BEA Tuxedo®

Product Overview

Version 10.0
Document Released: September 28, 2007

Contents

1. Introducing BEA Tuxedo

2. BEA Tuxedo ATMI Core Components

3. BEA Tuxedo CORBA Components

4. BEA Tuxedo Web-Accessible Services

5. BEA Tuxedo Product Support and Resources

CHAPTER 1

Introducing BEA Tuxedo

The following sections describe the architecture and major features of the BEA Tuxedo product:

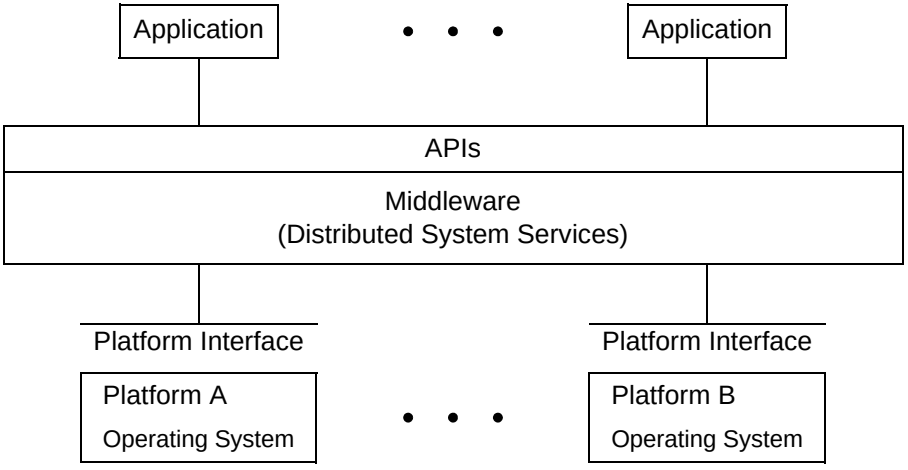
- About BEA SALT
- About BEA Jolt
- About BEA SNMP Agent

What is BEA Tuxedo?

BEA Tuxedo provides the framework, or middleware, for building scalable multi-tier client/server applications in heterogeneous (dissimilar), distributed environments that extend from the Web to the Enterprise. Using BEA Tuxedo, users can develop, manage, and deploy distributed applications independently of the underlying hardware, operating system, network, and database environment.

As indicated in the following figure, middleware consists of software services that exist *between* a client or server application *and* the operating system and network services on a system node in the network.

Figure 1-1 Use of Middleware



Middleware services provide a more functional set of application programming interfaces (API) than the operating system and network services. A main purpose of middleware services is to help solve application connectivity and interoperability problems.

BEA Tuxedo offers the following middleware services:

- An ATMI programming interface

  ATMI, for Application-to-Transaction Monitor Interface, is the main API for the Tuxedo system. It includes transaction management functions (routines, verbs); request/response, conversational, queuing, and publish-and-subscribe message-handling functions; service interface functions; and buffer management functions for distributed application communication.

- A CORBA programming interface

  CORBA, for Common Object Request Broker Architecture, is a language-independent, distributed-object model specified by the Object Management Group (OMG). The CORBA programming interface consists of C++ and Java ORBs. An ORB, or Object Request Broker, is a library that enables CORBA objects to locate and communicate with one another.

Note: The BEA Tuxedo CORBA Java client and BEA Tuxedo CORBA Java client ORB were deprecated in Tuxedo 8.1 and are no longer supported. All BEA Tuxedo CORBA Java client and BEA Tuxedo CORBA Java client ORB text references, associated code samples, should only be used to help implement/run third party Java ORB libraries, and for programmer reference only.

Technical support for third party CORBA Java ORBs should be provided by their respective vendors. BEA Tuxedo does not provide any technical support or documentation for third party CORBA Java ORBs.

- A high-performance transaction processing application server

  The transaction processing application server oversees all aspects of a distributed ATMI transaction, regardless of the systems or resource managers used. It provides the run-time engines for running ATMI transactions on top of ordinary computer hardware and operating systems.

- A high-performance object application server

  The object application server, based on the CORBA Object Transaction Service (OTS), combines the Tuxedo ATMI transaction processing technology with the BEA CORBA C++ ORB to provide high performance for distributed-object applications using transactions.

BEA Tuxedo includes the ATMI services and CORBA C++ objects needed for transaction management, security, message transport, administration and manageability, and XA-compliant database support for two-phase commit processing. It also includes a high-speed, highly reliable server-side message switch especially tuned for handling distributed transactions across many server machines.

A Brief History of the Tuxedo System

BEA Tuxedo is a proven, mature system spanning 20 years of continuous development and enhancement:

- In 1983, the Tuxedo system began as an applied, forward-looking work project within the Bell Laboratories division of AT&T. The target applications for the Tuxedo system were UNIX-based operations support systems within AT&T.
- In 1989, the Tuxedo system was transferred to the UNIX System Laboratories (USL) division of AT&T, and its client/server framework was offered as a commercial product.
- In 1993, the Tuxedo system was transferred to Novell, Inc., when Novell acquired USL in 1993.
- In 1996, BEA Systems, Inc., entered into an exclusive agreement with Novell to distribute and continue development of the Tuxedo system on a variety of computer platforms, including Windows and most UNIX systems.

Releases 1.0 Through 7.1

From release 1.0 in 1983 through release 7.1 in 2000, the Tuxedo system was extended and enhanced in a number of significant ways, always with the intent of making communication between client and server processes easier and more flexible. The Tuxedo system evolved into the de facto standard for open (open standard) online transaction processing (OLTP) solutions.

Release 4.0 introduced the *ATMI API* and *transaction processing*. Release 5.0 introduced the *Domains component*, which provided for the federation of Tuxedo applications and inter-application transaction processing. Release 7.1 introduced a *security plug-in architecture*, which allowed for the installation of third-party security systems.

Release 7.1 also introduced *multithreading and multicontexting*—ATMI functions that enabled programmers to write multithreaded and/or multicontexted application clients and servers—and *XML buffer support*—the ability to use extensible markup language (XML) typed buffers to exchange XML data within and between ATMI applications. In release 7.1, the BEA Jolt product was bundled with BEA Tuxedo for the first time.

For an overview of BEA Tuxedo ATMI, see Chapter 2, “BEA Tuxedo ATMI Core Components.”

Release 8.0

Release 8.0 introduced the BEA *CORBA API* and *CORBA Object Transaction Monitor (OTM)* capability. The CORBA OTM combined the advantages of a CORBA-compliant programming model with the proven power and reliability of the BEA Tuxedo core technology infrastructure.

For an overview of BEA Tuxedo CORBA, see Chapter 3, "BEA Tuxedo CORBA Components."

Release 8.1

Release 8.1 introduced the following features and enhancements:

- Localization enhancements

  Enables customers to install and interface with the Tuxedo system in English or Japanese.

- Multibyte character encoding

  Provides a new ATMI application typed buffer to handle multibyte character encoding.

- XML parser integration

  Incorporates the Apache Xerces C++ Version 1.7 parser into Tuxedo for use by customer applications to read and write XML data.

- Single point security administration

  Enables customers to use the BEA WebLogic Server Administration Console to administer security for both BEA Tuxedo and BEA WebLogic Server.

- Domain gateway performance improvement

  Improves the performance of the Tuxedo domain gateway process without any changes in the user interface.

- Remote domain connection policy

  Changes the behavior of the `ON_STARTUP` type connection policy of the Tuxedo domain gateway process to allow customers to selectively establish connections on a per remote domain basis.

- Domains keepalive

  Keeps interdomain connections open through firewalls during extended periods of no application activity and enables the Tuxedo domain gateway process to quickly detect interdomain connection failures.

- Multithreaded bridge

  Allows users to configure the Tuxedo Bridge process for multithreaded execution (as opposed to single-threaded execution) to improve Bridge performance.

- Parameter length expansion

  Increases the maximum allowable length of certain Tuxedo configuration parameters from 64 or 78 characters to 256 characters.

- Global maximum transaction timeout

  Adds a global maximum transaction timeout parameter to cap ATMI transaction timeout values that are excessively long.

- Enhanced CORBA C++ client ORB

  Enables CORBA C++ clients to participate in global transactions with WebLogic Server application servers in the same way that WebLogic Server T3 clients do.

In addition, both the BEA Jolt product and the BEA SNMP Agent product are bundled with BEA Tuxedo 8.1. For details about the new features and enhancements offered by BEA Tuxedo 8.1, see the *BEA Tuxedo Release Notes*.

Release 9.0

Release 9.0 introduced the following features and enhancements:

- Enhanced Web Services

  Provides XML schema and transformation (XML to and from FML) support. Also provides a Tuxedo service metadata repository that provides access to Tuxedo service definitions. It is designed to process interactive queries by developers and administrators during application development or modification, and is not designed for the processing of high volumes of automated queries during the application production phase.

- Domain Gateway performance improvements
- Infrastructure improvements in the following areas:
  - Timeout controls
  - Domain connection policies
  - CORBA IIOP client failover.
- Security

  - Cert-C PKI Plug-in security for data protection and non-repudiation
  - Kerberos authentication support

- Tuxedo .NET Workstation Client support

  The Tuxedo .NET workstation client provides customers with access to the Tuxedo system using the .NET Framework environment. It is implemented as a set of APIs and development utilities for developers.

In addition, both the BEA Jolt product and the BEA SNMP Agent product are bundled with BEA Tuxedo 9.0. For details about the new features and enhancements offered by BEA Tuxedo 9.0, see the *BEA Tuxedo Release Notes* and the What's New link on the BEA Tuxedo Documentation page.

Release 9.1

Release 9.1 introduced the following features and enhancements:

- Oracle RAC Support

  Supports clustering of machines that utilize replicated Oracle database services accessing the same Oracle database. For more information, see Using Tuxedo with Oracle Real Application Clusters (RAC) in *Setting Up a BEA Tuxedo Application*.

- Tuxedo .NET Workstation Client

  A facilitating tool that helps to efficiently develop Tuxedo .NET Workstation Client applications leveraging the benefit of Microsoft's .NET Framework. For more information, see Creating Tuxedo .NET Workstation Client Applications in *Using the Tuxedo .NET Workstation Client* .

- Remote Desktop Enhancement

  Allows Tuxedo to start up, be accessed and shut down using MS Windows Remote Desktop.

- Performance Enhancements
  - TDomain transaction performance enhancement
  - Memory usage enhancement
  - CORBA/Java interoperability hardening

- Customer Enhancements

- TPESYSTEM enhancement
- TPEXIT enhancement
- Increased RDOMS enhancement

Release 10.0

Release 10.0, the current release of the BEA Tuxedo product, introduces the following new features and enhancements:

- BEA Tuxedo System and Application Monitor (TSAM) Agent

  BEA TSAM provides comprehensive monitoring and reporting for BEA Tuxedo system and applications. It includes two components: BEA TSAM Agent and BEA TSAM Manager.

  The BEA TSAM Agent enables collection of various performance metrics for applications, including XA and non-XA transactions, services, system servers. The BEA TSAM Agent provides an open plug-in framework which can be used to customize performance metrics collection and send this information to management tools other than the BEA TSAM Manager.

  The BEA TSAM Agent can be used in conjunction with the BEA TSAM Manager. The BEA TSAM Manager provides a graphical user interface to correlate and aggregate performance metrics collected from one or more Tuxedo domains. It displays the information in *real time*.

  Note: A separate license is required to enable the BEA TSAM Agent.

- SSL Support for ATMI applications

  This feature provides support for SSL encryption over all network links in Tuxedo where LLE encryption is available. For more in formation, see Introducing ATMI Security, in "*Using Security in ATMI Applications*."

- MQ Adapter

  The MQ Adapter provides bi-directional, transactional connectivity to and from WebSphere MQSeries. For more information, see Running the Tuxedo MQ Adapter.

- Generic AUTHSVR

  Generic AUTHSVR is a new Tuxedo system server (GAUTHSVR) that enables Tuxedo users to be authenticated with LDAP based directory servers without need to write custom

code. For more information, see Implementing Single Point Security Administration, in "*Using Security in ATMI Applications*."

- DoS

  Provides Tuxedo TDomain domain gateway features used to defend against DoS attacks, and Tuxedo Domain improved password pair configuration flexibility. For more information, see see Introducing ATMI Security in "*Using Security in ATMI Applications*."

- Integrating ACUCOBOL in buildclient/buildserver

  In Tuxedo 10.0, buildclient/buildserver can accept COBOL source files and generate C stub code automatically using ACUCOBOL compiler version 6.2.0 or above.

- OpenLDAP for X.509 Certificate Lookup

  Tuxedo 10.0 PKI plug-in added support for OpenLDAP for X.509 certificate lookup. For more information, see Administering Security,Cert-C PKI Encryption Plug-In Configuration, Configure Certificate Lookup in "*Using Security in ATMI Applications.*"

Support for Industry Standards

The BEA Tuxedo system complies with the Open Group's X/Open standards, including support of the XA standard for two-phase commit processing, the X/Open ATMI API, and the X/Open Portability Guide (XPG) standards for language internationalization. BEA Tuxedo also supports the CORBA specification for distributed application development, as well as any relational database management system, object-oriented database management system, file manager, or queue manager.

The BEA Tuxedo system and ATMI together implement the X/Open distributed transaction processing (DTP) model of online transaction processing (OLTP). The DTP model ensures that work being done throughout a client/server application is *atomically* completed, meaning that all involved databases are updated properly if the work is successful, or all involved databases are "rolled-back" to their original state if the work fails.

Other standards supported by the BEA Tuxedo system include:

- Lightweight Directory Access Protocol (LDAP)—A set of protocols for accessing information directories. These directories can be physically distributed across multiple systems for access by many applications within an enterprise. LDAP is based on the standards contained within the X.500 standard, but is significantly simpler. And unlike X.500, LDAP supports TCP/IP, which is necessary for any type of Internet access. LDAP is an ideal way to publish certificates because it is closely coupled with the X.509 standard for certificates.

- X.509 Digital Certificates—A digital statement that associates a particular public key with a name or other attributes. The statement is digitally signed by a certificate authority. By trusting that authority to sign only true statements, you can trust that the public key belongs to the person named in the certificate. BEA Tuxedo public key security recognizes certificates that comply with X.509 version 3.0.
- Public-Key Cryptography Standard 7 (PKCS-7)—One of a set of Public-Key Cryptography Standards developed by RSA Laboratories in cooperation with an informal consortium, originally including Apple, Microsoft, DEC, Lotus, Sun and MIT. PKCS-7 defines a general syntax for messages that include cryptographic enhancements such as digital signatures and encryption. BEA Tuxedo public key security complies with the PKCS-7 standard.
- Secure Sockets Layer (SSL)—The standard protocol for establishing secure communications over the Internet (TCP/IP).

Support for Popular Platforms

A client/server application separates the calling (client) software and the called (server) software into separate programs. The advantage of a client/server application is that multiple client processes can interface with a single server process, where the processes do *not* need to run on the same host machine. Thus, clients and servers can run on hardware and software platforms suited to their particular functions. For example, clients can run on inexpensive platforms such as workstations or personal computers, and database management servers can run on platforms specially designed and configured to perform queries.

