection. JMS BytesMessage are mapped into Tuxedo TypedBuffer (TypedCarray) and vice versa.

FML32 Considerations

Remember to consider the following information when working with FML32:

- For XML input, the root element is required but ignored.
- For XML output, the root element is always <FML32>.
- The field table names must be loaded as described in "FML Field Table Administration" on page 7-2.
- The tBridge translator is capable of only "flat" or linear grouping. This means that information describing FML32 ordering is not maintained, therefore buffers that contain a series of repeating data could be presented in an unexpected fashion. For example, consider a FML32 buffer that contains a list of parts and their associated price. The expectation would be PART A, PRICE A, PART B, PRICE B, etc. however since there is no structural group information contained within the tBridge, the resulting XML could be PART A, PART B, etc., PRICE A, PRICE B, etc.
- When translating XML into FML32, the translator ignores blank values. For example, `<STRING></STRING>` is skipped in the resulting FML32 buffer.
- Embedded FML is not supported in this release.
- TypedCArray is not supported for FML to XML conversion. Select from the following list of supported field types:
  - SHORT
  - LONG
  - CHAR
  - FLOAT

- DOUBLE
- STRING
- INT (FML32)
- DECIMAL (FML32)

- If you have TypedCArray in your FML, encode to TypedString and decode the XML to TypedCArray.
- If you need to pass binary data, encode to a field type of your choice and decode the XML on the receiving side.

CHAPTER

8 Connecting WebLogic Process Integrator and Tuxedo Applications

Note: For more information on how to integrate applications, see BEA WebLogic Integration at http://e-docs.bea.com/wlintegration/v2_0/index.html.

The WebLogic Tuxedo Connector tBridge provides the necessary infrastructure for WebLogic Process Integrator users to integrate Tuxedo applications into their business workflows. The following sections discuss WebLogic Process Integrator - Tuxedo integration using the WebLogic Tuxedo Connector.

- Synchronous WebLogic Process Integrator-to-Tuxedo Connectivity
- Synchronous Non-Blocking WebLogic Process Integrator-to-Tuxedo Connectivity
- Asynchronous WebLogic Process Integrator-to-Tuxedo Connectivity
- Asynchronous Tuxedo /Q-to-WebLogic Process Integrator Connectivity
- Bi-directional Asynchronous Tuxedo-to-WebLogic Process Integrator Connectivity

Synchronous WebLogic Process Integrator-to-Tuxedo Connectivity

WebLogic Process Integrator executes a blocking invocation against a Tuxedo service using a JATMI EJB. This process consists of three parts:

- Defining WebLogic Process Integrator Business Operations.
- Invoking an eLink Adapter.
- Defining WebLogic Process Integrator Exception Handlers.

Defining Business Operations

Define WebLogic Process Integrator Business Operations for the JATMI methods to be used:

- TypedFML32 buffer manipulation methods.
- Use the JATMI `tpcall()` method.

  Example: out_buffer = tpcall (service_name, in_buffer, flags)

Invoking an eLink Adapter

Invoke an eLink adapter from a WebLogic Process Integrator process flow:

- Build TypedFML32 request buffers using defined Business Operations.
- Using the defined Business Operation invoke the JATMI `tpcall()` method specifying the service name.
- Process TypedFML32 response buffers using defined Business Operations.

Define Exception handlers

Define WebLogic Process Integrator Exception handlers to process exceptions.

Synchronous Non-Blocking WebLogic Process Integrator-to-Tuxedo Connectivity

WebLogic Process Integrator sends a message to synchronously invoke a Tuxedo service:

- 1:1 relationship between JMS queue and the call to a Tuxedo service.
- 1:1 relationship between the response from the Tuxedo service and a JMS queue.
- WebLogic Process Integrator writes a message to JMS queue.
- Once the message is on the JMS queue then tBridge moves the message to the target Tuxedo service.
- The message is translated from/to XML/FML32.
- The response is written to the specified JMS reply queue.
- The WebLogic Process Integrator event node waits on the response queue for a response message.

Asynchronous WebLogic Process Integrator-to-Tuxedo Connectivity

WebLogic Process Integrator sends a guaranteed asynchronous message to a Tuxedo /Q:

- 1:1 relationship between JMS queue and Tuxedo /Q.
- WebLogic Process Integrator writes a message to JMS queue.
- Once the message is on the JMS queue then tBridge moves the message to the target Tuxedo /Q on a per message basis.

- Messages in error are forwarded to a specified JMS error queue:
  - Infrastructure errors.
  - XML/FML32 translation errors.

