THE ENTERPRISE MIDDLEWARE SOLUTION

BEA TUXEDO

Product Overview

BEA TUXEDO Release 6.5
Document Edition 6.5
February 1999

BEA TUXEDO Product Overview

Document Edition	Date	Software Version
6.5	February 1999	BEA TUXEDO Release 6.5

Contents

BEA TUXEDO Product Overview

Introduction

The BEA TUXEDO system is about communication between processes. Most often the process that starts the communication is a client and the process that receives the communication is a server.

The BEA TUXEDO system is also about being able to architect, design, configure, and operate a distributed, business-critical application as a single unit.

The original design of the product was based on the classic design of OnLine Transaction Processing (OLTP) software in which a limited range of client requests is passed for quick bursts of processing against a central store of data. A frequently-used example of an OLTP application is a bank Automatic Teller Machine (ATM); the range of services offered to customers is limited—deposit, withdrawal, and a few others—and the store of data is the bank's customer record database.

Over the years since its introduction, the BEA TUXEDO system has been extended and enhanced in a number of significant ways, always with the intent of making communication between processes easier and more flexible.

BEA TUXEDO System Components

To introduce the product we are going to break the BEA TUXEDO system down into five components. These are not components in the sense of being packaged or sold separately; indeed the product is fully integrated and is sold as a single unit (with add-on options). However, the parts we want to highlight *are* components in the sense that each is a piece of the total capability of the BEA TUXEDO system that you may elect to use or not use.

Here are the five components:

BEA TUXEDO System (also known as /T)
: provides the bedrock client-server architecture, request/response and conversational communications interfaces, transaction support, and application administration for a distributed application

BEA TUXEDO Workstation (also known as /WS)
: extends the client side capability to intelligent workstations

BEA TUXEDO /Q
: provides store-and-forward queue management

BEA TUXEDO Domains (also known as /Domain)
: offers the ability to connect applications that are logically (and perhaps physically) separate so that the combination appears to the user to be a single unit

BEA TUXEDO TxRPC
: adds remote procedure calls. BEA TUXEDO TxRPC follows the X/OPEN and OSF DCE standards for the Interface Definition Language and for remote procedure calls.

Purpose of the Overview

In this overview we will look at each component from three points of view: from the point of view of the designer or architect of a BEA TUXEDO application, from the point of view of a programmer, and finally, from the point of view of the implementer or administrator of such an application.

BEA TUXEDO System

Features

- Application Programming Interface (API): a rich set of communication techniques for writing distributed applications called ATMI, a superset of X/Open's XATMI interface; COBOL and C language bindings supported
- X/Open TX Compliance: X/Open's interface for transaction demarcation; COBOL and C language bindings supported
- X/Open XA Compliance: allows XA-compliant database systems (called resource managers) to be mixed and matched within the same application while allowing total data integrity
- Web-based administration: a graphical user interface for the configuration and control of BEA TUXEDO applications, which is available through the World Wide Web.
- Distributed Services: allows transparent access to application and/or system services located on different hardware platforms
- Distributed Transaction Processing (DTP): allows work being done throughout a distributed application to be atomically completed--an essential characteristic of OLTP
- Typed Buffers: provides transparent handling of application data across heterogeneous platforms
- Password Security: allows application designers to control access by requiring users to furnish an application password or by accepting only users known to an authentication routine
- Access Control Security: is a level of security that allows an administrator to organize users into groups and associate the group with objects that can be accessed.

Architectural Aspects of the BEA TUXEDO System

A fundamental question an application designer may ask about a software product is, “What does this software do that will help my business?” The description of BEA TUXEDO features that follows will, we hope, provide answers to that question.

Client/Server Computing

The term *client/server computing* has been applied to so many cases that its meaning has become fuzzy.

In our definition, clients are responsible for bundling data items gathered from outside the application or computer and forwarding these bundles to servers for processing. Client processes are often made available to users through devices such as ATMs and data entry terminals.

Servers are software modules responsible for processing requests and sending replies back to clients. Services are application-specific code within servers that process clients’ requests. Often the processing calls for accessing a database.