The BEA Tuxedo system has been ported to most popular *client* platforms, including Microsoft Windows Server and XP, and a variety of UNIX workstations. The BEA Tuxedo system has been ported to most popular *server* platforms, including Microsoft Windows Server, HP-UX, IBM AIX, and Sun Solaris.

For a complete list of supported platforms for BEA Tuxedo 10.0, see BEA Tuxedo 10.0 Platform Data Sheets on page A-1 in *Installing the BEA Tuxedo System*. For a complete list of supported platforms for earlier releases of BEA Tuxedo, see *Supported Platforms* at `http://www.bea.com/products/tuxedo/platforms.shtml`.

Support for Multiple Programming Models and Languages

BEA Tuxedo supports two programming models and five languages. The supported programming models are ATMI and CORBA. The supported programming languages are:

- C and COBOL—supported for ATMI application clients and servers
- C++—supported for ATMI application clients and CORBA C++ application clients and servers
- Java—supported for CORBA Java application clients and Jolt application clients

Note: The BEA Tuxedo CORBA Java client and BEA Tuxedo CORBA Java client ORB were deprecated in Tuxedo 8.1 and are no longer supported. All BEA Tuxedo CORBA Java client and BEA Tuxedo CORBA Java client ORB text references, associated code samples, should only be used to help implement/run third party Java ORB libraries, and for programmer reference only.

Technical support for third party CORBA Java ORBs should be provided by their respective vendors. BEA Tuxedo does not provide any technical support or documentation for third party CORBA Java ORBs.

Mission-Critical Software

ATMI and CORBA applications developed with BEA Tuxedo are *mission-critical*, meaning that they are reliable, scalable, secure, and manageable. Applications can grow as the company grows, and they continue running when various parts of the network fail. Applications can expand and contract as the demand requires.

Distributed Transaction Management

BEA Tuxedo specializes in managing transactions, on behalf of ATMI and CORBA applications, from their point of origin—typically on the client—across one or more server machines, and then back to the originating client. When a transaction ends, Tuxedo ensures that all the systems involved in the transaction are left in a consistent state. Tuxedo knows how to run transactions, route them across systems, load-balance their execution, and restart them after failures.

BEA Tuxedo ensures the integrity of data accessed across several sites or managed by different database products. It tracks transaction participants and supervises a two-phase commit protocol, ensuring that transaction commit and rollback are properly handled at each site.

X/Open XA and TX Compliance

The BEA Tuxedo system also coordinates the recovery of transactions in the event of site failure, network failure, or global resource deadlocks. The BEA Tuxedo system uses the X/Open XA interface for communicating with the various resource managers. This interface, proposed by Tuxedo developers and accepted by X/Open, is the standard interface for distributed transaction control between the transaction manager and resource managers.

The BEA Tuxedo system incorporates the X/Open TX interface for transaction demarcation, in addition to its own ATMI transaction management functions (routines, verbs). This interface allows an application writer to bracket a group of operations—define transaction boundaries—within an application such that all the operations will be done or none of them get done. That is, the transaction is either committed or rolled back as a single atomic unit of work, which keeps all involved databases synchronized, even if machine failures occur.

Transactions Documentation

For more information about transactions, see *Introducing BEA Tuxedo ATMI* and *Using CORBA Transactions*.

Scalability and Performance

In an enterprise environment, applications may need to support hundreds of execution contexts (where the context can be a thread or a process), tens of thousands of client applications, and millions of objects at satisfactory performance levels. Subjecting an application to exponentially increasing demands quickly reveals any resource shortcomings and performance bottlenecks in the application. Scalability is therefore an essential characteristic of BEA Tuxedo applications.

BEA Tuxedo enables distributed applications to scale is response to changing transaction loads by dynamically spawning and terminating servers (ATMI) or by dynamically activating and deactivating objects (CORBA) to meet the workload demands. BEA Tuxedo balances the workload among all the available services or objects to ensure that they are all evenly used.

Applications built on BEA Tuxedo can support a single client on a single server, or they can support tens of thousands of clients and thousands of servers without changing application code. As an application scales, the BEA Tuxedo system continues to provide end users with consistently high performance and good responsiveness.

For more information about scaling, see “Tuning a BEA Tuxedo ATMI Application” in *Administering a BEA Tuxedo Application at Run Time* and *Scaling, Distributing, and Tuning CORBA Applications*.

High Availability and Fault Management

In a distributed client/server environment, thousands of independent processors and processes must cooperate to run the application. Many malfunctions can happen. In spite of failures, BEA Tuxedo keeps the application running in the following ways:

- Ensures no single point of failure by providing replicated server groups that can continue when something breaks.
- Restores the running application to good condition after failures occur.

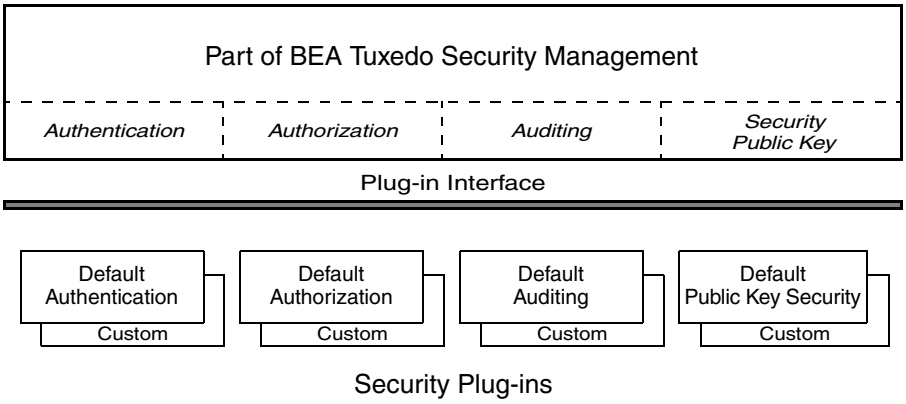
Ensuring constant access to e-business applications is a key feature of BEA Tuxedo. System components are constantly monitored for application, transaction, network, and hardware failures. When a failure occurs, BEA Tuxedo logically removes that component from the system, manages any necessary recovery procedures, and re-routes messages and transactions to surviving systems—all transparently to the end user and without disruption in service.

Security

BEA Tuxedo security includes authentication, authorization, and encryption to ensure data privacy when deploying BEA Tuxedo applications across networks. Two levels of encryption are supported: (1) network-level encryption using BEA Tuxedo's proprietary Link-Level Encryption (LLE) or Secure Sockets Layer (SSL); and (2) application-level encryption using the SSL protocol and public key encryption.

In order to integrate BEA Tuxedo security with other security systems, BEA Tuxedo provides the following security plug-in interface. The plug-in interface allows Tuxedo customers to independently define and dynamically add their own security plug-ins.

Figure 1-2 BEA Tuxedo Plug-in Security Architecture



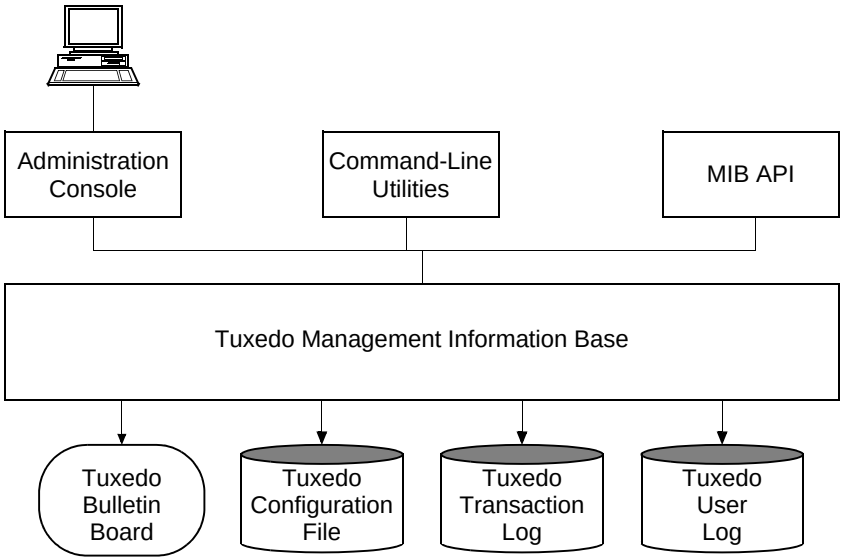
For more information on security in ATMI and CORBA applications, see *Using Security in ATMI Applications* and *Using Security in CORBA Applications*.

For information about security when interoperating with earlier releases of BEA Tuxedo software or BEA WebLogic Enterprise, or when interoperating with BEA WebLogic Server, see *BEA Tuxedo Interoperability*.

Management Tools

The BEA Tuxedo system gives administrators a choice of several methods for performing the same set of administrative tasks for either BEA Tuxedo ATMI or CORBA environments. The following figure illustrates the BEA Tuxedo tools available to write an application's configuration file and dynamically administer a BEA Tuxedo application during run time.

Figure 1-3 Simplified View of Administration Tools



In addition to using these tools to administrator BEA Tuxedo applications, administrators use these tools to perform fault-isolation and recovery tasks when application failures occur. BEA Tuxedo automatically recovers from many types of failures. However, some failures—often the most serious ones—require operator intervention to determine what has actually failed.

BEA Tuxedo Administration Console

The BEA Tuxedo Administration Console is a graphical user interface that enables administrators to perform most administration and configuration tasks for BEA Tuxedo applications. An administrator can display and change configuration information, determine the state of each component in the system, and obtain statistical information about items such as executed requests and queued requests.

The BEA Tuxedo Administration Console is implemented as a set of Java applets, which can run on most platforms that support a Java-capable Web browser. The server-side components of the BEA Tuxedo Administration Console reside on one of the server machines in a BEA Tuxedo application. To use the Console, an administrator must enter the URL of the server and download the Java applets.

For the BEA Tuxedo Administration Console startup procedure, see "Starting the BEA Tuxedo Administration Console" in *Installing the BEA Tuxedo System*. For information about how to use the BEA Tuxedo Administration Console, either access Help directly from the console or see *BEA Tuxedo Administration Console Online Help* . Also see "BEA Tuxedo Management Tools" in *Introducing BEA Tuxedo ATMI*.

Note: Limitation: The BEA Tuxedo Administration Console has not been updated to support any new features introduced after BEA Tuxedo release 7.1.

Command-Line Interface

Most of the functionality needed for dynamic modification of a BEA Tuxedo application is provided by the `tmadmin` and `tmconfig` commands. Most of the functionality needed for dynamic modification of a BEA Tuxedo Domains configuration is provided by the `dmadmin` command. Each of these commands is an interactive meta-command having many subcommands for performing various administrative tasks, including the modification of configuration entries while the system is running.

For details about these commands, see reference pages `tmadmin(1)`, `tmconfig, wtmconfig(1)`, and `dmadmin(1)`in *BEA Tuxedo Command Reference*. Also, see "BEA Tuxedo Management Tools" in *Introducing BEA Tuxedo ATMI*.

MIB Interface

The MIB interface is an application programming interface for directly accessing and manipulating system settings in the BEA Tuxedo management information bases. The interface allows administrators to have total control over Tuxedo applications. The MIB interface is powerful because it is implemented with the same APIs that Tuxedo developers use to write business-critical client/server applications.

There are MIB interfaces to administer the access control list, disk-based queues, Domains, events, core Tuxedo, and the workstation extension. The following are the corresponding MIB component names: `ACL_MIB`, `APPQ_MIB`, `DM_MIB`, `EVENT_MIB`, `TM_MIB`, and `WS_MIB`. Through the MIB interface, administrators control the application by programmatically querying the Tuxedo bulletin board for the current state of MIB objects, and then effecting administrative changes by either setting and resetting specific MIB values or creating new MIB objects.

The level of control available through the MIB interface really comes in handy in failover and fallback situations. The MIB programming interface is the only way to handle all the possible complications that can occur in a failover situation. During a failover, scripts can be used to

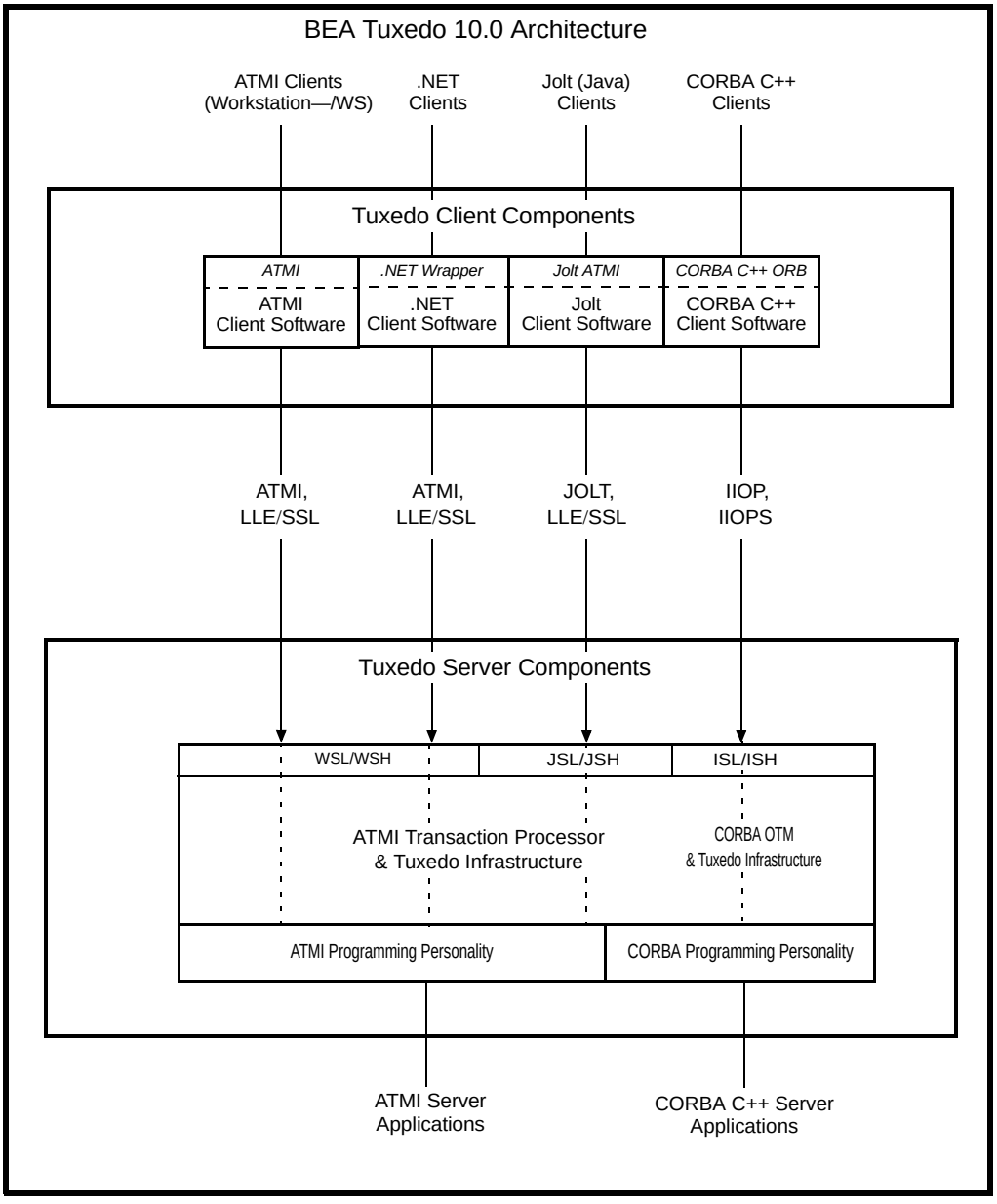
execute client MIB programs that perform specifics task such as shutting down and migrating server groups, and verifying the state of the application.