Asynchronous Tuxedo /Q-to-WebLogic Process Integrator Connectivity

Tuxedo /Q sends a guaranteed asynchronous message to WebLogic Process Integrator:

- 1:1 relationship between JMS queue and Tuxedo /Q.
- Tuxedo writes a message to Tuxedo /Q.
- Once the message is committed on Tuxedo /Q, the message is forwarded via the Tuxedo /T Domain Gateway to the WebLogic Tuxedo Connector tBridge and target JMS queue.
- Messages which cannot be forwarded from Tuxedo are enqueued on a Tuxedo /Q error queue.
- Messages in error are forwarded to a specified Tuxedo /Q error queue, including:
  - Infrastructure errors.
  - FML32/XML translation errors.
- A workflow is created that waits for the message on the JMS queue. It is defined in the Start workflow node or in the Event node of an existing workflow instance.

Bi-directional Asynchronous Tuxedo-to-WebLogic Process Integrator Connectivity

Tuxedo executes a blocking invocation of a WebLogic Process Integrator process flow. Use two asynchronous instances to connect from JMS to Tuxedo /Q and from Tuxedo /Q back to JMS.

CHAPTER

9 Troubleshooting The WebLogic Tuxedo Connector

The following sections provide WebLogic Tuxedo Connector troubleshooting information.

- Monitoring the WebLogic Tuxedo Connector
- Frequently Asked Questions

Monitoring the WebLogic Tuxedo Connector

The WebLogic Tuxedo Connector uses the WebLogic Server log file to record log information. To record log information you must:

- Set Trace Levels
- Enable Debug Mode

Set Trace Levels

Note: For more information about setting WebLogic Server properties, see "How to Set WebLogic Tuxedo Connector Properties" on page 2-7.

To enable tracing, update the `JAVA_OPTIONS` variable in your server start script to the to the desired level.

Example:

```
JAVA_OPTIONS=-Dweblogic.wtc.TraceLevel=100000
```

Use the following values to set the TraceLevel:

Value	Components Traced	Description
10000	TBRIDGE_IO	tBridge input and output
15000	TBRIDGE_EX	more tBridge information
20000	GWT_IO	Gateway input and output, including the ATMI verbs
25000	GWT_EX	more Gateway information
50000	JAMTI_IO	JAMTI input and output, including low-level JAMTI calls
55000	JAMTI_EX	more JAMTI information
60000	CORBA_IO	CORBA input and output
65000	CORBA_EX	more CORBA information
100000	All Components	information on all WebLogic Tuxedo Connector components

Enable Debug Mode

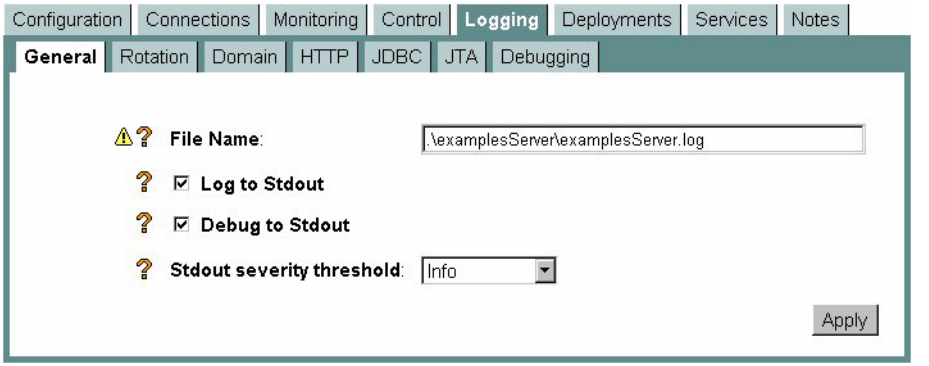
Use the following procedure to specify that trace information is written to the log file:

1. Click the Server node in the left pane.
2. Select your server in the left pane.
3. Select the Logging tab.
4. In the General tab:

a. Check **Debug to Stdout**

b. Set **Stdout severity threshold** to **Info**.

Figure 9-1 Setting Debug Mode



Frequently Asked Questions

This section provides solutions to common user questions.

What does this EJB Deployment Message Mean?

When I build the `simpserv` example, I get the following error:

```
<date> <Error> <EJB> <EJB Deployment: Tolower has a class
weblogic.wtc.jatmi.tpserviceHome which is in the classpath. This class should only
be located in the ejb-jar file.>
```

This error message can be ignored for this release of the WebLogic Tuxedo Connector. The EJB wants all of the interfaces for an EJB call in the EJB jar file. However, some interfaces for the WebLogic Tuxedo Connector are implemented through the CLASSPATH, and the compiler throws an exception. When the EJB is deployed, the compiler complains that the EJB cannot be redeployed because some of its classes are found in the CLASSPATH.

How do I Start the Connector?

