Oracle® Tuxedo

Glossary

10*g* Release 3 (10.3)

January 2009

ORACLE

Tuxedo Glossary, 10*g* Release 3 (10.3)

Glossary

A

abort

Terminate a transaction in such a way that the values assigned to all protected resources, such as database records, are unchanged from the beginning of the transaction.

absolute OID

(*CORBA)* An *object identifier* (OID) that specifies a unique path to a managed object from the root of the OID tree.

See **object identifier (OID)**.

Abstract Syntax Notation One (ASN.1)

A formal notation used to define data types and encode data values. A language that describes the data structures that make up an abstract syntax. ITU-T (formerly CCITT) specification X.409 is equivalent to ASN.1. ASN.1 Distinguished Encoding Rules (DER) object identifiers are used in public key interface (PKI) security programming.

access control list (ACL)

A BEA Tuxedo security feature that controls client access to services, objects, and methods by means of lists of clients that are authorized to receive specific services or invoke specific objects and methods. When ACL security is being used, the BEA Tuxedo system checks the appropriate list whenever a client requests a service, to determine whether the client is authorized to access the service.

Access Decision object

(*CORBA)* The object in the security infrastructure of a CORBA application that enforces the checking of authorized access before a request to the target object is delivered.

access machine

The processor within the administrative domain of an application at which a client first accesses the system. For a native client, this is the processor on which it is running. For a Workstation client, this is the site at which it contacts the application.

ACID properties

The essential characteristics of transaction processing systems include the following:

- Atomicity, which requires that all changes made by a transaction to a database are made permanent, or else all are nullified.
- Consistency, which defines the success of a transaction as the transformation of a database from one valid state to another.
- Isolation, which requires that changes made by a transaction to a database must not be visible to other operations until the transaction completes its work.
- Durability, which guarantees that changes made by a transaction to a database survive future system or media failures.

ACL

See **access control list (ACL)**.

activate

(*ATMI)* To move a server from the inactive (unavailable) state to the state in which it is up and running (active).

(*CORBA)* To prepare an object for execution.

activation policy

(*CORBA)* The policy that determines the in-memory activation duration for a CORBA object.

See also **activate**, **CORBA object**, and **policy**.

active object

(*CORBA)* A running instance of an object interface.

See also **active object map**, **client application**, **CORBA object**, **object ID (OID)**, **object reference**, **Portable Object Adapter (POA)**, and **servant**.

active object map

(*CORBA)* A table maintained by a POA and the TP Framework that maps the association of object IDs to servants.

See also **object ID (OID)**, **Portable Object Adapter (POA)**, and **servant**.

active server

A BEA Tuxedo server that is either processing a service request or is available to do so.

ADE

See **application development environment (ADE)**.

AdminAPI

See **administrative API**.

administrative API

An application programming interface for writing programs that configure and control a BEA Tuxedo application by setting and changing attribute values in the Management Information Base (MIB). It can be used by both ATMI and CORBA programmers.

administrative domain

That portion of an application that is actively administered at run time by a Bulletin Board Liaison (BBL) process. It does not include workstations or host processors.

administrator

The person who installs the BEA Tuxedo system, configures and monitors a BEA Tuxedo application, and updates application information, such as computer names and locations.

advertised

A service is advertised when a service table entry exists for it in the BEA Tuxedo bulletin board. When a Domains gateway server is booted, it advertises all the remote services that it is importing from remote domains in the bulletin board of the local domain (that is, the domain on which the gateway server is booted). After a remote service is advertised by a domain gateway server, it remains advertised until either an `unadvertise` command is issued or a MIB request removes the service.

AEQ

See **application entity qualifier (AEQ)**.

AET

See **application entity title (AET)**.

agent

- A component of a network management system that exchanges data about managed objects with a manager at a network management workstation. At a manager's request, agents provide a software interface to, and gather data about, managed resources.
- In a two-phase commit syncpointing sequence (LU6.2 or MRO), a task that receives syncpoint requests from an initiator (a task that initiates syncpointing activity).

See **SNMP agent**.

agent-manager model

A model where a manager communicates with many distributed agents through a system management protocol.

alarm

A means of reporting that a managed object is in an abnormal state (that is, a managed object has passed a predefined threshold).

allocation

The assignment of various types of programs and record categories for system storage locations, such as main storage or disk storage.

alternate facility

In distributed transaction programming, a session that a transaction obtains by means of an `ALLOCATE` command.

alternate remote domain

A remote domain that is used when a primary remote domain is unavailable.

AP

See **application program (AP)**.

API

See **application programming interface (API)**.

applet

An interactive Java program that can be run via a Web page displayed by a Java-capable browser. An applet enhances the Web page's display or enables the user to complete a task.

application

A business program, built around the BEA Tuxedo system, that is defined and controlled by a single BEA Tuxedo configuration file, and that is administered as a single entity. Such an application may comprise one or more clients (local or remote), one or more servers, and one or more machines. At a minimum, a BEA Tuxedo application includes one machine, one server, and one client. Also referred to as a BEA Tuxedo *domain*.