For details about the BEA Tuxedo MIBs, see reference pages `ACL_MIB(5)`, `APPQ_MIB(5)`, `DM_MIB(5)`, `EVENT_MIB(5)`, `MIB(5)`, `TM_MIB(5)`, and `WS_MIB(5)` in *BEA Tuxedo File Formats, Data Descriptions, MIBs, and System Processes Reference*; start with the `MIB(5)` reference page. Also, see "BEA Tuxedo Management Tools" in *Introducing BEA Tuxedo ATMI*.

Client and Server Components

The following figure identifies the BEA Tuxedo client and server components and shows the connectivity between the clients and servers. Only remote Tuxedo clients are shown in the figure.

Figure 1-4 BEA Tuxedo Client and Server Components



A *remote* Tuxedo client—ATMI (/WS), Jolt, or CORBA C++—interfaces with a Tuxedo server via a network connection and a pair of Tuxedo gateway processes: Workstation Listener/Handler (WSL/WSH), Jolt Server Listener/Handler (JSL/JSH), or IIOP Listener/Handler (ISL/ISH). A remote Tuxedo client may run on a machine that is *not* part of the Tuxedo server application (typically a workstation or personal computer), or the remote client may run on a machine that *is* part of the Tuxedo server application. For the latter case, the local operating system intercepts the messages destined for the network and redirects them to the destination process—the Tuxedo remote client or handler process—running locally.

A *native* Tuxedo client—a native ATMI client or a native CORBA C++ client—is co-located on a machine that is part of the Tuxedo server application and interfaces with a Tuxedo server via the Tuxedo infrastructure using interprocess communication. Native Jolt clients are *not* supported. These clients can only access a Tuxedo server via a pair of JSL/JSH gateway processes.

The following brief descriptions of other terms shown in the previous figure should prove helpful in understanding the connectivity between BEA Tuxedo clients and servers:

IIOP
: Internet Inter-ORB Protocol. A protocol used for communication between CORBA ORBs over the Internet (TCP/IP).

IIOPS
: IIOP layered over the SSL protocol.

LLE
: Link-Level Encryption. A BEA Tuxedo protocol for establishing data privacy over network links between BEA Tuxedo server machines.

SSL
: Secure Sockets Layer. The standard protocol for establishing secure communications over the Internet (TCP/IP).

BEA Tuxedo Client Components

The following client component software is included in the BEA Tuxedo 10.0 distribution:

- BEA ATMI Workstation (/WS) client software
- BEA Jolt client software
- BEA C++ client ORB including environmental objects

BEA Tuxedo Server Components

The following server component software is included in the BEA Tuxedo 10.0 distribution:

- BEA ATMI server software (includes native ATMI client software)
- BEA CORBA C++ server software (includes native CORBA C++ client software)
- BEA Jolt server software
- BEA TSAM Agent software
- BEA SALT software
- BEA SNMP Agent software
- BEA Tuxedo Administration Console software

Invocation Capabilities

The following table lists the invocation capabilities for an application built on the BEA system. A BEA Tuxedo application may span multiple BEA Tuxedo server machines and may provide ATMI services, CORBA objects, or both.

This component . . .	Can call a . . .	Through . . .
ATMI client *	ATMI service	WSL/WSH
Jolt client	ATMI service	JSL/JSH
CORBA C++ client *	CORBA C++ object	ISL/ISH
ATMI server	ATMI service	Tuxedo infrastructure
CORBA C++ object	CORBA C++ object	Tuxedo infrastructure
CORBA C++ object	ATMI service	Tuxedo infrastructure

* A native Tuxedo ATMI or CORBA C++ client does not use listener or handler gateway processes.

For information on how BEA Tuxedo 10.0 interoperates with older releases of BEA Tuxedo software, BEA WebLogic Enterprise, and third-party products, or how BEA Tuxedo 10.0 interoperates with BEA WebLogic Server, see *BEA Tuxedo Interoperability*.

Note: A BEA Tuxedo client cannot invoke another BEA Tuxedo client.

Client-to-Server Invocation Capabilities

The following client-to-server invocations are supported by a BEA Tuxedo application:

- A BEA ATMI client invoking a BEA Tuxedo service

  For more information about remote ATMI clients, see *Using the BEA Tuxedo ATMI Workstation Component*.

- A BEA Jolt client invoking a BEA Tuxedo service

  For more information about Jolt, see *Using BEA Jolt* and the *BEA Jolt API Javadoc* reference information.

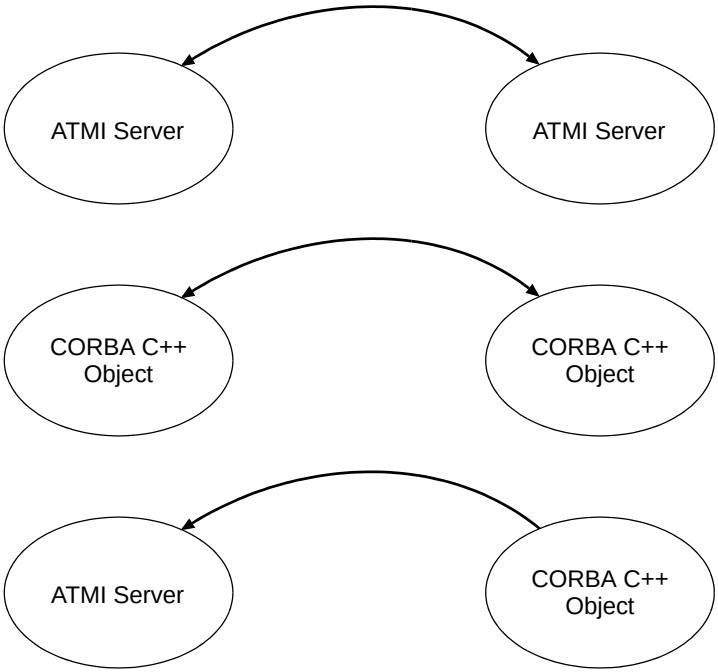
- A BEA CORBA C++ client invoking a BEA Tuxedo CORBA C++ object

  For details, see *Creating CORBA Client Applications*.

Server-to-Server Invocation Capabilities

The following figure shows the invocation capabilities between BEA Tuxedo ATMI and CORBA C++ application servers.

Figure 1-5 BEA Tuxedo ATMI and CORBA C++ Server Invocations



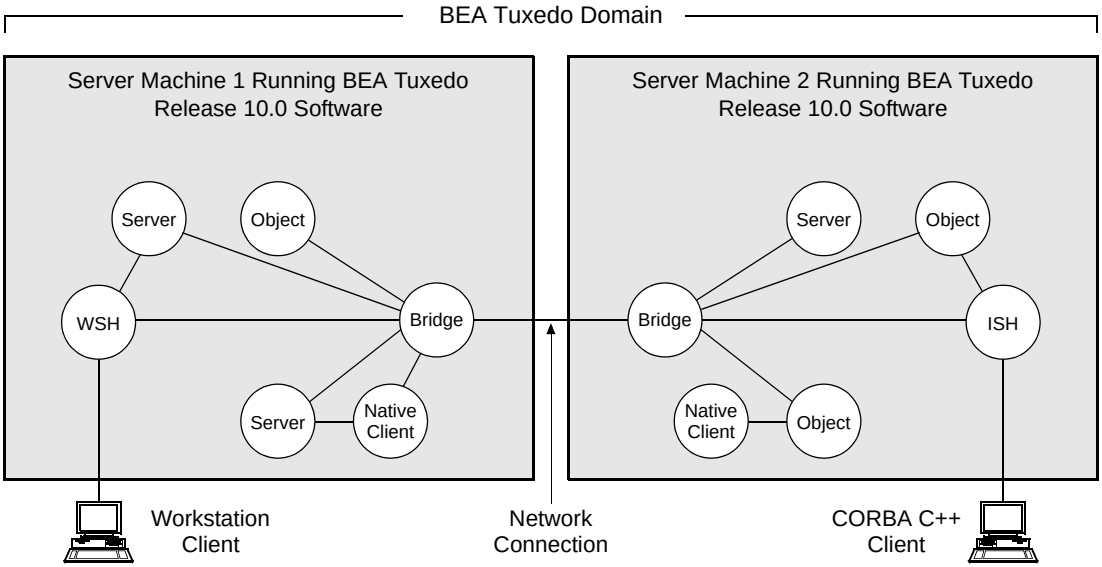
As shown in the figure, a CORBA C++ object can include ATMI calls to BEA Tuxedo services. For an example, see the Wrapper University sample application, available in the *Guide to the CORBA University Sample Applications*.

Domains

A BEA Tuxedo domain, or application, is defined and controlled by a single configuration file. A Tuxedo domain consists of many Tuxedo system processes, one or more application client processes, one or more application server processes, and one or more computer machines connected over a network. It is administered as a single unit.

A BEA Tuxedo domain may provide ATMI services, CORBA objects, or both. The Tuxedo domain in the following example contains a mixture of ATMI services and CORBA objects.

Figure 1-6 Simplified View of a BEA Tuxedo Domain



In BEA Tuxedo terminology, a *domain* is the same as an *application*—a business application; both terms are used as synonyms throughout the BEA Tuxedo user documentation. Examples of business applications currently running on Tuxedo are airline and hotel reservation systems, credit authorization systems, stock-brokerage systems, banking systems, and automatic teller machines.

For more information about Tuxedo domains, see "Important BEA Tuxedo Terms and Concepts" on page 2-1. For information about interconnecting Tuxedo domains, see "BEA Tuxedo Domains" on page 2-14.

About BEA TSAM

BEA TSAM (Tuxedo System and Application Monitor) provides comprehensive monitoring and reporting for BEA Tuxedo system and applications. It includes two components: BEA TSAM Agent and BEA TSAM Manager. The BEA TSAM Agent enables collection of various performance metrics for applications, including XA and non-XA transactions, services, system servers. BEA TSAM Manager provides graphical user interface to correlate and aggregate performance metrics collected from one or more Tuxedo domains and display it in real time.

The major features included in BEA TSAM are:

- Call path monitoring and analysis
- Service monitoring and statistics
- System server monitoring and statistics
- Transaction monitoring
- SLA Events

BEA TSAM Licensing

BEA TSAM 1.1, which is partly included in the BEA Tuxedo 10.0 distribution, remains a separately sold and licensed product.

BEA TSAM Documentation

For more information about BEA TSAM, see the following documentation: http://edocs.bea.com/tsam/docs11.

About BEA SALT

BEA SALT (Service Architecture Leveraging Tuxedo) is an add-on product that runs on top of Tuxedo. BEA SALT allows external Web service applications to invoke native Tuxedo services (inbound), and conversely, allows Tuxedo applications to invoke external Web services (outbound).

BEA SALT complies with most primary Web Services specifications: SOAP 1.1, SOAP 1.2, and WSDL 1.1, allowing BEA SALT to interoperate with other Web Service products and development toolkits. With BEA SALT, Tuxedo applications can easily integrate with Web Services applications.

BEA SALT Licensing

BEA SALT 2.0, which is included in the BEA Tuxedo 10.0 distribution, remains a separately sold and licensed product.

BEA SALT Documentation

For more information about BEA SALT, see the following documentation:

http://edocs.bea.com/salt/docs20.

About BEA Jolt

BEA Jolt is a Java class library and API that enables remote Java clients to access existing BEA Tuxedo ATMI services. It enables users to build client applets and applications that can remotely invoke Tuxedo ATMI services—such as application messaging, component management, and distributed transaction processing—using an ordinary Web browser.

BEA Jolt extends the functionality of existing Tuxedo ATMI applications to include intranet- and Internet-wide availability. BEA Jolt also enables BEA WebLogic Server to invoke Tuxedo ATMI services. For clarification, see "Making Tuxedo Services Web Accessible Through BEA Jolt" on page 4-8.

BEA Jolt Licensing

BEA Jolt 10.0, which is included in the BEA Tuxedo 10.0 distribution, is controllable through the BEA Tuxedo 10.0 license. BEA Jolt remains a separately sold and licensed product.

BEA Jolt Documentation

For more information about BEA Jolt, see the following documentation:

- *Using BEA Jolt*
- *Using BEA Jolt with BEA WebLogic Server*

About BEA SNMP Agent

BEA SNMP Agent for BEA Tuxedo enables SNMP-compliant network management frameworks to manage BEA Tuxedo systems and BEA Tuxedo applications. BEA SNMP Agent complies with the Simple Network Management Protocol version 1 (SNMPv1) specification.

BEA SNMP Agent provides the SNMP links from Tuxedo applications to SNMP-based system-management consoles. It also allows multiple SNMP agents and subagents—from any vendor—to operate on the same machine.

BEA SNMP Agent Licensing

BEA SNMP Agent 10.0, which is included in the BEA Tuxedo 10.0 distribution, is considered part of the Tuxedo 10.0 product. As such, the only product license required for BEA SNMP Agent is a valid BEA Tuxedo 10.0 license on the host Windows or UNIX system.

BEA SNMP Agent Documentation

For more information about BEA SNMP Agent, see the following documentation:

- *BEA Tuxedo SNMP Agent Administration Guide*
- *BEA Tuxedo SNMP Agent MIB Reference*

CHAPTER 2

BEA Tuxedo ATMI Core Components

The following sections describe the BEA Tuxedo ATMI components and the BEA Tuxedo infrastructure:

- Important BEA Tuxedo Terms and Concepts
- BEA Tuxedo ATMI Overview
- BEA Tuxedo ATMI Architecture
- BEA Tuxedo Transaction Processor and Infrastructure
- BEA Tuxedo Workstation
- BEA Tuxedo /Q
- BEA Tuxedo EventBroker
- BEA Tuxedo Domains

Important BEA Tuxedo Terms and Concepts

The following terms and concepts are fundamental to understanding the BEA Tuxedo system and applications built on the BEA Tuxedo system:

- Tuxedo domain

  A BEA Tuxedo domain, also known as a BEA Tuxedo application, is a set of Tuxedo system, client, and server processes administered as a single unit from a single Tuxedo configuration file. A Tuxedo domain consists of many system processes, one or more

application client processes, one or more application server processes, and one or more computer machines connected over a network. A BEA Tuxedo domain may provide ATMI services, CORBA objects, or both.