Previous releases of the connector used a WebLogic Server Startup class to start a WebLogic Tuxedo Connector session and a WebLogic Server Shutdown class to end a session. In WebLogic Server 7.0, WebLogic Tuxedo Connector does not use a Startup or a Shutdown class. WebLogic Tuxedo Connector sessions are managed using a WTCServer MBean.

- A WebLogic Tuxedo Connector session is started when a configured WTCServer MBean is assigned to a selected server.
- A WebLogic Tuxedo Connector session is ended by removing a WTCServer MBean from the WebLogic server or when you shutdown the WebLogic server.

How do I Start the tBridge?

The tBridge is started if the `WTCtBridgeGlobal` and `WTCtBridgeRedirect` MBeans of a WTCServer MBean are configured and the WTCServer MBean is assigned to a selected server

How do I Assign a WTCServer MBean to a Server?

The console displays an exception when I try to assign my WTCServer MBean to a server. What should I do?

Make sure you have a valid WTCServer MBean configured. Each WTCServer MBean must have 1 or more Local WLS Domains configured before it can be assigned to a server. Your server log will display the following:

```
<Apr 22, 2002 4:21:35 PM EDT> <Error> <WTC> <180101> <At least one
local domain has to be defined.>
```

How do I Resolve Connection Problems?

I'm having trouble getting a connection established between WebLogic Tuxedo Connector and Tuxedo. What should I do?

- Make sure you have started your Tuxedo server.
- Set the `TraceLevel` and enable Debug mode. Repeat the connectivity test and check the WebLogic Tuxedo Connector and Tuxedo log files for error messages.
- Avoid using machine names or localhost. Always use an IP address when specifying a network location.
- Check your `AclPolicy` and `CredentialPolicy` attributes. If your `AclPolicy` is `LOCAL`, you must register the remote domain `DOMAINID` as a WebLogic Server user. For more information, see "User Authentication" on page 3-11.
- If you are migrating from WebLogic Server 6.x and your applications use security, you need to set `PasswordKey` as a WebLogic Server property. For more information, see "How to Set WebLogic Tuxedo Connector Properties" on page 2-7.
- Check the WebLogic Tuxedo Connector configuration against the Tuxedo remote domain. The remote domain must match the name of a remote domain configured in WebLogic Tuxedo Connector.

  For example: If the name `simpapp` is configured in the Tuxedo DMCONFIG *DM_LOCAL_DOMAINS section, then this name must match the name in your `WTCRemoteTuxDom` MBean `AccessPointId` attribute.
- Request assistance from BEA Customer Support.

How do I Migrate from Previous Releases?

You must make some changes in your WebLogic Tuxedo Connector 6.x applications (including WebLogic Tuxedo Connector 1.0) to use them with WebLogic Server 7.0. For more information, see Upgrading WebLogic Server 6.x to Version 7.0 at http://e-docs.bea.com/wls/docs70/upgrade/upgrade6xto70.html.

CHAPTER

10 WebLogic Tuxedo Connector MBean Attributes

Mbeans are used to provide information to configure domains in the WebLogic Tuxedo Connector. The configuration attributes are analogous to the DM_MIB attributes and classes used for Tuxedo domains. The following sections contain reference information on MBean attributes used to configure the WebLogic Tuxedo Connector:

- WTCServerMBean
- WTCLocalTuxDomMBean
- WTCRemoteTuxDomMBean
- WTCExportMBean
- WTCImportMbean
- WTCPasswordMbean
- WTCResourcesMBean
- WTCtBridgeGlobalMBean
- WTCtBridgeRedirectMBean

WTCServerMBean

Parent WebLogic Tuxedo Connector MBean containing the interoperability attributes required for a connection between WebLogic Server and Tuxedo.

Attribute	Description
DeploymentOrder	Specifies the order of deployment. Ordering is with respect to other deployable units of the same class, e.g. EJB's and Web Applications. Deployments with the lowest DeploymentOrder are deployed first.
Exports	Provides information on services exported by a local domain using a WTCExportMBean.
Imports	Provides information on remote services imported by a local domain using a WTCImportMbean.
LocalTuxDoms	Provides information on local domains as they appears to a remote domain using a WTCLocalTuxDomMBean.
Name	The name of the WTCServer MBean.
Passwords	Provides configuration information for inter-domain authentication using a WTCPasswordMbean.
RemoteTuxDoms	Provides information on remote domains as they appear to a local domain using a WTCRemoteTuxDomMBean.
Resource	Provides information on global field table classes, view table classes, and application passwords for domains using a WTCResourcesMBean.
Targets	The name of the selected target server on which the WTCServer MBean is deployed.
Type	Type of WebLogic Server Mbean: WTCServer
tBridgeGlobal	Specifies global configuration information for the transfer of messages between WebLogic Server and Tuxedo using a WTCtBridgeGlobalMBean.