Multiple BEA Tuxedo applications can communicate with each other through a *domain gateway group*.

Note: In this context, the term *business program* is defined as a set of one or more programs that work together. Similarly, in conversation the term application is often used loosely: it may refer to a standalone program or to a set of programs that work together to accomplish a particular business objective.

See **TUXCONFIG file** *and* **UBBCONFIG file**.

application association

The association between a process and a BEA Tuxedo application (domain). A multicontexted process may have associations with multiple BEA Tuxedo domains. It may also have multiple associations with the same domain.

application code

Code that is written by the user, as opposed to system code that is provided by BEA Systems, Inc.

application context

(*ATMI*) A reference to a particular application association. In the BEA Tuxedo system, an application context is set via an explicit call and is then used implicitly by subsequent ATMI calls. Thus, in an implicit context interface, the terms *application context* and *default context* are often used interchangeably.

application context name (ACN)

A set of rules that govern associations between application entities.

application-controlled deactivation

(*CORBA)* A feature used with the `process` activation policy to keep an object active in memory until the application explicitly deactivates the object by invoking the `TP::deactivateEnable()` operation on that object.

application development environment (ADE)

A set of tools (often presented or accessed via a GUI) to help programmers build applications.

application entity

A set of software components that make up a distributed transaction processing application on one computer system.

application entity qualifier (AEQ)

The locally unique component of a title that is used to identify an OSI TP application entity.

application entity title (AET)

The globally unique component of the title that is used to identify an OSI TP application entity.

application framework

The software that provides the infrastructure that makes it possible for sets of applications or other software components to work together.

application program (AP)

A single instance of a user program that performs one or more specific tasks. An AP defines transaction boundaries and accesses resources within those boundaries; it interacts with other system components using interfaces specified in the X/Open Distributed Transaction Processing model. An AP is a single thread of control involved in, at most, one global transaction at any time.

application programming interface (API)

- The functions and environment that exist at the application level to support a particular system software product.
- A set of code that enables a developer to initiate and complete client/server requests within an application.
- A set of calling conventions that define how to invoke a service. A set of well-defined programming interfaces (entry points, calling parameters, and return values) by which one software program utilizes the services of another.

Application-to-Transaction Monitor Interface (ATMI)

The Application-to-Transaction Monitor Interface is the application programming interface to the BEA Tuxedo system. It includes transaction routines, message handling routines, service interface routines, and buffer management routines.

architecture

- A hardware and software platform (for example, SPARC/Solaris, Intel Pentium/Windows NT, or Intel Pentium/Linux).
- The structure and interrelationship of components in a system or in an environment.

ASN.1

See **Abstract Syntax Notation One (ASN.1)**.

asymmetric algorithm

An encryption algorithm that has two keys: a public key and a private key. The public key can be distributed openly while the private key is kept secret. Asymmetric algorithms may be capable of a number of operations, including encryption, digital signatures, and key agreements.

asymmetric outbound IIOP

See **outbound IIOP**.

asynchronous

An event that occurs at a time that is unrelated to the time at which another event occurs. The two events are mutually asynchronous. The relationship between the times at which they occur is unpredictable.

asynchronous connections

Virtual circuits set up to execute independently of each other, that is, asynchronously. An asynchronous connection does not block the processing of working circuits while attempts are being made to reconnect failed circuits. The `BRIDGE` allows the use of nonfailing network paths by listening and transferring data using multiple network address endpoints.

asynchronous process

A process that executes independently of another process. When a request is processed asynchronously, the client application continues to perform other operations while it waits for the service request to be filled.

asynchronous request

A request that lets the client do other work while the request is being processed, enhancing parallelism within an application.

ATMI

See **Application-to-Transaction Monitor Interface (ATMI)**.

atomic set

A behavior of SNMP agents that operates as follows: When an SNMP agent receives an SNMP set request that contains more than one variable, the agent either sets all requested objects or sets none. This behavior is a requirement of the SNMP standard.

attribute

(*CORBA)* An identifiable association between an object and a value.

When using OMG IDL, that part of an OMG IDL interface that is similar to a public class field or data member. The compiler maps an OMG IDL attribute to accessor and modifier methods in either the C++ or Java programming language. For example, an interface `ball` might include the attribute `color`. The `idltojava` compiler would generate a C++ or Java programming language method to get the `color`, and, unless the attribute is read-only, would generate a method to set the `color`. CORBA attributes correspond closely to JavaBeans properties.