Note: A Tuxedo *domain* has the same meaning as a Tuxedo *application*.

- Tuxedo configuration file

  Each BEA Tuxedo domain is controlled by a configuration file in which installation-dependent parameters are defined. The text version of the configuration file is referred to as `UBBCONFIG`, although the configuration file may have any name, as long as the content of the file conforms to the format described on reference page `UBBCONFIG(5)` in *BEA Tuxedo File Formats, Data Descriptions, MIBs, and System Processes Reference*.

  The binary version of the `UBBCONFIG` file is referred to as `TUXCONFIG`. As with `UBBCONFIG`, the `TUXCONFIG` file may be given any name; the actual name is the device or system filename specified in the `TUXCONFIG` environment variable.

- Tuxedo master machine

  The master machine, or master node, for a BEA Tuxedo domain is a server machine containing the domain's `UBBCONFIG` file, and is designated as the master machine in the `RESOURCES` section of the `UBBCONFIG` file. Starting, stopping, and administering the one or more server machines in a Tuxedo domain is done through the master machine.

  The master machine for a Tuxedo domain also contains the master copy of the `TUXCONFIG` file. Copies of the `TUXCONFIG` file are propagated to every other server machine—referred to as *non-master machines*—in a Tuxedo domain whenever the Tuxedo system is booted on the master machine.

- Tuxedo bulletin board

  The BEA Tuxedo system uses the `TUXCONFIG` file to set up a *bulletin board* on each server machine in a Tuxedo domain. When a Tuxedo server process becomes active, it advertises the names of its services in the bulletin board. Some information in the bulletin board is global and is replicated on every server machine in the Tuxedo domain (for example, the names and locations of all servers offering a particular service). Other information is local and is visible only on the local bulletin board (for example, the actual number and type of client requests currently waiting on a local server request queue).

  The bulletin board provides location and namespace transparency within a Tuxedo domain. Location transparency means that Tuxedo client and server processes do not have to be aware of the location of a resource (ATMI service, CORBA C++ object) within the Tuxedo domain. Namespace transparency means that Tuxedo client and server processes

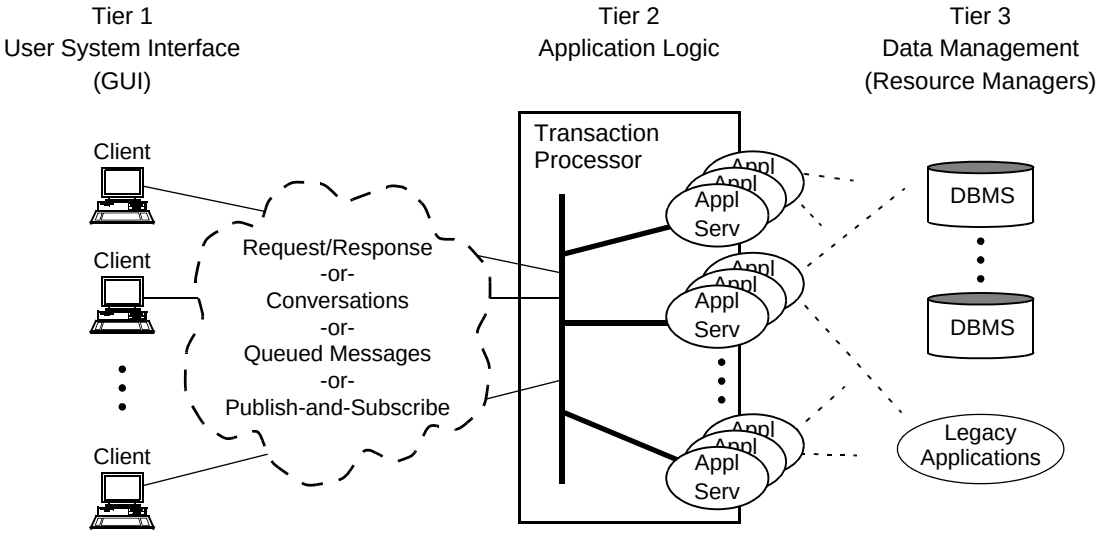
can use the same naming conventions (and namespace) to locate any resource in the Tuxedo domain.

BEA Tuxedo ATMI Overview

BEA Tuxedo ATMI is a set of core technologies that enables application designers to create ATMI applications that mix and match hardware platforms, databases, and operating systems. It provides all the features and benefits of a high-end online transaction processing (OLTP) system, including scalability, high-performance, mission-critical reliability, and open standards support.

At the foundation of BEA Tuxedo ATMI is a proven, reliable transaction processor, also known as a transaction processing (TP) monitor. As shown in the following figure, a transaction processor is an example of a 3-tier client/server architecture, where the transaction processor supports the application logic between the GUI front-end and the back-end resource managers. Examples of resource managers are SQL databases, message queues, legacy applications, and other back-end services.

Figure 2-1 3-Tier Client/Server Architecture Using a Transaction Processor



By breaking the direct connection between the user interface front-end and the resource managers, a transaction processor controls all the traffic that links hundreds or thousands or even tens of thousands of clients with application programs and the back-end resources. A transaction

processor ensures that global (distributed) transactions are completed accurately, provides load balancing, and improves the overall system performance. More importantly, a transaction processor makes an application's server processes independent of the user interface front-end and any resource manager.

BEA Tuxedo ATMI is a transaction application server that runs server-side applications and components. Besides managing an application's server processes and managing transactions, BEA Tuxedo ATMI also manages client/server communications, that is, allows clients (and servers) to invoke an application service in a variety of ways, including:

- Request/response

  Request/response transactions usually involve people and thus require immediate attention; they run in high-priority mode. BEA Tuxedo ATMI provides an ATMI request/response transactional communication interface.

- Conversations

  Conversational transactions also usually involve people and thus require immediate attention; they run in high-priority mode. BEA Tuxedo ATMI provides an ATMI conversational transactional communication interface.

- Queuing

  Queued transactions can run as high-priority or low priority messages. BEA Tuxedo ATMI includes its own bundled version of recoverable queues called */Q*.

- Publish-and-subscribe

  Publish-and-subscribe transactions usually run as high-priority messages. BEA Tuxedo ATMI has a transactional publish-and-subscribe system called *EventBroker*.

Transactional communications use highly augmented versions of remote procedure calls, conversational peer-to-peer, queues, and publish-and-subscribe. However, most of the value-added elements are transparent to the programmer: The transactional client/server exchanges look like ordinary exchanges bracketed by start and end transaction calls. The distinguishing factor is that all resource managers and processes invoked through these calls become part of the transaction. A transaction processor, such as BEA Tuxedo ATMI, orchestrates the actions of all the participants and makes them act as part of a transaction.

BEA Tuxedo ATMI Architecture

BEA Tuxedo ATMI consists of the following main components:

- BEA Tuxedo transaction processor and infrastructure

  Provides the core services needed to run and administer a distributed ATMI application.

- BEA Tuxedo Workstation

  Allows ATMI clients to reside on intelligent workstations and communicate over a network connection with an ATMI server application

- BEA Tuxedo /Q

  Provides a messaging and queuing capability to allow ATMI clients and servers to communicate across networks without being linked by a private, dedicated, logical connection.

- BEA Tuxedo EventBroker

  Provides a publish-and-subscribe capability that brokers the distribution of application and system events between ATMI clients and servers.

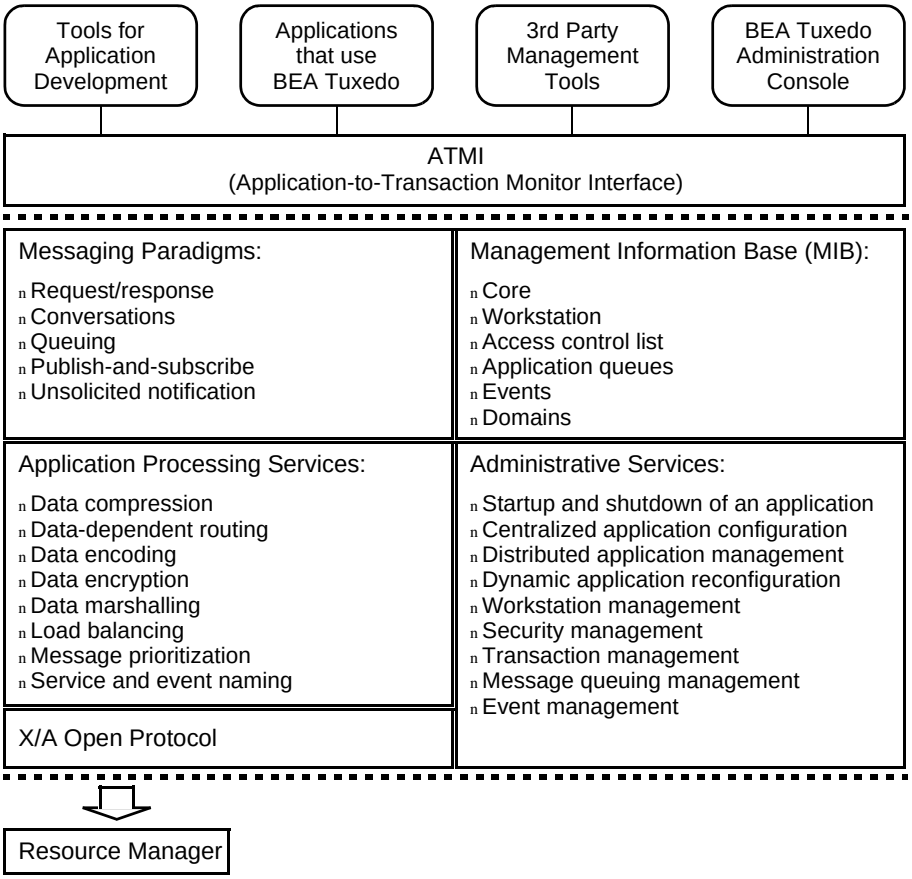
- BEA Tuxedo Domains

  Offers the ability to connect ATMI applications that are logically and physically separate so that the combination appears to the user as a single application.

BEA Tuxedo Transaction Processor and Infrastructure

The BEA Tuxedo infrastructure provides the bedrock client/server architecture for both BEA Tuxedo ATMI and BEA Tuxedo CORBA. The transaction processor and infrastructure discussed here and illustrated in the following figure constitute the BEA Tuxedo ATMI environment, which provides request/response and conversational communication interfaces, transaction support, and application-processing and administrative services for a distributed ATMI application.

Figure 2-2 BEA Tuxedo ATMI Environment



System Management Interface

The BEA Tuxedo system management interface, common to both BEA Tuxedo ATMI and BEA Tuxedo CORBA, can accommodate tools for administration, such as those described in "Management Tools" on page 1-14, and tools for application development, such as Simple Network Management Protocol (SNMP) agents. BEA Tuxedo provides an open tool environment that is supported by many third-party tools.

The BEA Tuxedo Administration Console and SNMP agents can interact with standard management consoles, which enables administrators to manage a BEA Tuxedo ATMI or

CORBA environment *and* a network configuration from one console. In addition, application architects and developers can build their own administrative tools or application-specific or market-specific tools on top of the Tuxedo management information base (TMIB) accessible through the MIB interface.

ATMI Programming Interface

BEA Tuxedo ATMI supports an ATMI programming interface that offers procedural library-based programming using a set of C or COBOL procedures. ATMI provides an interface for communications, transactions, and data-buffer management that works in all ATMI environments supported by the BEA Tuxedo system. The ATMI interface and the BEA Tuxedo system implement the X/Open distributed transaction processing (DTP) model for transaction processing.

The BEA Tuxedo ATMI interface provides a foundation for request/response and conversational communications.

Request/Response Communications

Programmers use the ATMI request/response functions to send a single request from a requesting process, and to receive a single response from the called request/response server process. Request/response is a simple type of dialogue. The rules for communication during request/response are fixed: the client asks for a service and the server responds. The client never sends more than one message as part of its request, and the server never sends more than one response in its reply.

For the requesting process, the execution of a request/response service can be synchronous or asynchronous.

Conversational Communications

Programmers use the ATMI conversational functions to establish and maintain state-preserving connections—context kept from message to message—between a requesting process and the called conversational server process. Specifically, programmers use the ATMI conversational functions to:

- Open a connection to a conversational server
- Begin and end a transaction during the conversation
- Have a conversation span multiple machines and resource managers

- Detect and provide notification of connection failures
- Terminate the connection when satisfied that the task has been completed

A conversational server is dedicated to the originating requester for the duration of the connection. The BEA Tuxedo system automatically spawns a new copy of a server if one is not available when a conversational connection is requested.

Thus, using the ATMI conversational programming interface, programmers can define transaction boundaries within their application so that the work performed can be treated as an *atomic unit*. What this statement means is that within a single BEA Tuxedo transaction, the work performed is either committed or rolled back *as a single unit of work*, which keeps all the databases synchronized, even if there are machine failures.

ATMI Interface Documentation

For more information on the BEA Tuxedo ATMI interface, see *Introducing BEA Tuxedo ATMI*.

FML Programming Interface

In addition to the ATMI interface, BEA Tuxedo ATMI supports a Field Manipulation Language (FML) programming interface, which is a set of C language functions for defining and manipulating storage structures called *fielded buffers*. Fielded buffers contain attribute-value pairs in fields, where the attribute is the field's identifier, and the associated value represents the field's data content.

If the FML and its fielded buffer concept are specified by the application designers, application programmers have a rich array of functions for the definition and management of FML fields and buffers. (See "Typed Buffers" on page 2-9 for a brief description of data buffers.) The selection includes functions to move data back and forth between a fielded buffer and a C structure or COBOL record (referred to as a VIEW), the members of which parallel the buffer's fields.

The FML function set has a companion function set, FML32, designed for use with larger records with more fields.

For more information on BEA Tuxedo FML, see *Programming a BEA Tuxedo Application Using FML*.

Typed Buffers

BEA Tuxedo ATMI applications send and receive their data in typed buffers. Instead of allocating memory directly from the operating system, applications allocate typed buffers from the BEA Tuxedo system in which to place their data.

Typed buffers are data structures defined by application programmers and made known to the BEA Tuxedo system. Because the BEA Tuxedo system knows about the application data buffers, it can optimally manipulate them during communication.

Typed buffers contain information about themselves (metadata), which allows application programmers to transfer data without needing to know which data representation scheme is used by the machines on which the application's clients and servers are running. Typed buffers allow applications to maintain machine independence.

Each buffer type supported by a BEA Tuxedo release has its own set of routines that can be called automatically to initialize; send and receive messages; and encode and decode data without programmer intervention. The set of routines is called a *typed buffer switch*.

BEA Tuxedo provides different kinds of typed buffers, including FML and FML32, and allows application designers to define their own typed buffers. For more information about typed buffers, see "What Are Typed Buffers?" in *Introducing BEA Tuxedo ATMI*.

BEA Tuxedo Workstation

The BEA Tuxedo Workstation component allows ATMI clients to reside on a remote machine that does not have a full BEA Tuxedo server-side installation, that is, a machine that does not support BEA Tuxedo administration servers or a bulletin board. All communication between a remote ATMI client and the BEA Tuxedo server application takes place over the network.