BEA TUXEDO System’s Enhanced Client/Server Model

The BEA TUXEDO system earns its reputation as an enhanced Client/Server model by virtue of numerous features, including:

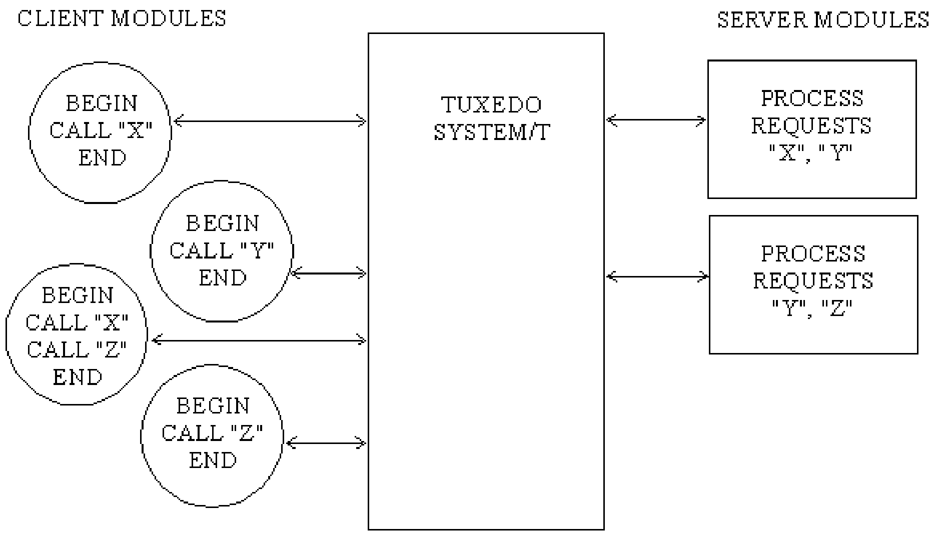
- the ability to spawn copies of server modules in response to client demand
- keeping communications links hidden from programmers and users so that the only information needed to send a request is the name of the service
- automatic data transformation as needed between dissimilar machines
- allowing several service requests from the same client to execute in parallel, thus increasing throughput
- the ability to assign priorities to client requests

BEA TUXEDO system applications are modular and extensible; the BEA TUXEDO system provides a natural way to structure the many different components of a distributed OLTP or general-purpose client/server application into well-defined units or programs.

When the BEA TUXEDO system is used with a commercial UNIX DBMS, the BEA TUXEDO system enhanced client/server architecture demands much less in the way of system resources than the same DBMS without the BEA TUXEDO system. This is because relatively few BEA TUXEDO system servers can service literally hundreds of clients. In fact, in benchmark tests with commercial UNIX DBMSs, the BEA TUXEDO system has increased performance, measured in transactions per second, by a factor of 40.

Figure 1 provides a high-level glimpse of the client/server architecture.

Figure 1 Client/Server Architecture



Web-based Graphical User Interface for Administration

The Administration Console is a new interface for the configuration and administration of BEA TUXEDO applications. It provides a helpful set of tools, principal among which are the “Tree” and the “Configuration Tool.” The Tree is a graphical depiction of the hierarchical composition of the objects that make up a BEA TUXEDO application. The Configuration Tool is a set of “tabbed folders” on the screen that display data and prompt the administrator for the information needed to configure any class of administrative objects. In addition, the context-sensitive help feature allows the user to get detailed information about any GUI tool or field for which input is required.

Event Broker/Monitor

The BEA TUXEDO system has defined a set of almost three dozen system events that are detected and reported on. The list of system events tracked is published in `EVENTS`(5). When an event is detected the Event Broker/Monitor is informed by the system; it, in turn, passes the information along to any subscribing processes. An event can be as innocuous as a change in the system configuration or as potentially harmful as a failure in the system network. Subscribers sign up to receive information for as many of the monitored events as they are interested in. They can also choose to receive notification in several ways: it can be sent via unsolicited notification, it can be queued to stable storage via the Queue Management facility, or the event can be used to kick off a service. To illustrate, notice of a message queue blockage could be used to start a server that activates more servers in that group. An application also has the ability to define its own set of events, publish a list like that in `EVENTS`(5), and allow users to subscribe.