Attribute	Description
tBridgeRedirects	Specifies the source, target, direction, and transport of messages between WebLogic Server and Tuxedo using a WTCtBridgeRedirectMBean.

WTCLocalTuxDomMBean

The WTCLocalTuxDom Mbean provides a view of local domains as they appears to other domains. The following attributes describe MBean type WTCLocalTuxDom:

Attribute	Description
AccessPoint	Required. Name used to identify a domain in a WTCServer MBean. This name must be unique within the scope of WTCLocalTuxDom and WTCRemoteTuxDom AccessPoint names in a WTCServer MBean. Example: TDOM2
AccessPointId	Required. Specifies the connection principal name used to identify a domain when establishing a connection to another domain. The AccessPointId of a WTCLocalTuxDom MBean must match the corresponding DOMAINID in the *DM_REMOTE_DOMAINS section of your Tuxedo DMCONFIG file. Example: TDOM2
BlockTime	Optional. Specifies the maximum wait time (seconds) allowed for a blocking call. ■ Minimum value: 0 ■ Maximum value: 2147483647 ■ Default value: 60

Attribute	Description
CmpLimit	Optional. Specifies the compression threshold used when sending data to the remote domain. Application buffers larger than this size are compressed. ■ Minimum value: 0 ■ Maximum value: 2147483647. ■ Default value: 2147483647 bytes
ConnPrincipalName	**Note:** ConnPrincipalName is not supported in this release of WebLogic Server. Optional. Specifies the connection principal name identifier. This is the principal name for identifying a domain when establishing a connection to another domain. ■ This parameter only applies to domains of type TDOMAIN that are running BEA Tuxedo 7.1 or later software. ■ If this element is not specified, the connection principal name defaults to the AccessPointID element for this domain.

Attribute	Description
ConnectionPolicy	Optional. Specifies the conditions under which a local domain tries to establish a connection to a remote domain. Valid values for local domains are: ON_DEMAND, ON_STARTUP, or INCOMING_ONLY. ■ Default setting is ON_DEMAND ■ ON_DEMAND: A connection is attempted only when requested by either a client request to a remote service or an administrative connect command. ■ ON_STARTUP: A domain gateway attempts to establish a connection with its remote domain access points at gateway server initialization time. Remote services (services advertised in JNDI by the domain gateway for this local access point) are advertised only if a connection is successfully established to that remote domain access point. If there is no active connection to a remote domain access point, then the remote services are suspended. By default, this connection policy retries failed connections every 60 seconds. Use the MaxRetry and RetryInterval elements to specify application specific values. ■ INCOMING_ONLY: A domain gateway does not attempt an initial connection to remote domain access points at startup and remote services are initially suspended. The domain gateway is available for incoming connections from remote domain access points and remote services are advertised when the domain gateway for this local domain access point receives an incoming connection. Connection retry processing is not allowed.
Interoperate	**Note:** Tuxedo 6.5 does not have the required security infrastructure to support security mapping. If you require security features and use the WebLogic Tuxedo Connector, you will need to upgrade to Tuxedo 7.1 or higher. Optional. Specifies whether the local domain interoperates with remote domains that are based upon Tuxedo Release 6.5. Valid values for this parameter are: Yes, No. ■ Yes: Interoperate with Tuxedo 6.5 ■ No: Operates with domains Tuxedo 7.1 and higher. ■ Default value: No

Attribute	Description
MaxEncryptBits	**Note:** A MaxEncryptBits value of 40 only applies to domains that are running BEA Tuxedo 7.1 or higher. Optional. Specifies the maximum level encryption key length (in bits) used when establishing a network link for this domain.Valid values for this parameter are: 0, 40, 56, and 128. ■ Default value: 128 bits ■ A value of 0: No encryption used
MaxRetries	Optional. The number of times that a domain gateway tries to establish connections to remote domain access points. Use only when ConnectionPolicy is set to ON_STARTUP. ■ Minimum value: 0 ■ Maximum value: 2147483647 ■ Default value: 2147483647 ■ The value of the MaxEncryptBits attribute must be greater than or equal to the value of the MinEncrypBits attribute. Use the maximum value to retry processing until a connection is established. Use the minimum value to disable the automatic retry mechanism.
MinEncryptBits	**Note:** A MinEncryptBits value of 40 only applies to domains that are running BEA Tuxedo 7.1 or later software. Optional. Specifies the minimum level encryption key length (in bits) used when establishing a network link for this domain.Valid values for this parameter are: 0, 40, 56, and 128. ■ Default value: 0 bits ■ A value of 0: No encryption used ■ The value of the MinEncrypBits attribute must be less than or equal to the value of the MaxEncrypBits attribute. ■ If this minimum level of encryption cannot be met, the network link fails.