Advantages of the BEA Tuxedo Workstation component include:

- Less administrative overhead
- Greater security—keeps clients off the BEA Tuxedo server machines
- Off loads CPU cycles and decreases process context switches on BEA Tuxedo server machines
- Smaller footprint

Workstation Communication

The Workstation component involves the following software processes:

- Workstation client

  An ATMI client process that runs on a machine on which the BEA Tuxedo Workstation client software is installed.

- Workstation Listener (WSL)

  A BEA Tuxedo listening process, running on a BEA Tuxedo server machine, that accepts connection requests from Workstation clients and assigns connections to a Workstation Handler also running on the server machine. It also manages the pool of Workstation Handler processes, starting them in response to load demands.

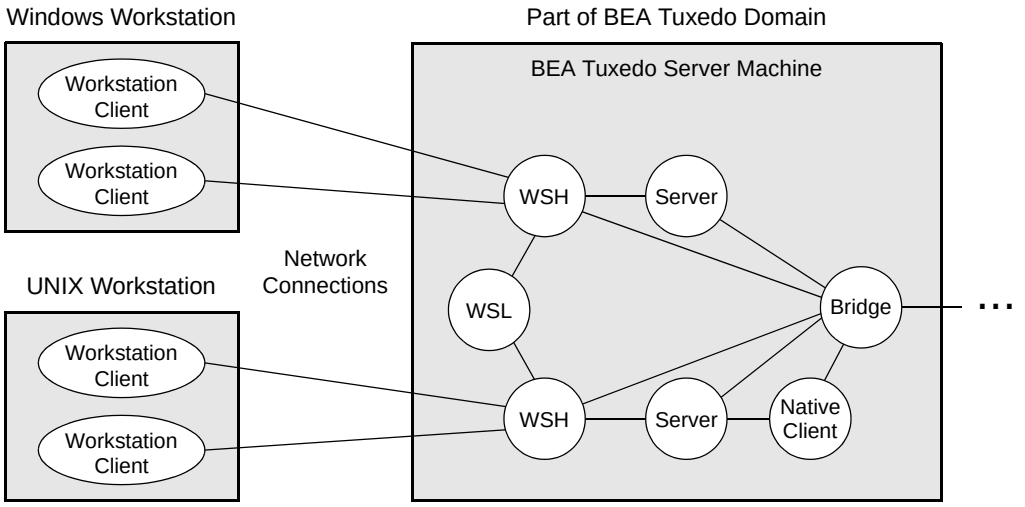
- Workstation Handler (WSH)

  A BEA Tuxedo gateway process, running on the BEA Tuxedo server machine, that handles communications between Workstation clients and the BEA Tuxedo server application. A WSH process resides within the administrative domain of the application and is registered in the local BEA Tuxedo bulletin board as a client.

  Each WSH process can manage multiple Workstation clients. A WSH multiplexes all requests and replies with a particular Workstation client over a single connection.

The following figure shows how these processes are used to connect remote ATMI clients to the BEA Tuxedo server application.

Figure 2-3 Connecting Remote ATMI Clients



Workstation Documentation

For more information about the BEA Tuxedo Workstation component, see the following documents:

- "BEA Tuxedo System Administration and Server Processes" in *Introducing BEA Tuxedo ATMI*
- *Using the BEA Tuxedo ATMI Workstation Component*
- "Administering Security" in *Using Security in ATMI Applications*
- `UBBCONFIG(5)`, `WS_MIB(5)`, and `WSL(5)` in *BEA Tuxedo File Formats, Data Descriptions, MIBs, and System Processes Reference*

BEA Tuxedo /Q

BEA Tuxedo /Q is a transactionally enabled, XA compliant, application queuing system incorporating typed buffers. /Q provides for time-independent communication among clients and servers in a BEA Tuxedo ATMI application.

/Q makes it possible for an ATMI application, within a global transaction, to store client and server generated messages to stable storage for processing later. A Q-enabled client or server

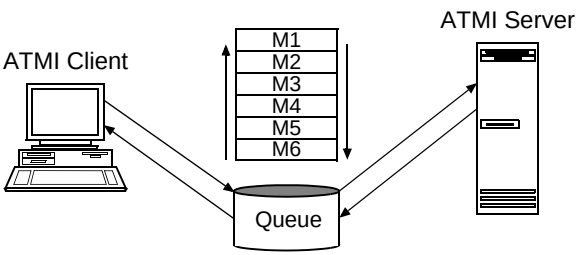
process decides when it wants to retrieve a message off its queue. However, because the operation is within the scope of a transaction, the BEA Tuxedo system ensures that either the message will eventually be processed or the entire transaction will be rolled back.

/Q can be combined with BEA Tuxedo Workstation to store and retrieve messages from Workstation clients. The interface for this combination is available in both the C and COBOL programming languages.

Storing and Retrieving Messages

Time-independent client and server programs communicate by storing (queuing) messages for each other in application queues. Messages can be retrieved (dequeued) in any of several ordering schemes, including *last in, first out* (LIFO), *first in, first out* (FIFO), priority order, and time-based order. More than one client and server can access the same queue. The following figure shows at a high level how message queuing communication works using /Q.

Figure 2-4 Queue-Based Messaging



/Q Documentation

For more information about the BEA Tuxedo /Q component, see the following documents:

- "BEA Tuxedo System Administration and Server Processes" in *Introducing BEA Tuxedo ATMI*
- *Using the ATMI /Q Component*
- `tpenqueue(3c)`and `tpdequeue(3c)` in *BEA Tuxedo ATMI C Function Reference*
- `APPQ_MIB(5)`, `TMQUEUE(5)`, `TMQFORWARD(5)`, and `UBBCONFIG(5)` in *BEA Tuxedo File Formats, Data Descriptions, MIBs, and System Processes Reference*

BEA Tuxedo EventBroker

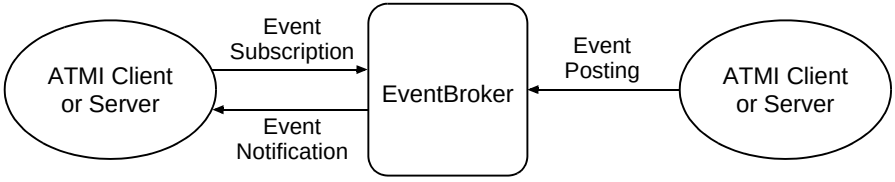
BEA Tuxedo EventBroker is a transactionally enabled, XA compliant, application publish-and-subscribe system that provides asynchronous routing of application events among the processes running in a BEA Tuxedo ATMI application. It also distributes system events to whichever application processes want to receive them.

An event is a state change or other occurrence in an application program or the BEA Tuxedo system that may be of interest to an administrator, an operator, or the software. Examples of events are “a stock traded at or above a specified price” or “a network failure occurred.”

Mediating Between Producers and Consumers of Events

There are producers of events, called *publishers* or *suppliers*, and consumers of events, called *subscribers*. EventBroker mediates between the producers and consumers about the distribution of events. The following figure shows how publish-and-subscribe communication works using EventBroker.

Figure 2-5 Event Subscription, Posting, and Notification



Posting an event in a global transaction means that all of the work, including work not related to the posting, is ensured to be complete if the transaction is successful. If any work performed within the transaction fails, all the work done within the transaction will be rolled back.

EventBroker Documentation

For more information about the BEA Tuxedo EventBroker component, see the following documents:

- “BEA Tuxedo System Administration and Server Processes” in *Introducing BEA Tuxedo ATMI*
- “About the EventBroker” in *Administering a BEA Tuxedo Application at Run Time*

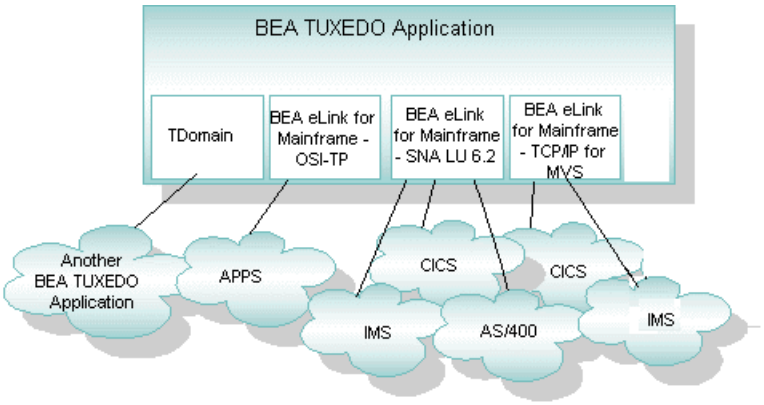
- `tppost(3c)`, `tpsubscribe(3c)`, and `tpunsubscribe(3c)` in *BEA Tuxedo ATMI C Function Reference*
- `EVENTS(5)`, `EVENT_MIB(5)`, `TMSYSEVT(5)`, `TMUSREVT(5)`, and `UBBCONFIG(5)` in *BEA Tuxedo File Formats, Data Descriptions, MIBs, and System Processes Reference*

BEA Tuxedo Domains

The BEA Tuxedo Domains component extends the BEA Tuxedo system client/server model to provide transaction interoperability across TP domains—business applications. This extension preserves the model and the ATMI interface by making access to services on the remote domain (or accepting service requests from a remote domain) transparent to both the application programmer and the end-user. The Domains component makes this possible via a highly asynchronous multitasking *domain gateway* that handles outgoing and incoming service requests to or from remote domains.

The BEA Tuxedo system offers the following types of domain gateways to allow a BEA Tuxedo application to communicate with other BEA Tuxedo applications or with applications running on other TP systems.

Figure 2-6 Domain Gateway Types



Note: BEA Tuxedo CORBA applications also use the Domains component to interoperate with one another and share resources. Only the *TDomain* gateway type—implemented by the `GWTDOMAIN` process—is applicable to BEA Tuxedo CORBA applications.

Transparency Between Domains

In a BEA Tuxedo Domains configuration, an administrator can configure which services of a domain are available to other domains in the configuration. The clients and the participating applications themselves do not need to know anything about the Domains configuration. All they need to know is what services or factory objects are available and how to access those services or objects. If applications were to include information about domains, changing configurations would require that the applications be rewritten as well.

Domains Documentation

For more information about the BEA Tuxedo Domains component, see the following documents:

- "BEA Tuxedo System Administration and Server Processes" in *Introducing BEA Tuxedo ATMI*
- *Using the BEA Tuxedo Domains Component*
- `DMADM(5)`, `DMCONFIG(5)`, `GWADM(5)`, `GWTDOMAIN(5)`, and `UBBCONFIG(5)` in *BEA Tuxedo File Formats, Data Descriptions, MIBs, and System Processes Reference*

CHAPTER 3

BEA Tuxedo CORBA Components

The following sections describe the BEA Tuxedo CORBA components built on the BEA Tuxedo infrastructure:

- BEA Tuxedo CORBA Overview
- BEA Tuxedo CORBA TP Framework
- BEA Tuxedo CORBA Architecture
- BEA Tuxedo OTM and Infrastructure
- BEA Tuxedo ORB Software
- BEA Tuxedo IIOP Listener/Handler
- BEA Tuxedo CORBA Environmental Objects
- BEA Tuxedo CORBA Object Services
- BEA Tuxedo TP Framework

Note: The BEA Tuxedo CORBA Java client and BEA Tuxedo CORBA Java client ORB were deprecated in Tuxedo 8.1 and are no longer supported. All BEA Tuxedo CORBA Java client and BEA Tuxedo CORBA Java client ORB text references, associated code samples, should only be used to help implement/run third party Java ORB libraries, and for programmer reference only.

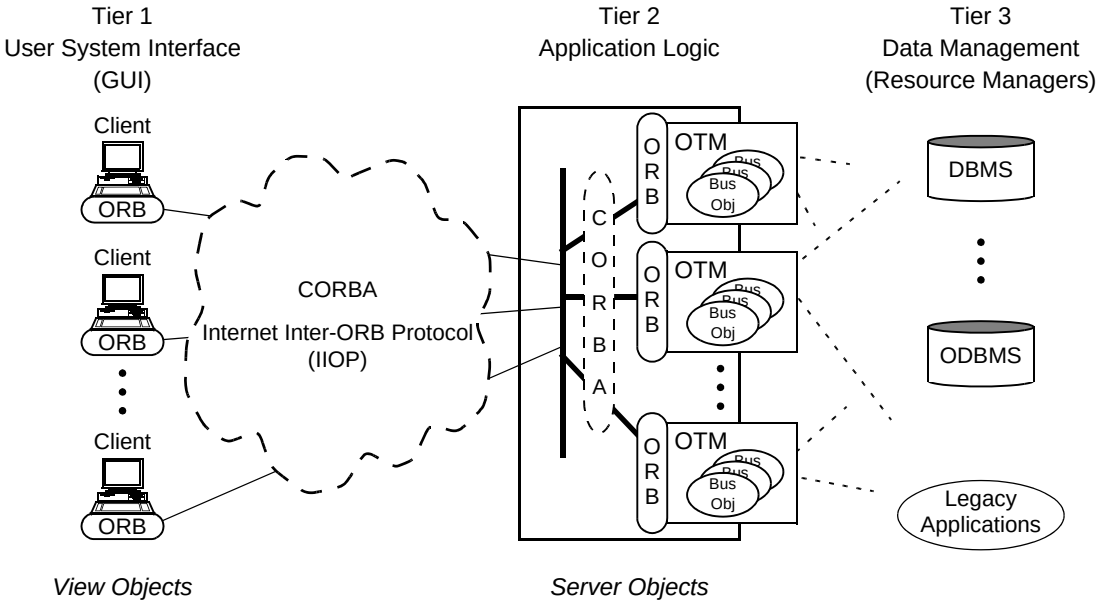
Technical support for third party CORBA Java ORBs should be provided by their respective vendors. BEA Tuxedo does not provide any technical support or documentation for third party CORBA Java ORBs.

BEA Tuxedo CORBA Overview

BEA Tuxedo CORBA provides businesses and organizations that depend on mission-critical applications with the advantages of the CORBA-compliant programming model, combined with the power, robustness, and proven reliability of the Tuxedo transaction processing technology. BEA Tuxedo CORBA also taps into the existing Tuxedo infrastructure for transaction management, security, message transport, administration and manageability, and XA-compliant database support.

BEA Tuxedo CORBA combines the ORB model with online transaction processing (OLTP) functions to create a top-of-the-line Object Transaction Monitor (OTM). As shown in the following figure, an OTM is an example of a 3-tier client/server architecture, where the OTM supports the application logic between the GUI front-end and the back-end resource managers. Examples of resource managers are object-oriented databases, relational databases, message queues, legacy applications, and other back-end services.

Figure 3-1 3-Tier Client/Server Architecture Using an OTM



By breaking the direct connection between the user interface front-end and the resource managers, an OTM controls all the traffic that links hundreds or thousands or even tens of

thousands of clients with run-time objects and the back-end resources. An OTM ensures that global (distributed) transactions are completed accurately, provides load balancing, and improves the overall system performance. An OTM also prestarts pools of objects and provides fault-tolerance. More importantly, an OTM makes an application's server processes independent of the user interface front-end and any resource manager.