Management Information Base (MIB)

The BEA TUXEDO system can be conceptually divided into several major components or subsystems such as the core system, Workstation, /Q, and others. A component contains classes of objects; classes have attributes for which a range of values can be defined. When all of the classes and their objects are viewed as one larger object, that object is referred to as a MIB. There are several MIB entries in Section 5 of the *BEA TUXEDO Reference Manual* and probably the quickest way to grasp the idea of MIBs is to spend a few minutes with one of those entries.

The significance of the MIBs lies in the fact that they provide the framework for programmed administration of a BEA TUXEDO system. By changing the value of an attribute an administrator or operator can change the behavior of the system and can do it by executing programs written specifically to do so. A MIB entry in the *BEA TUXEDO Reference Manual* not only tells the allowed values for an attribute, it also tells the circumstances under which the value can be changed. For example, the description of the attribute `TA_STATE` in class `T_SERVER` of the `TM_MIB` tells how the administrator can use the attribute to change the state of a server (and provides a list of valid states.) It also says that the value can be changed by the administrator or the operator when the server is active; others can retrieve the state, but may not change it. The real power of this lies in the fact that a program can be written to make the change when some condition is detected.

Distributed Services

The BEA TUXEDO System provides for the connectivity of a network of loosely coupled computers running clients and servers within the framework of a single application. Application designers have great flexibility when choosing platforms on which to implement applications. BEA TUXEDO system networking libraries work with either the Transport Layer Interface (TLI) library or Sockets interface, and can also successfully mix the two interfaces within one application. What this provides, at the application level, is independence from the underlying network. In addition, because the BEA TUXEDO system networking libraries hide the specifics of a particular transport mechanism, other networking interfaces, such as X.25 or SPX/IPX, can be seamlessly introduced.

Distributed Transaction Processing (DTP)

The DTP capability of the BEA TUXEDO system guarantees the integrity of data accessed across several sites or managed by different database products. The BEA TUXEDO system coordinates distributed transactions to allow multi-site updates against heterogeneous databases on networked computers. The BEA TUXEDO system tracks transaction participants via the notion of a global transaction and supervises a two-phase commit protocol, ensuring that transaction commit and rollback are properly handled at each site. For instance, in a banking example of transferring money between accounts, a transaction will ensure that the debit to the “from” account won't happen unless the credit to the “to” account occurs, guaranteeing consistency.

The BEA TUXEDO system incorporates X/Open’s TX interface for transaction demarcation. This interface allows an application writer to bracket a group of operations such that all of them will be done or none of them get done.

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

Typed Buffers

BEA TUXEDO system applications communicating with one of the available API methods 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 designers 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. The BEA TUXEDO system provides several different kinds of typed buffers, but also allows application designers to define their own.

X/Open XA Compliance

The BEA TUXEDO system incorporates X/Open's XA Interface standard for coordinating transactions among multiple database managers participating in an application. The XA interface is part of X/Open's DTP model. Having this interface in the BEA TUXEDO system gives a customer the following benefits:

- Customer Choice of Database: Any database product that complies with the XA standard may be fully incorporated into a BEA TUXEDO application.
- Seamless Integration: If a customer's application currently uses, for example, an SQL database product and needs to replace it with another, the switch from the old to the new can be done without changing a single line of application code.
- Two-Phase Commit: The BEA TUXEDO system coordinates updates against several heterogeneous database or resource managers.
- Multi-Site Update: Ability to access data across multiple sites within the same transaction (atomic unit); servers at each site may be accessing different database products.

Security

The BEA TUXEDO system provides five levels of application security. This enables application developers to select the degree of security appropriate for the application.

- Level One: the normal read, write and execute permissions of the UNIX operating system. Applications generally are set up to have the ownership of the system assigned to an administrator, with other users running as members of the same group.
- Level Two: access control lists (ACLs). The administrator sets up and maintains two files that list users and server groups, and a third file that maps users to groups.
- Level Three: application password security. The administrator establishes a password that users must enter as they join the application.
- Level Four: full client authentication. Users attempting to gain access to the application are validated by a system-provided authentication service.
- Level Five: full client authentication plus ACL access checks. Access to services, events, and queues are validated against defined Access Control Lists.