Attribute	Description
NWAddr	Required. The network address of the local domain gateway. Specify the TCP/IP address in one of the following formats: ■ //hostname:port_number ■ //#.#.#.#:port_number If hostname is used, the domain finds an address for hostname using the local name resolution facilities (usually DNS). If dotted decimal format is used, each # should be a number from 0 to 255. This dotted decimal number represents the IP address of the local machine. The port_number is the TCP port number at which the domain process listens for incoming requests. **Note:** When configuring the NWAddr for a T_DM_LOCAL_DOMAIN, the port number used should be different from any port numbers assigned to other WebLogic Server processes. Example: Setting the NWAddr to `//mymachine:7001` is not valid if the WebLogic Server listening port is assigned to `//mymachine:7001`.
Name	The name of the WTCLocalTuxDom MBean.
RetryInterval	Optional. The time (seconds) between automatic attempts to establish a connection to remote domain access points. Use only when ConnectionPolicy is set to ON_STARTUP. ■ Minimum value: 0 ■ Maximum value: 2147483647 ■ Default setting: 60

Attribute	Description
Security	**Note:** Tuxedo 6.5 users should set the security parameter to NONE. Optional. Security specifies the type of application security to be enforced. Valid values for this parameter are: NONE, APP_PW, or DM_PW. NONE is the default value. ■ NONE: No security is used. ■ APP_PW: Password security is enforced when a connection is established from a remote domain. The application password is defined in a WTCResources MBean. ■ DM_PW: Domain password security is enforced when a connection is established from a remote domain. Domain passwords are defined in WTCPassword MBeans.
Type	Type of WebLogic Server Mbean: WTCLocalTuxDom

WTCRemoteTuxDomMBean

The WTCRemoteTuxDom Mbean provides a view of remote domains as they appears to a local domain.The following attributes describe MBean type WTCRemoteTuxDom:

Attribute	Description
AccessPoint	Required. Name used to identify a domain in a WTCServer MBean. This name must be unique within the scope of WTCLocalTuxDom and WTCRemoteTuxDom AccessPoint names in a WTCServer MBean. Example: TDOM2

Attribute	Description
AccessPointId	Required. Specifies the connection principal name used to identify a domain when establishing a connection to another domain. The AccessPointId of a WTCRemoteTuxDom MBean must match the corresponding DOMAINID in the *DM_LOCAL_DOMAINS section of your Tuxedo DMCONFIG file. Example: TDOM2
AclPolicy	**Note:** If the Interoperate parameter is set to Yes, the AclPolicy is ignored. For more information see, "WTCRemoteTuxDomMBean" on page 10-8. Optional. Specifies the inbound access control list (ACL) policy toward requests from a remote domain. Valid values for this parameter are: LOCAL, GLOBAL. ■ LOCAL: The local domain modifies the identity of the service requests received from a given remote domain to the principal name specified in the local principal name for a given remote domain. ■ GLOBAL: The local domain passes the service request with no change in identity. ■ Default value: LOCAL
CmpLimit	Optional. Specifies the compression threshold used when sending data to the remote domain. Application buffers larger than this size are compressed. ■ Minimum value: 0 ■ Maximum value: 2147483647. ■ Default value: 2147483647 bytes
ConnPrincipalName	**Note:** ConnPrincipalName is not supported in this release of WebLogic Server. Optional. Specifies the connection principal name identifier. This is the principal name for identifying a domain when establishing a connection to another domain. ■ This parameter only applies to domains of type TDOMAIN that are running BEA Tuxedo 7.1 or later software. ■ If this element is not specified, the connection principal name defaults to the AccessPoint element for this domain.

Attribute	Description
ConnectionPolicy	Optional. Specifies the conditions under which a local domain tries to establish a connection to a remote domain. Valid values for local domains are: ON_DEMAND, ON_STARTUP, or INCOMING_ONLY. ■ Default setting is ON_DEMAND ■ ON_DEMAND: A connection is attempted only when requested by either a client request to a remote service or an administrative connect command. ■ ON_STARTUP: A domain gateway attempts to establish a connection with its remote domain access points at gateway server initialization time. Remote services (services advertised in JNDI by the domain gateway for this local access point) are advertised only if a connection is successfully established to that remote domain access point. If there is no active connection to a remote domain access point, then the remote services are suspended. By default, this connection policy retries failed connections every 60 seconds. Use the MaxRetry and RetryInterval elements to specify application specific values. ■ INCOMING_ONLY: A domain gateway does not attempt an initial connection to remote domain access points at startup and remote services are initially suspended. The domain gateway is available for incoming connections from remote domain access points and remote services are advertised when the domain gateway for this local domain access point receives an incoming connection. Connection retry processing is not allowed.
CredentialPolicy	**Note:** If the Interoperate parameter is set to Yes, the CredentialPolicy is ignored. For more information see, "WTCRemoteTuxDomMBean" on page 10-8. Optional. Specifies the outbound access control list (ACL) policy toward requests to a remote domain. Valid values for this parameter are: LOCAL, GLOBAL. ■ LOCAL: The remote domain controls the identity of service requests received from the local domain to the principal name specified in the local principal name for this remote domain. ■ GLOBAL: The remote domain passes the service request with no change. ■ Default value: LOCAL