BEA Tuxedo CORBA is an object application server that runs server-side distributed objects. Besides managing an application's server objects and managing transactions, BEA Tuxedo CORBA also manages client/server communications.

Object-oriented transactional communications use a highly augmented version of ORB invocations. However, most of the value-added elements are transparent to the programmer: The transactional client/server exchanges look like ordinary exchanges bracketed by start and end transaction calls. The distinguishing factor is that all resource managers and processes invoked through these calls become part of the transaction. An OTM, such as BEA Tuxedo CORBA, orchestrates the actions of all the participants and makes them act as part of a transaction.

BEA Tuxedo CORBA TP Framework

The BEA Tuxedo CORBA OTM provides a TP framework, or organized environment, for running server-side distributed objects. A TP framework not only calls the objects and the Tuxedo CORBA services at the appropriate time and in the correct sequence, but it also simplifies the server-side programming model.

BEA Tuxedo CORBA Architecture

BEA Tuxedo CORBA consists of the following main components:

- BEA Tuxedo OTM and infrastructure

  Provides the services needed to run and administer a distributed CORBA application.

- BEA Tuxedo ORBs

  Allows Tuxedo CORBA client and server objects to locate and communicate with one another.

- BEA Tuxedo IIOP Listener/Handler

  Allows Tuxedo CORBA clients to reside on intelligent workstations and communicate over a network connection with a CORBA server application.

- BEA Tuxedo environmental objects

Provides a set of objects for helping Tuxedo CORBA clients and servers work with the Tuxedo CORBA environment.

- BEA Tuxedo CORBA object services

  Provides object services to Tuxedo CORBA clients.

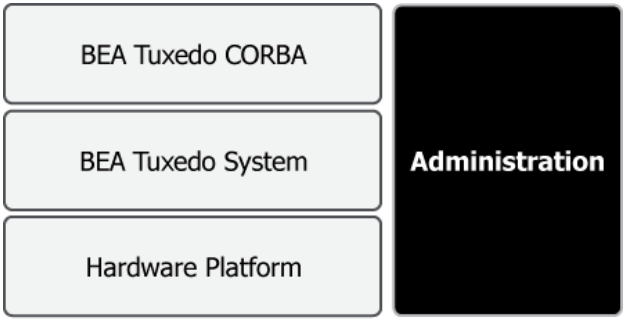
- BEA Tuxedo TP Framework

  Provides a programming model for the rapid construction of Tuxedo CORBA server applications.

BEA Tuxedo OTM and Infrastructure

The BEA Tuxedo infrastructure provides the bedrock client/server architecture for both BEA Tuxedo CORBA and BEA Tuxedo ATMI. The OTM and infrastructure discussed here and illustrated in the following figure constitute the BEA Tuxedo CORBA environment, which provides the communication interfaces, transaction support, and application-processing and administrative services for a distributed CORBA application.

Figure 3-2 BEA Tuxedo CORBA Environment



System Management Interface

The BEA Tuxedo system management interface, common to both BEA Tuxedo CORBA and BEA Tuxedo ATMI, can accommodate tools for both application development and administration. For information about the BEA Tuxedo system management interface, see "System Management Interface" on page 2-6.

Application Programming Interface

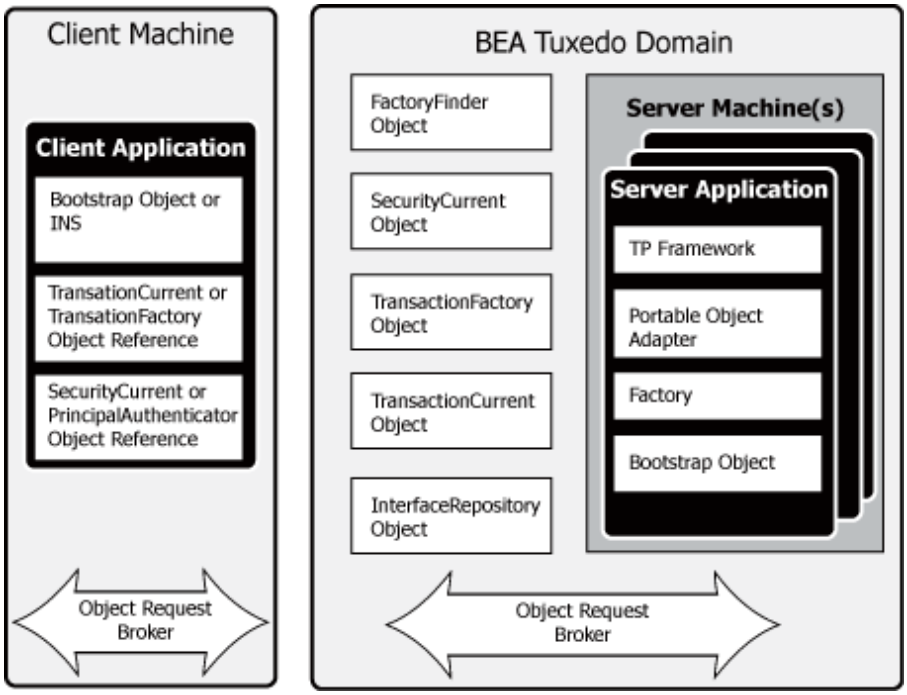
The BEA Tuxedo CORBA programming interface consists of a C++ server ORB and a C++ client ORB. On the server side, instead of using the CORBA API directly, application programmers use an API that automates many of the functions required in a standard CORBA application.

The BEA Tuxedo CORBA server-side *TP Framework* component and client-side *environmental objects* enable programmers to use the deployment environment with a minimal amount of programming. For information about the TP Framework component, see "BEA Tuxedo TP Framework" on page 3-11. For information about client-side environmental objects, see "BEA Tuxedo CORBA Environmental Objects" on page 3-9.

Application Programming Environment

Application programmers develop a Tuxedo CORBA application as a set of CORBA objects, using the OMG Interface Definition Language (IDL) and, optionally, using standard, off-the-shelf programming tools. These objects communicate with other objects using the CORBA Internet Inter-ORB Protocol (IIOP). The following figure identifies the architectural components of the BEA Tuxedo CORBA programming environment.

Figure 3-3 Components in a BEA Tuxedo CORBA Application



BEA Tuxedo CORBA runs the objects in the server processes that it manages. BEA Tuxedo CORBA can also manage server processes that run Tuxedo ATMI services, thereby allowing programmers to combine object-based and service-based components in the same Tuxedo application.

Note: A Tuxedo *application* has the same meaning as a Tuxedo *domain*. For a definition of a Tuxedo domain, see "Important BEA Tuxedo Terms and Concepts" on page 2-1.

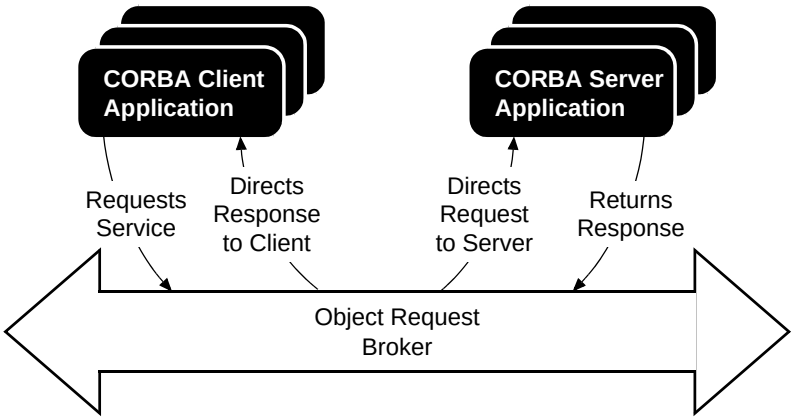
BEA Tuxedo ORB Software

The ORB, or Object Request Broker, is a library that enables clients to locate and communicate with servers independently of server location and network connections. The ORB is sometimes referred to as the *object bus*.

Programmers define their object's interface via OMG IDL, and the ORB takes care of the rest. The ORB serves as an intermediary for requests that CORBA clients send to CORBA server

applications, so that the client and server do not need to contain information about each other. The following figure shows the relationship between an ORB, a CORBA application client, and a CORBA server application.

Figure 3-4 The ORB in a CORBA Client/Server Environment



BEA Tuxedo CORBA includes C++ server ORB and C++ client ORB. The ORBs have built-in transaction support, meaning that the CORBA Object Transaction Service (OTS), on which a CORBA OTM is based, is patterned after the XA standard for two-phase commit processing.

The C++ server ORB is linked directly into the Tuxedo CORBA server processes. Other client ORBs can communicate with BEA Tuxedo CORBA via CORBA's IIOP protocol.

BEA Tuxedo IIOP Listener/Handler

The BEA Tuxedo IIOP Listener/Handler allows a CORBA client residing on a remote machine that does not have a full BEA Tuxedo server-side installation (that is, a machine that does not support BEA Tuxedo administration servers or a bulletin board) to be able to communicate with a BEA Tuxedo CORBA server application. All communication between a remote CORBA client and the CORBA server application takes place over a network connection using the IIOP protocol.

Advantages of remote CORBA clients include:

- Less administrative overhead

- Greater security—keeps clients off the BEA Tuxedo server machines
- Off loads CPU cycles and decreases process context switches on BEA Tuxedo server machines
- Smaller footprint

IIOP Listener/Handler Communication

The IIOP Listener/Handler communication architecture involves the following software processes:

- CORBA client

  A client process that runs on a machine on which the BEA Tuxedo CORBA C++ client ORB software is installed.

- IIOP Listener (ISL)

  A BEA Tuxedo listening process, running on a BEA Tuxedo server machine, that accepts connection requests from CORBA clients and assigns connections to an IIOP Handler also running on the server machine. It balances client connections across handlers. In addition, an IIOP Listener manages the pool of IIOP Handler processes, starting them in response to load demands.

- IIOP Handler (ISH)

  A BEA Tuxedo gateway process, running on the BEA Tuxedo server machine, that handles IIOP communications between CORBA clients and the BEA Tuxedo server application. An ISH process resides within the administrative domain of the application and is registered in the local BEA Tuxedo bulletin board as a client.

  Each ISH process can manage multiple CORBA clients. An ISH multiplexes all requests and replies with a particular CORBA client over a single connection.

IIOP Listener/Handler Documentation

For more information about the IIOP Listener/Handler, see the following documents:

- *Setting Up a BEA Tuxedo Application*
- `ISL(1)` in *BEA Tuxedo Command Reference*

BEA Tuxedo CORBA Environmental Objects

BEA Tuxedo CORBA provides a set of objects for helping the client work with the Tuxedo CORBA environment; the objects enable client applications to easily log on to a Tuxedo CORBA environment, invoke CORBA objects, and start and end transactions. Like the server-side TP Framework component, these objects interact with the Tuxedo CORBA services.

Here is what these objects do for application clients:

- Bootstrap object

  The Bootstrap object provides references to the Tuxedo CORBA objects in a Tuxedo CORBA application. An application client can connect to multiple BEA Tuxedo CORBA applications using different Bootstrap objects.

  One of the first things that an application client does after startup is to create a Bootstrap object by supplying the host and port number of the IIOP Listener. After the application client contacts an IIOP Listener, the Listener assigns an IIOP Handler to the application client, and the Boostrap object establishes a communication link with the assigned IIOP Handler.

  The Bootstrap object also provides references to well-known objects that application clients use, such as TransactionCurrent, SecurityCurrent, InterfaceRepository, and FactoryFinder.

- CORBA OTS TransactionCurrent object

  The CORBA OTS TransactionCurrent object coordinates transaction demarcations with the transaction coordinator.

- SecurityCurrent object

  The SecurityCurrent object gets the application client's security credentials from the Security Service. The SecurityCurrent object registers the certificate with the IIOP Handler, which uses the certificate to permit or deny invocations.

BEA Tuxedo CORBA Object Services

BEA Tuxedo CORBA provides environmental objects for the C++ programming environment and for the Java programming environment. As of release 8.0, BEA Tuxedo CORBA also supports the use of the OMG CORBA Interoperable Naming Service (INS) by third-party client ORBs, to obtain initial object references.

Each environmental object provides object services to application clients. Application clients access the environmental objects through a bootstrapping process that accesses the services in a particular BEA Tuxedo server application. BEA client ORBs use the BEA Bootstrap object mechanism, and third-party client ORBs use the CORBA INS mechanism. For more information about bootstrapping a BEA Tuxedo application, see *BEA Tuxedo CORBA Programming Reference*.

The BEA Tuxedo CORBA environmental objects provide the following services:

- Object Life Cycle service

  The Object Life Cycle service is provided through the FactoryFinder environmental object. The FactoryFinder object is a CORBA object that can be used to locate a factory, which in turn can create object references for CORBA objects. Factories and FactoryFinder objects are implementations of the CORBA Services Life Cycle Service. BEA Tuxedo CORBA applications use the Object Life Cycle service to find object references.

- Security service

  The Security service is accessed through either the SecurityCurrent environmental object or the PrincipalAuthenticator object. The SecurityCurrent and PrincipalAuthenticator objects are used to authenticate an application client attempting to access a BEA Tuxedo server application. The BEA Tuxedo software provides an implementation of the CORBA Services Security Service.

- Transaction service

  The Transaction service is accessed through either the TransactionCurrent environmental object or the TransactionFactory object. The TransactionCurrent and TransactionFactory objects allow an application client to demarcate a transaction—that is, begin, suspend, resume, and commit a transaction. The BEA Tuxedo software provides an implementation of the CORBA Services Object Transaction Service (OTS).

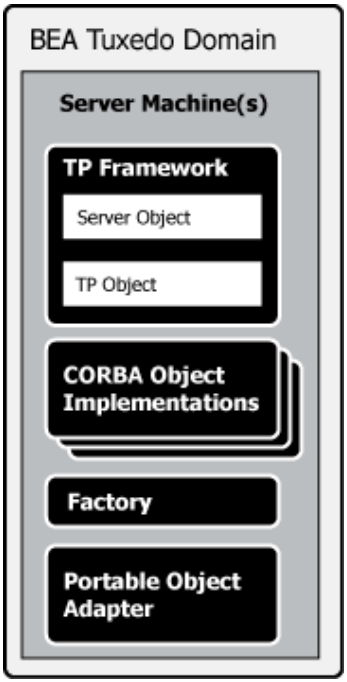
- Interface Repository service

  The Interface Repository service is accessed through the InterfaceRepository object. The InterfaceRepository object is a CORBA object that contains interface definitions for all the available CORBA interfaces and the factories used to create object references to the CORBA interfaces. The InterfaceRepository object is used with application clients that use Dynamic Invocation Interface (DII).

BEA Tuxedo TP Framework

The TP Framework component, shown in the following figure, provides a programming model that achieves high levels of performance while shielding the application programmer from the complexities of the CORBA interfaces.