Documentation

For a more in-depth discussion of the BEA TUXEDO system we recommend the *BEA TUXEDO Application Development Guide*, the *Administering the BEA TUXEDO System*, and the chapter entitled “The BEA TUXEDO System Development Environment” in the *BEA TUXEDO Programmer’s Guide.*

Programming Aspects of the BEA TUXEDO System

Once application architects have agreed on the overall design of a BEA TUXEDO application, programmers have a number of possibilities.

Introduction to the BEA TUXEDO API

The BEA TUXEDO system provides an API that offers both library-based and language-based programming. Library-based programming involves the use of a set of C or COBOL procedures. The BEA TUXEDO procedure library is known as the Application-to-Transaction Manager Interface (ATMI). It is a superset of X/Open's XATMI and TX interfaces.

Two additional choices are the BEA TUXEDO system's Field Manipulation Library (FML) and UFORM, a set of statements for describing data entry forms.

The BEA TUXEDO API provides a remote procedure call facility called TxRPC, which is a language-based programming paradigm. (It is described later in this chapter.) With this facility the application defines the procedures by means of an Interface Definition Language (IDL).

ATMI

Selected by X/Open as the reference technology for OLTP application programming, ATMI was designed to support the enhanced client/server architecture of the BEA TUXEDO system; it is well suited to support OLTP applications.

ATMI lets programmers define transaction boundaries within their application so the work performed within services can be treated as an atomic unit. That means, within a single BEA TUXEDO transaction, the work performed in various services, accessing various databases across many computers, is either committed or rolled back as a single atomic unit of work. This keeps all the databases synchronized, even if there are machine failures.

ATMI also supports a conversational paradigm. Several types of applications, for example, database servers, require the notion of a conversation with a server during which context is kept from message to message. Application programmers can use the conversational functions to establish and maintain state-preserving connections between the requesting process and conversational server processes. Specifically, programmers can do the following:

- Open a connection to a conversational server
- Begin and end a global 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 system automatically spawns a new copy of a server if one is not available when a conversational connection is requested.

Client and server processes may be written in either the COBOL or C programming language. An application may contain both languages. Alternatively, any one of 20 or more Rapid Application Development (RAD) tools can be used.

FML

If the BEA TUXEDO System Field Manipulation Language (FML) and its fielded buffer concept are specified by the application designers, programmers have a rich array of functions for the definition and management of FML fields and 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.

The /AdminAPI

The BEA TUXEDO /AdminAPI is more a concept than a set of function calls. It refers to the use of the ATMI, FML buffers, and FML32 buffers to bring about programmed administration of an application. Programs can be written to manipulate the value of attribute fields in the Management Information Base (MIB). The programs can be set up to run at a certain time or when certain conditions are detected. This means that the administration of a BEA TUXEDO application can be planned in advance and automated; it no longer needs to rely solely on the judgement and attention span of a person at a system console.

Documentation

For a more in-depth discussion of BEA TUXEDO programming, we recommend the *BEA TUXEDO Application Development Guide*, the *BEA TUXEDO Programmer's Guide*, the *BEA TUXEDO FML Programmer's Guide*, the *BEA TUXEDO COBOL Guide*, and of course, Section 3C of the *BEA TUXEDO Reference Manual*.

Implementation and Administration Aspects of the BEA TUXEDO System

After the shape of a BEA TUXEDO application has been determined by the application architects, and in parallel with the development of application clients and services, the work of specifying the configuration can begin.

Configuration and Administration

The BEA TUXEDO system provides a rich set of configuration and administration facilities required for production OLTP applications. The following are some of the features provided for the BEA TUXEDO application administrator:

- Application Definition: The BEA TUXEDO system allows application developers to define in one configuration the hardware, software and networking resources that make up an OLTP application. Application designers can state where servers and services should run as well as where they should be migrated in the event of a processor failure. They may assign various characteristics to the application's servers, including scheduling information, process recovery criteria and time-out periods. The BEA TUXEDO system provides for central configuration management and tools for dynamically starting, stopping or administering a distributed OLTP application.