Attribute	Description
FederationName	The context at which to federate to a foreign name service. If ommitted, the federation point is *tuxedo.domains*.
FederationURL	The URL for a foreign name service that is federated into the JNDI. ■ If ommitted, the WebLogic Tuxedo Connector assumes that there is a CosNaming server in the foreign domain. The WebLogic Tuxedo Connector federates to the CosNaming server using TGIOP. It is possible to federate to non-CORBA service providers.
LocalAccessPoint	Required. The local domain name from which a remote domain is reached. Example: TDOM2
MaxEncryptBits	**Note:** A MaxEncryptBits value of 40 only applies to domains that are running BEA Tuxedo 7.1 or higher. Optional. Specifies the maximum level encryption key length (in bits) used when establishing a network link for this domain.Valid values for this parameter are: 0, 40, 56, and 128. ■ Default value: 128 bits ■ A value of 0: No encryption used ■ The value of the MaxEncryptBits attribute must be greater than or equal to the value of the MinEncrypBits attribute.
MaxRetries	Optional. The number of times that a domain gateway tries to establish connections to remote domain access points. Use only when ConnectionPolicy is set to ON_STARTUP. ■ Minimum value: 0 ■ Maximum value: 2147483647 ■ Default value: 2147483647 ■ Use -1 to default to the Local WLS Domain domain value. Use the maximum value to retry processing until a connection is established. Use the minimum value to disable the automatic retry mechanism.

Attribute	Description
MinEncryptBits	**Note:** A MinEncryptBits value of 40 only applies to domains that are running BEA Tuxedo 7.1 or later software. Optional. Specifies the minimum level encryption key length (in bits) used when establishing a network link for this domain.Valid values for this parameter are: 0, 40, 56, and 128. ■ Default value: 0 bits ■ A value of 0: No encryption used. ■ The value of the MinEncrypBits attribute must be less than or equal to the value of the MaxEncrypBits attribute. ■ If this minimum level of encryption cannot be met, the network link fails.
NWAddr	Required. The network address of the local domain gateway. Specify the TCP/IP address in one of the following formats: ■ //hostname:port_number ■ //#.#.#.#:port_number If hostname is used, the domain finds an address for hostname using the local name resolution facilities (usually DNS). If dotted decimal format is used, each # should be a number from 0 to 255. This dotted decimal number represents the IP address of the local machine. The port_number is the TCP port number at which the domain process listens for incoming requests.
Name	The name of the WTCRemoteTuxDom MBean.
RetryInterval	Optional. The time (seconds) between automatic attempts to establish a connection to remote domain access points. Use only when ConnectionPolicy is set to ON_STARTUP. ■ Minimum value: 0 ■ Maximum value: 2147483647 ■ Default setting: 60 ■ Use -1 to default to the Local WLS Domain domain value.
TpUsrFile	Optional. Full path to user password file containing uid/gid information. This is the same file generated by the Tuxedo `tpusradd` utility on the remote domain. Username, uid and gid information must be included and valid for correct authorization, authentication, and auditing.

Attribute	Description
Type	Type of WebLogic Server Mbean: WTCRemoteTuxDom.

WTCExportMBean

WTCExportMBean provides information on services exported by a local domain.

- If not specified, all local domains accept requests to all of the services according to the default JNDI lookup rules.
- If the section is defined, use it to restrict the set of local services requested from a remote domain.

Attribute	Description
EJBName	Optional. The complete name of the EJB home interface to use when invoking a service. If this element is not specified, the default interface used is `tuxedo.services.servicenameHome`. Example: If the service being invoked is TOUPPER and the EJBName attribute is not specified, the home interface looked up in JNDI would be `tuxedo.services.TOUPPERHome`.
LocalAccessPoint	Required. The local access point name. Example: TDOM2
Name	The name of the WTCExport MBean.
RemoteName	Optional. The remote name of the service. If not specified, the ResourceName attribute is used.
ResourceName	The ResourceName attribute describes a exported service entry.
Type	Type of WebLogic Server Mbean: WTCExport

WTCImportMbean

The WTCImportMBean provides information on services imported and available on remote domains.