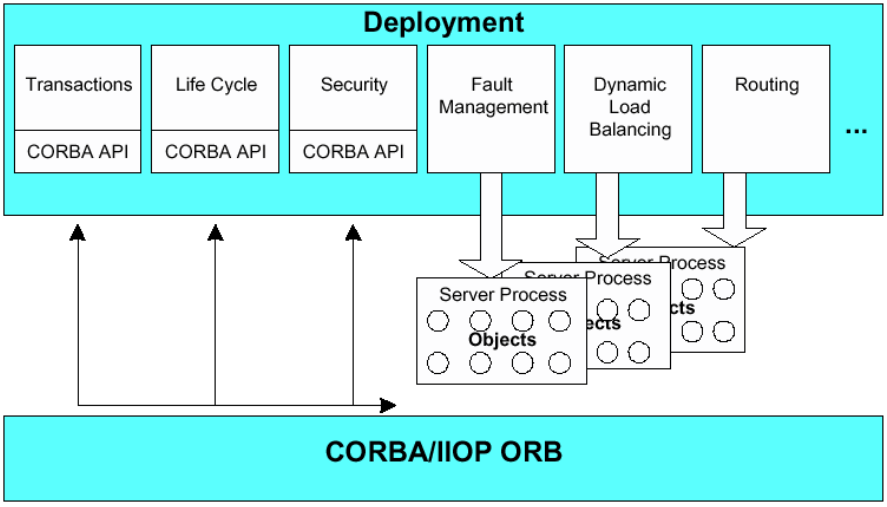
Figure 3-5 The TP Framework



The TP Framework API provides routines that perform many of the functions required in a standard CORBA application. Application programmers are responsible only for writing the business logic of the CORBA application and overriding default actions provided by the TP Framework.

The TP Framework and environmental objects help make development easy. The following figure illustrates the BEA Tuxedo CORBA development environment.

Figure 3-6 BEA Tuxedo CORBA Deployment Environment



CHAPTER 4

BEA Tuxedo Web-Accessible Services

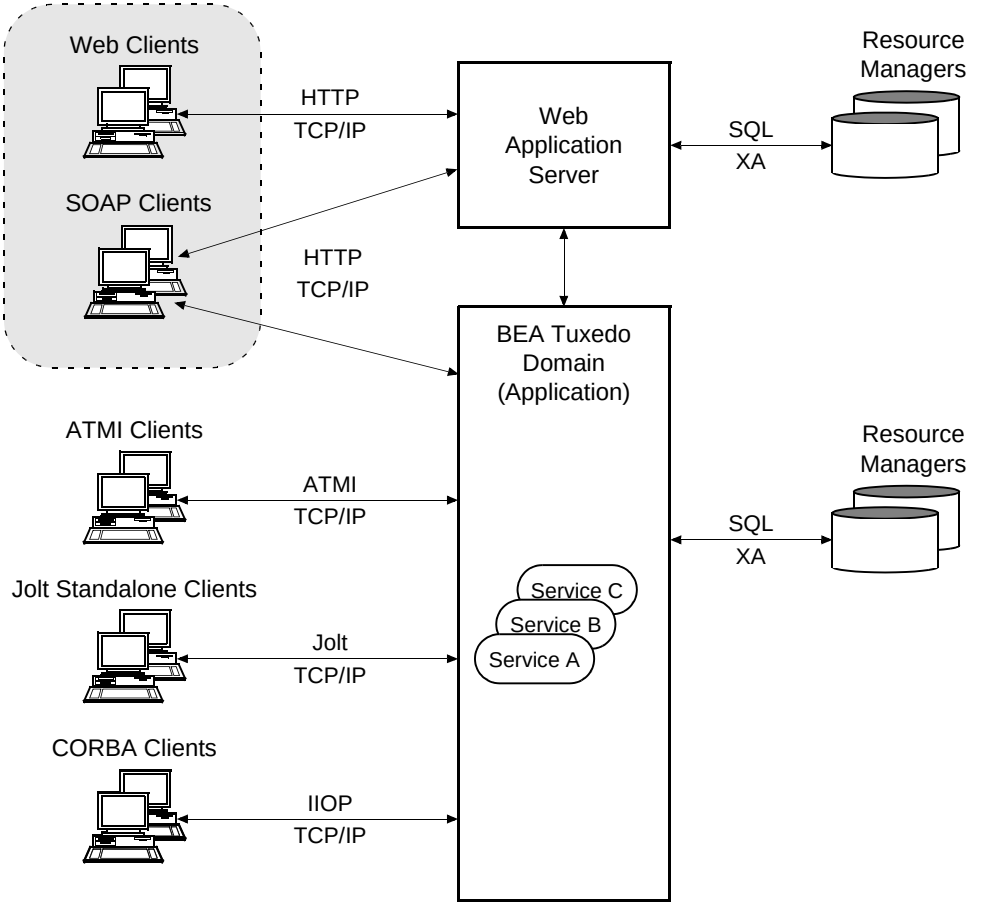
The following sections present the many ways of making BEA Tuxedo services available to Web based applications:

- What Does Web Accessible Mean?
- Exposing Tuxedo Services as Web Services
- Acessing Tuxedo Services Using a Web Application Server

What Does Web Accessible Mean?

"Web accessible," as used in the discussions that follow, means making BEA Tuxedo application services available to Web based applications. The following figure helps clarify the meaning of "Web accessible."

Figure 4-1 Web Client Access to BEA Tuxedo Application Services



A Web based client application may be a simple Web client, e.g. a Web browser displaying Hypertext Markup Language (HTML) pages to the end users or a Web client with SOAP engine that can initiating Web Service standard requests.

A Web application server may be a Web server or a cross between a Web server and an application server. The standard definition of a *Web server* is "a server software system that serves static content to a Web browser by loading a file from a disk and serving it across the network to a user's Web browser. This entire exchange is mediated by the browser and server

talking to each other using Hypertext Transfer Protocol (HTTP).” The standard definition of an *application server* is “a server software system that occupies a large chunk of computing territory between database servers and the end user, and often connects the two. An application server is sometimes referred to as a type of middleware.” The BEA Tuxedo system, itself, is essentially two high-performance application servers, a transaction processing application server and an object application server.

A Web application server serves Web clients one of two types of pages (documents): Hypertext Markup Language (HTML) pages or Extensible Markup Language (XML) pages.

Exposing Tuxedo Services as Web Services

Exposing Tuxedo services as Web services opens the application to the outside world *without any application code changes*. The application can be broken down into smaller modular components, or shared services, that can be shared by and used as components of distributed Web-based applications.

BEA provides both Tuxedo native solution and Tuxedo-Other BEA products integrated solution for exposure of Tuxedo Services as Web Services.

Web Services Standards at a Glance

The Web services technologies and programmatic interfaces are being developed by the World Wide Web Consortium (W3C). Web services are based on HTTP and XML as well as the following relatively new XML-based Internet technologies:

- Web Services Description Language (WSDL)

  The XML-based language for describing (1) the methods provided by a Web service, (2) the input and output parameters of the Web service, and (3) the instructions for connecting to the Web service. WSDL is the standardized way to describe a Web service to clients so that they can invoke it.

- Simple Object Access Protocol (SOAP)

  An XML/HTTP-based protocol for accessing services, objects, and servers in a platform-independent manner. SOAP is the standardized way to transmit data and Web service invocation calls between users and providers of a Web service.

- Universal Description, Discovery, and Integration (UDDI)

A repository that stores descriptions, in a common XML format, about companies and the services they offer. UDDI is the standardized way for client applications to find a registered Web service and to register a Web service on an Internet server.

Web services communicate with clients, both end-user applications and other Web services, through XML messages transmitted by HTTP. Web services can reside on different computers and can be implemented by vastly different technologies, but they are packaged and transported using standard Internet protocols, thus making them easily accessible by any user on the Internet.

For information about the Web services technologies, see *W3C - Web Services Activity* at `http://www.w3.org/2002/ws`.

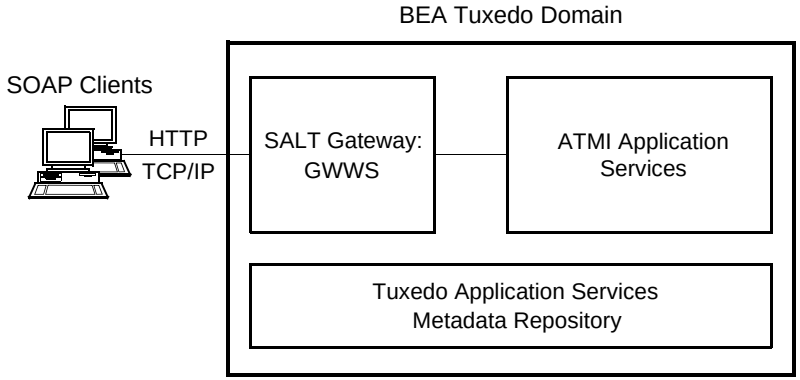
Exposing Tuxedo Services as Web Services Through BEA SALT

BEA SALT is one of the latest add-ons to the Tuxedo product family. Created in 2006, BEA SALT is designed to provide a seamless Tuxedo solution of integrating Tuxedo applications and standard Web services application. BEA SALT allows external Web service applications to invoke native Tuxedo services (inbound), and conversely, allows Tuxedo applications to invoke external Web services (outbound). BEA SALT is a native Tuxedo Web service integration solution.

BEA SALT complies with most primary Web Services standards: SOAP 1.1, SOAP 1.2, and WSDL 1.1. With BEA SALT, Tuxedo applications can be easily exposed as Web Services.

The following figure shows the principal software components comprising BEA Tuxedo native Web Services solution to expose Tuxedo application services as Web services.

Figure 4-2 Exposing Tuxedo Application Services as Web Services Through BEA SALT



BEA SALT is the preferred product for exposing Tuxedo ATMI services as Web services. It reduces Tuxedo/Web Service integration costs and decreases conversion processes that may exist with other solutions for accessing Tuxedo services. It enables seamless connectivity between Tuxedo applications and external Web service applications.

SALT Gateway Server — GWWS

BEA SALT provided Tuxedo system server (`GWWS`), connects with other Web service applications via SOAP over HTTP/S protocol. The `GWWS` server acts as a Tuxedo gateway process and is managed in the same manner as general Tuxedo system servers. Each `GWWS` server has bi-directional (inbound/outbound) capability. The `GWWS` server:

- accepts SOAP requests from Web service applications and issue Tuxedo native calls to Tuxedo services.
- accepts Tuxedo ATMI requests and issues SOAP calls to Web Service applications.

SALT Documentation

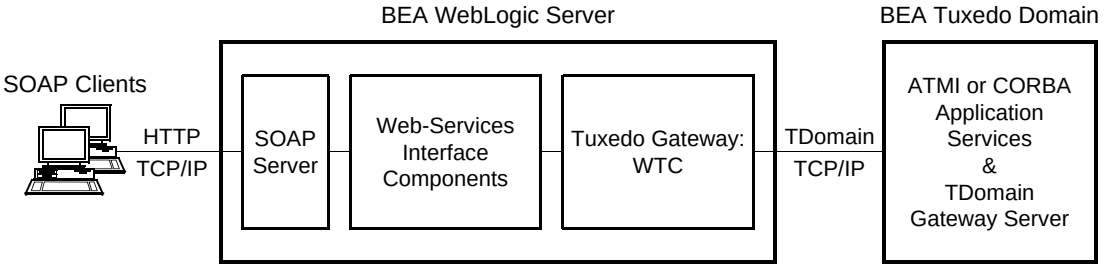
For information about BEA SALT, see http://edocs.bea.com/salt/docs20.

Exposing Tuxedo Services as Web Services Through Other BEA Products

Through BEA WebLogic Server

The following figure shows the principal software components comprising BEA Tuxedo-WebLogic integrated solution to expose Tuxedo application services as Web services.

Figure 4-3 Exposing Tuxedo Application Services as Web Services Through BEA WebLogic Server



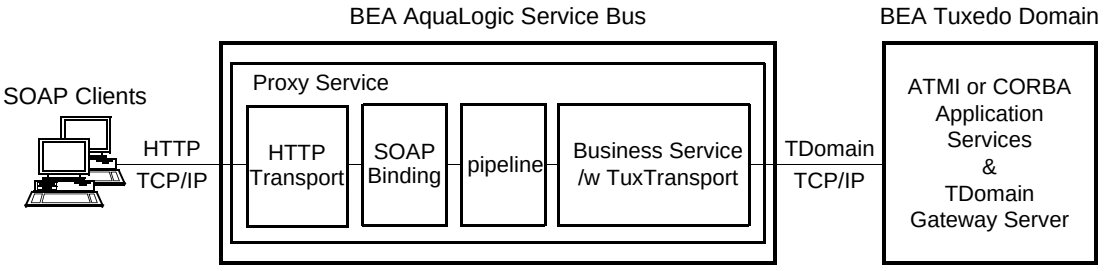
Both Java and non-Java client applications (such as Microsoft .Net Framework clients) can invoke Tuxedo services exposed as Web services through WebLogic Server. The client application assembles a SOAP message describing the Web service it wants to invoke and includes all the necessary data, either in the SOAP body or in a SOAP attachment. The client then sends the SOAP message over HTTP to WebLogic Server, which executes the Web service by performing the following tasks.

1. Calls the associated Tuxedo service via the WTC gateway.
2. Packages the Tuxedo response in a SOAP message.
3. Sends the SOAP message back to the client over HTTP.

Through BEA AquaLogic Service Bus

The following figure shows the principal software components comprising BEA Tuxedo-AquaLogic Service Bus integrated solution to expose Tuxedo application services as Web services.

Figure 4-4 Exposing Tuxedo Application Services as Web Services Through BEA AquaLogic Service Bus



AquaLogic Service Bus is an Enterprise-class service bus that connects, manages, and mediates interactions between heterogeneous services. To connect between SOAP client applications and Tuxedo ATMI services, you should deploy both SOAP connectivity components and Tuxedo domain connectivity components based on AquaLogic architecture.

BEA Web Services Documentation

For information about BEA Web services and BEA products, see:

- *BEA WebLogic Web Services* at `http://edocs.bea.com/wls/docs100/webservices.html`
- *BEA AquaLogic Service Bus* at `http://edocs.bea.com/alsb/docs26/index.html`
- *BEA Dev2Dev Web Services* at `http://dev2dev.bea.com/webservices/`

Acessing Tuxedo Services Using a Web Application Server

Besides being made available as Web Services, BEA Tuxedo application services are also made available to Web client programs through a Web application server. Applications embedded in the Web Application Servers can access Tuxedo ATMI services through one of the following approaches:

- Making Tuxedo Services Web Accessible Through BEA Jolt
- Making Tuxedo Services Web Accessible Through BEA WebLogic Server

Making Tuxedo Services Web Accessible Through BEA Jolt

BEA Jolt provides Internet access to Tuxedo ATMI services for both Web-browser and standalone Java clients. Using Jolt, Java programmers can build client applets and applications that remotely invoke existing and new Tuxedo applications, allowing secure, scalable, intranet/Internet transactions between client and server.