See **CORBA object** *and* **object**.

auditing

A security mechanism that involves saving a secure, tamper-resistant record of requested system operations, along with the identity of the requesting party.

audit trail

A manual or computerized means for tracing the transactions affecting the contents of a record.

authentication

The process used by the server to verify the identity of an entity, such as a user or a process, before allowing that user or process to join an application. This process relies on the use of passwords and other security mechanisms.

authorization

The process of determining which services may be accessed by a particular entity, such as a user or a process, and giving the entity permission to access those services.

autoinstall

A method of creating and installing terminal definitions dynamically (that is, as terminals log on) and deleting them when logging off.

auto-spawning

The creation, by a BEA Tuxedo application server, of new threads to handle messages as they are received. The number of new threads that can be created is restricted by a configurable limit.

availability

Features of transaction processing systems that contribute to the smooth, continued systems operation in the presence of failures.

B

backup

The act of a resource manager in using a log to restore resources to some predetermined state by applying log entries sequentially to the resource until the desired state has been achieved.

bandwidth

The transmission capacity of a computer or communications channel.

BBL

See **Bulletin Board Liaison (BBL)**.

BEA Administration Console

A Web-based, graphical user interface for the remote administration of BEA Tuxedo applications (running in either ATMI or CORBA environments). It is delivered as a Java applet that you can download into your Internet browser.

See also **BEA Tuxedo system**.

BEA Transaction Processing

See **transaction processing (TP)**.

BEA Tuxedo-JSE Connectivity

(*ATMI)* Formerly JoltWAS for Servlet.

BEA Tuxedo-WebLogic Connectivity

Formerly JoltWAS for WebLogic. A customized version of BEA Tuxedo-JSE Connectivity for WebLogic.

BEA Tuxedo application

See **application**.

BEA Tuxedo Bulletin Board

See **bulletin board**.

BEA Tuxedo client

See **client**.

BEA Tuxedo domain

A BEA Tuxedo domain consists of a BEA Tuxedo application running one or more business applications. A single domain is defined in one configuration file and is administered as a single entity. A domain can be connected to other BEA Tuxedo domains through the Domains feature.

See also **domain**, **TUXCONFIG file**, *and* **UBBCONFIG file**.

BEA Tuxedo Domains

A BEA Tuxedo component that extends the BEA Tuxedo system client/server model to provide transaction interoperability across TP domains. This extension preserves the client/server model and the ATMI interface by making access to services on a remote domain (and service requests from a remote domain) transparent to both the application programmer and user.

BEA Tuxedo server

A program that performs a task requested of it by a client application.

See also **server**.

BEA Tuxedo system

A robust e-commerce platform from BEA Systems, Inc. for developing and deploying business-critical client/server applications. It handles distributed transaction processing, application messaging, and the full complement of services necessary to build and run enterprise-wide applications.

BEA Wrapper Callbacks API

(*CORBA*) An application programming interface designed to simplify the implementation of callback objects for CORBA joint client/server applications. The API provides specific methods for defining, starting, stopping, and destroying callbacks objects.

See also **application programming interface (API)**, **Callbacks Wrapper object**, **CORBA callback object**, *and* **joint client/server application**.

bidirectional outbound IIOP

See **outbound IIOP**.

bind

(*CORBA*) The process of associating a name with an application object or a naming context object. Also used to describe the process of connecting a client application to an application object.

bitmap

In temporary storage, a control block used by intrapartition transient data to show the VSAM control intervals that have been used and are available. It is updated whenever a control interval or track is assigned to or released from a destination.

blocking

The process of combining two or more records into one block.

blocking mode

A synchronous style of message delivery, where a program must wait for an action to complete before the program can proceed. Contrast with nonblocking mode.

Bootstrap environmental object

(*CORBA)* The object that brings a CORBA application into a BEA Tuxedo domain and provides initial object references to that application. Every CORBA client or server application that interacts with a BEA Tuxedo domain needs a Bootstrap environmental object.

See **environmental object**, **object**, **object reference**, *and* **CORBA domain**.

bootstrapping

(*CORBA)* The process of setting up an application to interact with CORBA objects that are located within a BEA Tuxedo domain.

See also **Bootstrap environmental object**, **CORBA object**, and **CORBA domain**.

BRIDGE

The BEA Tuxedo system process that maintains virtual circuits to other nodes participating in an application for the purpose of transferring application messages between the nodes.

broadcast

To send the same message to every node on a network.

broker

A system-level entity whose role is to maintain subscriptions and to cause subscribers' actions to occur when events are posted.

buffer types

An abstract name for a message type. BEA Tuxedo provides predefined types for message communication: FML, VIEW, STRING, CARRAY, and XML. These buffer types are transparently encoded and decoded across a network of heterogeneous machines. Applications can define additional types.

bulletin board

A collection of shared data structures designed to keep track of a running BEA Tuxedo system application. It contains information about servers, services, clients, and transactions pertaining to a BEA Tuxedo application. The bulletin board is replicated on each logical native (non-foreign) machine in the application.