Attribute	Description
LocalAccessPoint	Specifies the local access point through which a service is offered. Example: TDOM2
Name	The name of the WTCImport MBean.
RemoteAccessPointList	A comma-separated failover list that identifies the remote domain access points through which a resource is imported. Example: TDOM3,TDOM4,TDOM5
RemoteName	The RemoteName for this resource. RemoteName is the remote name of the service. If not specified, this defaults to the ResourceName attribute.
ResourceName	The ResourceName attribute describes an imported service entry. The combination of the ResourceName, LocalAccessPoint and RemoteAccessPointList attributes must be unique among all objects of this type. Example: //simpapp
Type	Type of WebLogic Server Mbean: WTCImport

WTCPasswordMbean

The WTCPasswordMbean provides information for inter-domain authentication through access points of type TDOMAIN.

Attribute	Description
LocalAccessPoint	The name of the local domain access point to which the password applies. Example: TDOM2
LocalPassword	The encrypted local password as returned from the `genpasswd` utility. This password is used to authenticate connections between the local domain access point identified by LocalAccessPoint and the remote domain access point identified by RemoteAccessPoint.
LocalPasswordIV	The initialization vector used to encrypt the local password.
Name	The name of the WTCPassword MBean.
RemoteAccessPoint	The name of the remote domain access point to which the password applies. Example: TDOM3
RemotePassword	The encrypted remote password as returned from the `genpasswd` utility. This password is used to authenticate connections between the local domain access point identified by LocalAccessPoint and the remote domain access point identified by RemoteAccessPoint.
RemotePasswordIV	The initialization vector used to encrypt the remote password
Type	Type of WebLogic Server Mbean: WTCPasswords

WTCResourcesMBean

Use to specify global field table classes, view table classes, and application passwords for domains.

Attribute	Description
AppPassword	The application password for this resource. The application password as returned from the genpasswd utility. This Tuxedo application password is the encrypted password to be used to authenticate connections
AppPassword IV	The initialization vector used to encrypt the global password. It is returned from the genpasswd utility with the AppPassword.
FldTbl classes	The name of the FldTbl16 classes which are loaded via a class loader and added to a FldTbl array. The class names used are the fully qualified names of the desired classes. Use a comma-separated list to enter multiple classes.
FldTbl32 classes	The name of the FldTbl32 classes which are loaded via a class loader and added to a FldTbl array. The class names used are the fully qualified names of the desired classes. Use a comma-separated list to enter multiple classes.
Name	The name of the WTCResources MBean.
Type	Type of WebLogic Server Mbean: WTCResources
ViewTbl classes	The name of the ViewTbl classes which are loaded via a class loader and added to a ViewTbl array. The class names used are the fully qualified names of the desired classes. Use a comma-separated list to enter multiple classes.
ViewTbl32 classes	The names of the ViewTbl32 classes which are loaded via a class loader and added to a ViewTbl32 array. The class names used are the fully qualified names of the desired classes. Use a comma-separated list to enter multiple classes.

WTCtBridgeGlobalMBean

Note: The tBridge handles one or more redirections by starting a new thread for each redirection defined. At least one redirection must be specified or the tBridge fails and an error is logged.

The WTCtBridgeMbean provides global configuration information for the transfer of messages between WebLogic Server and Tuxedo.

Attribute	Description
allowNonStandardTypes	Optional. Specifies if non-standard data types are allowed to pass through the tBridge. Valid values for this parameter are: NO, YES. ■ NO: Non-standard data types are rejected and placed onto the specified error location. ■ YES: Non-Standard data types are placed on the target location as BLOBs with a tag indicating the original type. Standard data types are: ■ ASCII text (TextMessage, STRING) ■ BLOB (BytesMessage, CARRAY)
defaultReplyDeliveryMode	**Note:** If neither `defaultReplyDeliveryMode` is specified or `JMS_BEA_TuxGtway_Tuxedo_ReplyDeliveryMode` is set, the default semantics defined for Tuxedo are used by the Tuxedo/Q subsystem. Optional. Specifies the reply delivery mode to associate with a message when placing messages onto the target location. Use this element for messages being redirected to Tuxedo/Q from JMS when `JMS_BEA_TuxGtway_Tuxedo_ReplyDeliveryMode` is not set for a message. Valid values for this parameter are: PERSIST, NONPERSIST, DEFAULT.
deliveryModeOverride	**Note:** If `deliveryModeOverride` is not specified, then the message is placed on the target location with the same delivery mode specified from the source location. Optional. Specifies the delivery mode to use when placing messages onto the target location.The `deliveryModeOverride` value overrides any delivery mode associated with a message. Valid values for this parameter are: PERSIST, NONPERSIST.
jmsFactory	Required. Name of the JMS connection factory. Example: `weblogic.jms.ConnectionFactory`