- Dynamic Reconfiguration: Servers can be dynamically started or stopped; services can be selectively made available. New machines, groups, servers and services can be added to a configuration without taking the application out of service. In addition, various parameters such as time-out intervals, priorities and load factors may be changed dynamically. If a processor fails or needs maintenance, the BEA TUXEDO system allows the servers and services on the out-of-service processor to be migrated to another processor without interruption to the running application.

- Dynamic Administration/Tuning: With mission-critical OLTP applications, it may often be necessary to respond to application demands at a moment's notice; in essence, to be able to alter key application definitions without bringing the application down. With the BEA TUXEDO system, additional capabilities in this area are provided to permit permanent changes to a running system.

- Robust Runtime Environment: To enhance an application's availability, the BEA TUXEDO system has extensive robustness features built in. These include process viability checks, time-out checks, automatic server restart and recovery procedures and user-definable recovery procedures. An application can specify (down to the server level) that access to the BEA TUXEDO system's internal tables should be `PROTECTED` for greater stability, or `FASTPATH` for faster processing.

- Data-dependent Routing: Application data can be partitioned across groups of servers that participate in the application and can be accessed by data-dependent routing of service requests. This enhances the distribution of services and the resiliency of the application to processor failures. The underlying routing is transparent to the application programmer providing data location transparency and can be easily changed based on business needs.
- Load Balancing: To ensure maximum throughput, the BEA TUXEDO system automatically performs load balancing and scheduling throughout the system. By using per-service load factors and keeping totals on outstanding work, the BEA TUXEDO system delivers a particular request to the server that can process the request most quickly.
- Internationalization: The internationalization feature enables the BEA TUXEDO system to furnish diagnostic and system messages in a language appropriate to the locale. All output messages are stored in catalogs so they can be easily translated and modified as needed. With this feature, date, time and currency representations can also be tailored to conform to the conventions of the user's country. The BEA TUXEDO system's implementation conforms to X/Open's Internationalization guidelines as specified in the X/Open Portability Guide (XPG).

Documentation

For a more in-depth discussion of BEA TUXEDO administration, we recommend the *Administering the BEA TUXEDO System*, and Sections 1 and 5 of the *BEA TUXEDO Reference Manual*.

BEA TUXEDO Workstation

Features

Originally client processes had to be located on a machine with a full BEA TUXEDO installation. The Workstation feature moves client processes off the main BEA TUXEDO machine.

- The following workstation platforms are supported:
  - UNIX System Workstations
  - OS/2 System Workstations
  - Windows 3.*x* and Windows95 Workstations
  - Windows NT Workstations
  - Macintosh Workstations
- Same APIs as for BEA TUXEDO Clients
- Languages: COBOL and C language binding supported for request/response, conversations and rpc communications; superset of X/Open's XATMI and TxRPC interfaces
- X/Open TX Compliance: COBOL and C language binding of X/Open's interface for transaction demarcation
- Security: provides additional security needed when accessing BEA TUXEDO applications from non-secure sites
- Workstation Handler: located on the native BEA TUXEDO machine; acts as a surrogate process for Workstation clients

Architectural Aspects of BEA TUXEDO Workstation

BEA TUXEDO Workstation allows customers to use ATMI's full client-side functionality with a UNIX System-based OLTP application from the PC/workstation tier. Each instantiation of BEA TUXEDO Workstation provides the power of the BEA TUXEDO system client functionality for the workstation tier without needing to support the full BEA TUXEDO system core capability on the workstation. Because the programming interface is the same as that used by application programmers in any BEA TUXEDO system environment, no retraining is needed to use the ATMI in BEA TUXEDO Workstation.