Using Jolt, Java programmers can also use HTTP servlets to perform server-side Java tasks in response to HTTP requests. This type of Jolt connectivity enables simple Web clients to access Tuxedo application services through any Web application server that supports generic servlets.

Jolt Class Library

The Jolt class library provides programmers with a set of object-oriented Java language classes for accessing BEA Tuxedo ATMI services. The class library contains the class files that implement the Jolt API.

Jolt Client Personalities

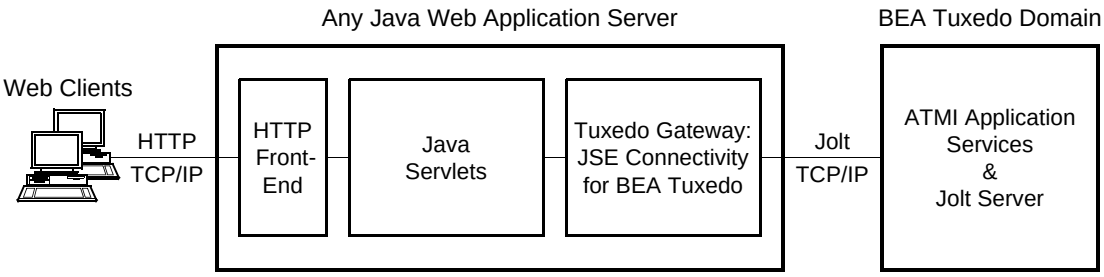
In addition to Jolt applets and Jolt standalone applications, BEA Jolt supports the following three types of Jolt client personalities for simple Web clients:

- JSE Connectivity for BEA Tuxedo
- WebLogic Connectivity for BEA Tuxedo

JSE Connectivity for BEA Tuxedo

This Jolt client personality is a Jolt HTTP servlet, running in a Java Web application server environment (for example, BEA WebLogic Server), through which simple Web-browser clients can invoke Tuxedo ATMI services. Accessing Tuxedo ATMI services in this manner requires the installation of Jolt class packages `jolt.jar` and `joltjse.jar` on the machine running the Web application server.

Figure 4-5 Web Access to Tuxedo Using Jolt JSE Connectivity

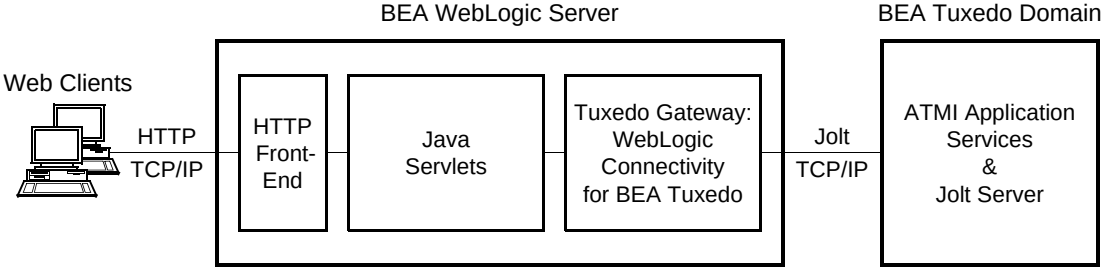


A Jolt HTTP servlet uses Jolt session pool classes to invoke Tuxedo services on behalf of simple browser clients. Thus, the servlet handles all Jolt transactions on the Web server, which enables simple browser clients to invoke BEA Tuxedo services without directly connecting to the Jolt server and BEA Tuxedo.

WebLogic Connectivity for BEA Tuxedo

This Jolt client personality is a customized version of Jolt JSE Connectivity for the BEA WebLogic Server. Accessing Tuxedo ATMI services in this manner requires the installation of Jolt class packages `jolt.jar`, `joltjse.jar`, and `joltwls.jar` on the machine running BEA WebLogic Server.

Figure 4-6 Web Access to Tuxedo Using Jolt WebLogic Connectivity



Note: The Jolt client personality "WebLogic Connectivity for BEA Tuxedo" is also known as "BEA Jolt for BEA WebLogic Server."

Jolt Servers

The Jolt server implementation acts as a proxy for the Jolt client, invoking the BEA Tuxedo service on behalf of the client. The Jolt server accepts requests from Jolt clients and maps those requests into BEA Tuxedo service requests.

Jolt Documentation

For information on configuring the Jolt server and the BEA Tuxedo server to work with Jolt, see "BEA Jolt 10.0 Overview and Installation Information" in *Installing the BEA Tuxedo System*.

For common client and Web server deployment considerations, see *Using BEA Jolt* and *Using BEA Jolt with BEA WebLogic Server*.

Making Tuxedo Services Web Accessible Through BEA WebLogic Server

BEA Tuxedo services have been Web accessible through BEA WebLogic Server ever since BEA WebLogic Server release 5.1. The following BEA Jolt software and BEA WebLogic Server gateways are central to this accessibility:

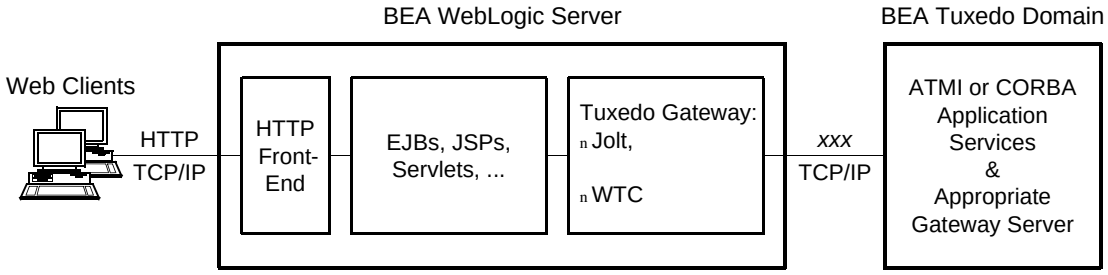
- BEA Jolt for BEA WebLogic Server—also known as Jolt client personality "WebLogic Connectivity for BEA Tuxedo"

  Enables WebLogic Server 5.1 or later EJBs, JavaServer Pages (JSPs), servlets, and other WebLogic Server application servers to call Tuxedo ATMI services on behalf of WebLogic Server Web-browser clients.

- WebLogic Tuxedo Connector (WTC) gateway

  Enables WebLogic Server 6.1 or later applications, such as servelets and other WebLogic Server applications, to call Tuxedo ATMI services or Tuxedo CORBA C++ objects on behalf of WebLogic Server Web-browser clients.

Figure 4-7 Web Access to Tuxedo Using Jolt or WTC



In addition to WTC, there is support for IIOP connections from WebLogic Server (WLS) via the WLS ORB and the Tuxedo ISL.

For details about using Jolt or WTC to achieve interoperability between BEA Tuxedo and BEA WebLogic Server, see "Interoperability with BEA WebLogic Server" in *BEA Tuxedo Interoperability*.

CHAPTER 5

BEA Tuxedo Product Support and Resources

The following sections describe the documentation and customer-support resources available to BEA Tuxedo customers:

- About the BEA Tuxedo Documentation
- Using the BEA Tuxedo Online Documentation
- BEA dev2dev Online
- BEA Consulting Services
- BEA Education Services

About the BEA Tuxedo Documentation

The BEA Tuxedo documentation is designed to provide you, the customer, with information at various levels about the BEA Tuxedo system. You may want to read all of the documentation or choose only those topics that will provide information for your immediate requirements.

The BEA Tuxedo documentation consists of the following components:

- Online documentation
- Context-sensitive online help for BEA Tuxedo GUI-based applications

BEA Tuxedo Online Documentation

The online documentation is on the "e-docs" Product Documentation page at http://e-docs.bea.com.

For a listing of the individual documents that are included in the online documentation, access the e-docs Product Documentation page, click "BEA Tuxedo 10.0" to display the BEA Tuxedo Online Documentation Home page, and then click "Site Map."

BEA Tuxedo Context-Sensitive Help

The BEA Tuxedo software includes a set of GUI-based tools designed to help you build and administer your BEA Tuxedo client and server applications. The following table lists the context-sensitive help components provided with each BEA Tuxedo software GUI.

Table 5-1 BEA Tuxedo Context-Sensitive Online Help

Help Component	Description
BEA Tuxedo Administration Console Online Help	Provides help topics that describe how to use the BEA Tuxedo Administration Console to remotely administer the BEA Tuxedo system from a Web browser.

Using the BEA Tuxedo Online Documentation

The BEA Tuxedo online documentation contains a comprehensive set of documents about the BEA Tuxedo system. This information is designed to help you:

- Understand the key functionality of the BEA Tuxedo system
- Design, develop, and deploy mission-critical client/server applications
- Manage your BEA Tuxedo application resources using the software administration tools provided with the BEA Tuxedo system

The online documentation provides easy-to-access information in HTML format for viewing in your favorite Web browser.

Note: Microsoft Internet Explorer 6.0 or later is recommended.

To view the online documentation, you need a Web browser that supports HTML 3.0 features, including tables and frames.

Accessing the Documentation in a Browser

To begin viewing the Online Documentation Home page, access BEA Tuxedo on the e-docs Product Documentation page. From this page you can:

- Browse through each online document
- See what's new in this release
- View a site map of all of the documents and click a graphic to display a document
- Search the entire set of HTML-based documentation using a word or a phrase
- Get a printed copy of each of the major documents by opening and printing an Adobe Acrobat PDF file

Each major topic area is displayed with its own table of contents so that you can see at-a-glance what each document contains.

Once you access the online documentation, you can quickly browse through all of the available information.

Select a topic in the table of contents or click "BEA Tuxedo Documentation" to return to the Home page.

The online documentation offers many options to access the documentation for the BEA Tuxedo system. The best way to use the documentation is to bring up the Home page in your browser and start exploring.

If you want a list of other resources and manuals that might be useful in understanding and working with the BEA Tuxedo system, click Site Map on the Home page, and then click Bibliography on the Site Map page.

Site Map

The site map page lists all the documents in the online information set. The documents are grouped by categories such as Installation, Getting Started, Administration, and Programming.

To open a document, click the document name.

PDF Document Files

The PDF Files page lists all the documents that are available in Adobe Acrobat PDF format. The documents are group by categories such as Installation, Getting Started, Administration, and Programming.

To open a PDF file, click the document name. Once the PDF is displayed you also have the option of printing it.

Using the Online Search Feature

The BEA Tuxedo online documentation includes a Java search applet, a platform-independent search tool, to assist in locating topics in the BEA Tuxedo online documentation. The search applet enables you to search for one or more keywords and returns a list of target HTML pages.

When using the search applet, keep these rules in mind:

- Searches are not case sensitive.
- Do not use “quotes” in your query.
- When doing wildcard searches, use the asterisk (*) as a suffix wildcard character in keywords. For example, enter (without quotes) “program*” to find pages with keywords such as program, programmer, or programming.

To perform a search, follow these steps:

1. Click Search in the top navbar. The Search window appears.
2. In the Search field, select the desired search category. If you accept the default, “All Topics,” as the search category, all documents in the online documentation will be searched. Other search categories limit the search to a specific set of documents, such as ATMI or CORBA documents. To see the search category choices, click the Search field drop-down button.
3. Enter the keyword in the Query field and click Find or press Enter on your keypad. The search results appear.
4. If no matches are found, reword your query and try again. If matches are listed, double-click a matched entry; or, highlight the entry and click Show.
5. When the destination page appears, you can enter Ctrl+F to use the browser Find function to locate the keyword on the page.

Learning Paths

To help you find the information you need, the following table lists user tasks and the documentation appropriate to each.

Table 5-2 Learning Paths

If You Want to . . .	You Need . . .	On the BEA Tuxedo Home Page, Click . . .
Evaluate the product	A high-level overview of the BEA Tuxedo system.	• Product Overview • Interoperability • ATMI Introduction • Getting Started with CORBA Applications
Install the software	Step-by-step procedures for installing and configuring each of the BEA Tuxedo system components.	• Installation • Upgrade Information • Migration Information
Design or architect a system	To know (1) BEA Tuxedo system capabilities, (2) the benefits these capabilities give you, (3) how to incorporate the benefits of the BEA Tuxedo system into your design, and (4) how to integrate applications in an BEA Tuxedo environment.	• Interoperability • ATMI Introduction • ATMI and CORBA Programming • System Administration • Reference • ATMI Tutorials • Messages
Write client or server applications	To know how to write, build, configure, and run applications.	The same topics as for design or architect.
Administer the system	To know how to configure, monitor, tune, migrate, and manage the BEA Tuxedo system.	• System Administration • Migration Information • Interoperability

Table 5-2 Learning Paths (Continued)

If You Want to . . .	You Need . . .	On the BEA Tuxedo Home Page, Click . . .
Learn about using BEA Jolt with the BEA Tuxedo system and BEA WebLogic Server	To know (1) how to configure and integrate BEA Jolt with BEA Tuxedo applications so that Tuxedo services are available to customers on the Internet and (2) how to use, configure, and integrate BEA Jolt to work with the BEA Tuxedo system and BEA WebLogic Server.	• Jolt Documentation
Learn about using BEA SNMP Agent with the BEA Tuxedo system	To know how to configure BEA SNMP Agent to manage BEA Tuxedo applications.	• Site Map to BEA SNMP Agent

BEA dev2dev Online

BEA dev2dev Online is a support Web site for BEA customers. It contains unsupported code examples and tools which may assist you in developing applications for BEA software.

Accessing Unsupported Code Examples

To access the unsupported code examples, follow these steps:

1. In your Web browser, enter `http://www.bea.com` to display the BEA corporate Web page.
2. Under Resources select Developer. The dev2dev Online site (http://dev2dev.bea.com) displays providing access to a variety of code samples and relevant technical information for the BEA developer.
3. Click Log in and enter your username and password. If you do not have a password, click “Become a Member” and complete the registration information so as receive a username and password.

BEA Consulting Services

BEA provides a suite of consulting services that will enable you to quickly transform your business into an e-business. BEA consultants have the deep product knowledge and experience to design, develop, and implement solutions right the first time. And we provide full knowledge

transfer to your team throughout your projects so your internal developers will become proficient faster.

BEA consulting solutions are designed to help you at any stage of your process, from gathering business requirements, to legacy and packaged application integration, to application development. BEA packaged consulting solutions can provide you with predefined consulting services to help you meet your time-to-market needs for your e-business.

For a description of the BEA Professional Services offerings and contact information, follow these steps:

1. In your Web browser, enter `http://www.bea.com` to display the BEA corporate Web page.
2. Click the Service drop-down menu and select Consulting Services. The BEA Consulting Services Web page appears.

BEA Education Services

BEA Education Services delivers integrated, client-driven education solutions that help ensure successful use of BEA products. In today's competitive e-business driven market, there is also an increasing need for organizations to quickly and effectively acquire skills to adopt new technologies, such as the Java 2 Enterprise Edition (J2EE), that are driving the next generation of distributed enterprise applications. BEA Education Services offers training in these technologies, which are incorporated in the BEA WebLogic suite of products.

For more information on BEA Education Services offerings and contact information, visit the BEA Education Services Web page at `http://www.bea.com/education`.