Attribute	Description
JmstoTuxPriorityMap	Required. Used to specify the priority mapping direction from JMS into Tuxedo /Q message priority.
JndiFactory	Required. Name of the JNDI lookup factory. Example: `weblogic.jndi.WLInitialContextFactory`
Name	The name of the WTCtBridgeGlobal MBean.
retries	Optional. Specifies the number of attempts to redirect a message before putting the message in the specified error location and logging an error. Default value is 0. The value must be 0 or a positive integer.
retryDelay	During the `retryDelay`, the thread processing the message can not redirect any other messages. Optional. Specifies the minimum amount of time (seconds) to wait before redirecting a message. Default value is 10. The value must be a positive integer.
timeout	The timeout for the transaction used to retrieve the message from the source location is larger than timeout. This parameter reflects the effective length of the timeout for the entire redirection. Optional. Specifies the transaction timeout value (seconds) used to place a message on a target location. Required when `transactional` parameter is set to YES. Default value is 60. The value must be 0 or a positive integer. 0 indicates an infinite wait.
transactional	**Note:** This element is not currently supported. Optional. Specifies the use of transactions when retrieving messages from a source location and when placing messages on a target location. Valid parameter values are: YES, NO. YES: Transactions are used for both operations. NO: Transactions are not used for either operation.

Attribute	Description
tuxErrorQueue	Optional. Name of the Tuxedo queue used to store a message that cannot be redirected to a Tuxedo/Q source queue. This queue is in the same queue space as the source queue. If `tuxErrorDestination` is not specified, all messages that cannot be redirected are lost. If the message cannot be placed into the `tuxErrorQueue` for any reason, an error is logged and the message is lost.
tuxFactory	Required. Name of the Tuxedo Connection factory. Example: `tuxedo.services.TuxedoConnection`
TuxtoJmsPriorityMap	Required. Used to specify the priority mapping direction from Tuxedo /Q into JMS message priority.
Type	Type of WebLogic Server Mbean: WTCtBridgeGlobal
userID	Optional. Specifies a user identity for all messages handled by the tBridge for ACL checks when security options are configured. All messages assume this identity until the security/authentication contexts are passed between subsystems. Until security contexts are passed, there is no secure method to identify who generated a message recieved from the source location. The argument user may be specified as either a user name or a user identification number (uid).
wlsErrorDestination	Optional. Name of the location used to store WebLogic Server JMS messages when a message cannot be redirected. If a `wlsErrorDestination` is not specified, all messages that cannot be redirected are lost. If the message cannot be placed into the `wlsErrorDestination` for any reason, an error is logged and the message is lost.

WTCtBridgeRedirectMBean

Used to specify the source, target, direction, and transport of a message.

Attribute	Description
direction	Required. Specifies the direction of data flow. Valid parameter values are: JmsQ2TuxQ, TuxQ2JmsQ, JmsQ2TuxS. ■ JmsQ2TuxQ: From JMS to TUXEDO /Q. ■ TuxQ2JmsQ: From TUXEDO /Q to JMS. ■ JmsQ2TuxS: From JMS to TUXEDO Service reply to JMS.
metadataFile	**Note:** This element is not currently supported. Optional. URL of the metadata file.
Name	The name of the WTCtBridgeRedirect MBean.
ReplyQ	Optional. Name of a JMS queue specifically for the synchronous call to a TUXEDO service. The response is returned to the JMS ReplyQ.
SourceAccessPoint	A unique identifier, within the scope of local and remote entry names in the domain configuration, where the source is located.
SourceName	Required. Name of a JMS queue name, TUXEDO queue name, or a TUXEDO service name.
SourceQspace	Optional. Name of the Qspace for a source location.
TargetAccessPoint	A unique identifier, within the scope of local and remote entry names in the domain configuration, where the target is located.
TargetName	Required. Name of a JMS queue name, TUXEDO queue name, or a TUXEDO service name.
TargetQspace	Optional. Name of the Qspace for a target location.

Attribute	Description
translateFML	**Note:** The WLXT parameter value is currently not supported. Required. Specifies the type of XML/FML translation. Valid parameter values are: NONE, FLAT, WLXT. ■ NO: No data translation is performed. `TextMessage` maps into STRING and vice versa depending on the direction of transfer. `BytesMessage` maps into CARRAY and vice versa. All other data types cause the redirection to fail. ■ FLAT: The message payload is transformed using the WebLogic Tuxedo Connector built-in translator. ■ WLXT: The translation is done by the XML-to-nonXML WL XML Translator (WLXT). The `metadataFile` URL provided is passed to call the WLXT external methods to do the translation. ■ Default value: NO
Type	Type of WebLogic Server Mbean: WTCtBridgeRedirect