Client development can be done on a workstation of the desired platform in a central laboratory and moved as binary files to the field locations. This feature also provides the advantage of removing terminal character processing or graphical processing overhead from the UNIX server CPUs in an open OLTP application. Character handling overhead associated with character-based ASCII terminals is extremely high. Processing associated with GUIs uses significant resources. The benefit of moving that portion of an open OLTP application to the intelligent processing available at the workstation tier is that it results in more efficient use of both the workstation tier and the server tier.

BEA TUXEDO System Workstation DLL

The BEA TUXEDO System Workstation DLL (Dynamic Link Libraries) product takes the next step by extending the Workstation programming interface to the Microsoft Windows, Windows NT and OS/2 environments. This capability allows for more efficient use of memory by having a single copy of a library support various applications. With improved memory management, applications can process larger amounts of data than before. Using this feature, it is now possible to invoke the BEA TUXEDO system libraries from interpretive environments such as Visual Basic, ObjectVision and SQL Windows.

Workstation Handler

A multi-threaded gateway process referred to as a workstation handler resides on the native BEA TUXEDO system side to handle communication between Workstation clients and native BEA TUXEDO servers.

Programming Aspects of BEA TUXEDO Workstation

Client-Side ATMI for MS-DOS, Windows and Windows NT Workstations

From the MS-DOS, Windows and Windows NT workstation tier, customers are provided with the same client functionality available to users logged directly onto the core BEA TUXEDO system. Client processes may be written in either the COBOL or C programming language.

Communications between a BEA TUXEDO system MS-DOS client and a BEA TUXEDO system application are provided for by Novell's LAN Workplace for DOS, which uses TCP/IP as the communications protocol. Networking under Windows makes use of the standard "Windows Sockets Interface."

Client-Side ATMI for UNIX System Workstations

From the UNIX workstation tier, customers are provided with the same client functionality available to users logged directly onto the core BEA TUXEDO system. Client processes may be written in either the COBOL or C programming language.

Client-Side ATMI for OS/2 System Workstations

From the OS/2 workstation tier, customers are provided with the same client API available to users logged directly onto the core BEA TUXEDO system in character mode. 32-bit libraries are provided. Client processes may be written in either the COBOL or C programming language.

Implementation/Administration Aspects of BEA TUXEDO Workstation

Security

BEA TUXEDO Workstation provides extended security that supports Kerberos or other authentication systems that do login authentication for the Workstation clients. This feature provides the additional security needed to authorize clients accessing a UNIX-based OLTP application from a non-secure MS-DOS or UNIX workstation environment.

Documentation

For a more in-depth discussion of BEA TUXEDO Workstation, we recommend the *BEA TUXEDO Workstation Guide*.

BEA TUXEDO /Q

Features

- ♦ Queue Messages: ability to queue a message to stable storage for later processing
- ♦ Administrative Support: provides administrators of BEA TUXEDO applications with functions for processing messages on the queues
- ♦ X/Open XA Compliance: the queue space is a resource manager that is compliant with X/Open's XA interface to ensure data integrity in a distributed environment

Architectural Aspects of BEA TUXEDO /Q

The BEA TUXEDO /Q feature makes it possible for an application, within a global transaction, to enqueue a message to stable storage for processing later. One advantage is that when BEA TUXEDO is used in an environment where a machine, server, or resource is unavailable or unreliable, as in a wide-area network, /Q can provide continuous processing, storing messages for processing when available. Another advantage is in batch processing of a transaction that could be long running. You are guaranteed the message will eventually be processed so you do not have to wait until it completes. This feature can also be used for work flow provisioning where each step generates a queued request to do the next step in a process. Properties such as data-dependent routing and priority handling are kept intact for queue-based requests and replies.

BEA TUXEDO /Q may be combined with BEA TUXEDO Workstation to enqueue and dequeue messages from Workstation clients. The interface is available to both the COBOL and C programming languages.

Programming Aspects of BEA TUXEDO /Q

In addition to the whole programming vocabulary available to BEA TUXEDO programmers, two others are available when the application is using /Q:

- `tpenqueue`(3c) stores a message on a named queue
- `tpdequeue`(3c) dequeues a message for processing

The two functions are available in both C and COBOL.

Implementation/Administration Aspects of BEA TUXEDO /Q

The administrative functions of BEA TUXEDO /Q provide the system administrator with a great deal of flexibility in establishing and managing the queues. They allow the system administrator to configure the following BEA TUXEDO system servers provided with the queued message facility:

- ♦ A “message queuing” server (`TMQUEUE`) enqueues and dequeues messages on behalf of clients and servers. This allows for transparent enqueueing and dequeueing of messages, whether or not the process is local to the queue.
- ♦ A “forwarding” server (`TMQFORWARD`) dequeues queued messages and forwards them for processing. This allows for transparent processing of enqueued messages by existing BEA TUXEDO system servers that are unaware whether the incoming message was sent as a normal request/response message or via the stable queue. A response by the server is automatically enqueued to the associated reply queue for the message.

Documentation

For a more in-depth discussion of BEA TUXEDO /Q, we recommend the *BEA TUXEDO /Q Guide*.

BEA TUXEDO Domains

Features

- A domain gateway, a BEA TUXEDO system program that enables access to and from remote domains. Two particular gateway instantiations will be introduced initially:
  - /TDOMAIN, an instantiation of a domain gateway that allows the interoperability of two or more BEA TUXEDO applications via a specially designed TP protocol that flows over network transport protocols such as TCP/IP.
  - /OSITP, a particular instantiation of a domain gateway that allows application interoperability via the OSI-TP protocol of the International Standards Organization. The OSI-TP instantiation, available as the BEA Connect OSI TP product, is an optional feature.
- A gateway administrative server, a BEA TUXEDO program that enables run-time administration of a particular domain gateway group.
- A domain administrative server, a BEA TUXEDO program that enables run-time administration of the configuration information required by domain gateway groups.
- An administrative interface for configuration and run-time administration of the information required by domain gateways for the interoperation with other domains.
- A facility for building customized domain gateways.

Architectural Aspects of BEA TUXEDO Domains

The BEA TUXEDO system provides an application programming framework that simplifies the development of open OLTP applications and hides the complexity associated with the distribution of the application. The framework is an extended client/server model that offers four programming paradigms: request/response, connection-oriented processing, enqueue/dequeue, and unsolicited notification. These paradigms are provided through the Application Transaction Management Interface (ATMI), and are designed to meet the requirements of open OLTP systems. ATMI helps open OLTP application developers structure their code for portability, location transparency, performance, modular growth, and reliability.

Domains extends the BEA TUXEDO system client/server model to provide transaction interoperability across TP domains. 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. Domains makes this possible via a highly asynchronous multi-tasking gateway that processes outgoing and incoming service requests to or from remote domains.

A Domains gateway is a BEA TUXEDO system-supplied server that enables access to and from remote domains. Domains gateways (DGW) run as a Multiple Server Single Queue set (MSSQ) where each gateway uses a common request queue and has its own reply queue. In addition, Domains provides a gateway administrative server (GWADM) that enables run-time administration of the Domains gateway group and a Domains administrative server (DMADM) that enables run-time administration of the Domains configuration information. Domains gateways support the following functionality:

- ATMI’s request/response model: Application programs using the BEA TUXEDO system can request services from applications running in another domain. Also, remote applications can request services from local servers. No changes are required to the application programs.
- ATMI’s conversational model: Application programs can establish conversations with programs running in another domain. Remote domains can establish conversations with conversational services offered by local servers.
- Transaction management: Application programs can interoperate with other domains within a transaction. The gateway coordinates the commitment or rollback of transactions running across domains.

- Administration: Gateways can be booted or shut down exactly as any other BEA TUXEDO server. Run-time administration is provided through an administrative server, `DMADM`. Using `DMADM`, application administrators can make changes to the domain configuration file, and tune the performance of a gateway group.
- Typed buffer support: Gateways can perform encoding and decoding operations for all typed buffers defined by the application.
- Multi-domain interaction: Gateways can communicate with multiple domains of the same type.
- Multi-network support: Gateways can communicate with other domains via a variety of networks such as Ethernet, Novell and others. However, a gateway is limited by the capabilities of the specific networking library linked with the gateway. In other words, a gateway typically supports a single type of network.

Programming Aspects of BEA TUXEDO Domains

The important point here is that programmers on a BEA TUXEDO application using the Domains feature deal with client/server communication in exactly the same way they would on an application that does not use Domains. The full set of API choices is available.

Implementation/Administration Aspects of BEA TUXEDO Domains

The application administrator enables remote domain access by specifying a gateway group and a domain administration group in the `GROUPS` section of the `TUXCONFIG` file, and by adding entries for the gateway and the two administrative servers in the `SERVERS` section.

Documentation

For a more in-depth discussion of BEA TUXEDO Domains, we recommend the *BEA TUXEDO Domains Guide*.

BEA TUXEDO TxRPC

Features

- The BEA TUXEDO system’s implementation of the X/Open TxRPC interface
- TxRPC is an extension of OSF’s DCE RPC facility to support transactional communication.
- The BEA TUXEDO system’s implementation adds new interface attributes.
- Clients call a remote service using a local function call.

Architectural Aspects of BEA TUXEDO TxRPC

BEA TUXEDO TxRPC gives application architects a language-based Request/Response paradigm.

Remote procedures are defined using an interface definition language (IDL) that specifies the procedure's parameters and return values plus additional information.

The set of interface definitions developed for an application can be codified and shared among all programmers on a project.

TxRPC can be used from BEA TUXEDO Workstation clients and in inter-domain communication. Applications can either use any of the typed buffers provided by the BEA TUXEDO system, or add buffer types of their own.

Programming Aspects of BEA TUXEDO TxRPC

BEA TUXEDO TxRPC manages the connection between the client and server processes transparently so that application programmers can concentrate on the work of the application.

The following shell commands and functions are added:

- `bldc_dce`(1) builds a BEA TUXEDO client that acts as an OSF/DCE -> BEA TUXEDO gateway.
- `blds_dce`(1) builds a BEA TUXEDO server that acts as a BEA TUXEDO -> OSF/DCE gateway.
- `tidl`(1) is the BEA TUXEDO IDL compiler.
- `uuidgen`(1) generates a universal unique identifier for an IDL interface definition.

  Manual pages for these shell commands can be found in Section 1 of the *BEA TUXEDO Reference Manual*.
- `TRY`(3c) is an exception-returning interface for TxRPC.
- `rpc_sm_* rpc_ss_*` functions make up a set of functions for memory management in RPC stubs and exception returning functions.

  Manual pages for these RPC functions can be found in Section 3c of the *BEA TUXEDO Reference Manual*.

Implementation/Administration Aspects of BEA TUXEDO TxRPC

Defining TxRPC servers in a BEA TUXEDO system configuration file is no different from defining a Request/Response server—unless the remote procedure call goes through a DCE Gateway process. In that case both the gateway and the DCE server must run on a POSIX platform that has DCE software installed on it. In the BEA TUXEDO system configuration file a gateway server group needs to be defined and DCE entities have to be added to the DCE Registry.

Documentation

Examples of TxRPC applications (both with a Gateway and without) are included in the *BEA TUXEDO TxRPC Guide.*

Summary

The BEA TUXEDO system is considered the de facto standard for open OLTP solutions. As the de facto standard, the BEA TUXEDO system offers several capabilities on the leading edge of open OLTP. The BEA TUXEDO system is a mature, proven, and widely available product currently used as the benchmark against which other UNIX-based OLTP transaction monitors are compared. This combination yields a state-of-the-art technical solution capable of providing industrial-strength open OLTP. Workstation is a product extension which pushes the definition of open OLTP past the UNIX environment into multiple workstation environments. BEA TUXEDO System/Q guarantees and schedules message delivery.

The BEA TUXEDO system's architecture and many of its features have existed in earlier versions of the product dating as far back as 1978. The current product incorporates these features as well as significant functionality that reflects customer input and technical advances in open systems. Applications have been built with the BEA TUXEDO system since 1983 and the product has been commercially available since 1986 with over 60 applications and over 1,000 sites in production worldwide. The present release represents the sixth generation of the product, which continues to evolve around customer demands for open, affordable, and flexible OLTP technology.