Bulletin Board Liaison (BBL)

A BEA Tuxedo administrative process responsible for maintaining a copy of the bulletin board on a particular processor. When the system is running, one BBL process runs continuously on each logical machine in the application.

business object

(*CORBA)* An application-level component that can be used in combinations that may not be defined ahead of time. A business object is independent of any single application and represents a *recognizable,* everyday-life entity, such as a document processor. A business object is a self-contained deliverable that has a user interface and a state, and that can cooperate with other separately developed business objects to perform a desired task.

See also **object**.

byte

A sequence of eight adjacent bits that are operated on as a unit.

C

C++

An object-oriented programming language developed at AT&T Bell Laboratories in the early 1980s. C++ is a "hybrid" language based on the non-object-oriented C language.

cache

A subset of memory that contains copies of the frequently accessed parts of a larger memory.

callback method

A method that is implemented by application code and that is invoked by system code when needed to perform a specific function. Callback methods are never intended to be invoked directly by application code.

***See also* application code** *and* **metadata interface**.

Callbacks Wrapper object

(*CORBA)* An object implemented to support callbacks on CORBA joint client/server applications using the BEA Wrapper Callbacks API.

See **BEA Wrapper Callbacks API**, **callback method**, **CORBA callback object**, **joint client/server application**, *and* **object**.

CARRAY buffer

A data structure that is an array of characters any of which can be the null character. The interpretation of the array is entirely application dependent.

catalog

See **message catalog**.

CCR

The Commitment, Concurrency, and Recovery OSI standard.

certificate

A digital statement that associates a particular public key with a name or other attributes. The statement is digitally signed by a certificate authority (CA). By trusting that authority to sign only true statements, you can trust that the public key belongs to the person named in the certificate.

See also **certificate authority (CA)**.

certificate authentication

A method that provides confident identification of a client by a server through the use of digital certificates. Certificate-based authentication is generally preferred over password-based authentication because it is based on what the user has (the private key) as well as what the user knows (the password that protects the private key).

See **authentication** *and* **certificate**.

certificate authority (CA)

A well-known and trusted entity that issues public key certificates. A certificate authority attests to a user's real-world identity, somewhat like a notary public.

See also **certificate**.

channel

A functional unit, controlled by a processor, that handles the transfer of data between the processor and local input/output devices.

cipher

In cryptography, a coding system used to create encrypted messages.

cipher suite

An SSL encryption method that includes the key exchange algorithm, the symmetric encryption algorithm, and the secure hash algorithm used to protect the integrity of the communication.

See also **Secure Sockets Layer (SSL)**.

ciphertext

In cryptography, text that is encrypted.

class

(*CORBA)* In Java, a type that defines the implementation of a particular kind of object. A class definition defines instances and class variables and methods, and specifies the interfaces and class implementations and the immediate superclass of the class. If the superclass is not explicitly specified, the superclass will implicitly be `Object`.

See also **IDL interface**, **instance**, **Java**, **metadata interface**, *and* **object**.

class library

A set of client programming tools. These tools can be used in a Java or C**++** program or Web page-embeddable Java applet.

client

(*ATMI)* A program that performs the following steps:

1. Collects, through a user interface, requests for services from users.

2. Transmits those requests to servers.

3. Receives the servers' responses and passes them to the users.

If a client is located on a machine that is part of the domain to which the target servers belong, then the client is called a *native client*. If the client is located on a machine that is outside that domain, then the client is called a *remote client* or a *Workstation client*. Remote clients communicate with servers through the BEA Tuxedo Workstation component.

(*CORBA)* Any code that invokes an operation on a distributed object.

See also **application**, **server**, *and* **Workstation**.

client application

A program, written for use with the BEA Tuxedo software, that requests services from other applications.

See also **server application**.

Client Data Caching design pattern

(*CORBA)* The design pattern that provides increased performance for client applications by caching server application data on the machine on which the client application resides, thereby avoiding repeated remote calls to retrieve data.

See also **design pattern**.

client naming

A BEA Tuxedo feature that enables client programs to carry both usernames and client name values.

client program

See **client**.

client/server

See **client/server computing**.

client/server computing

A programming model in which application programs are structured as clients or servers to achieve distributed processing. A client program is an application program that requests services to be performed. A server program is an entity that dispatches service routines to satisfy requests from client programs. A service routine is an application program module that performs one or more specific functions on behalf of client programs.

Client/server computing can be configured in a 2-tier or a 3-tier structure. A 2-tier configuration consists of only the client and the server. A 3-tier configuration includes a client, a server, and an intermediate level that acts as a router or a broker.

closed framework

A software infrastructure in which developers cannot remove and replace software components easily in a *plug-and-play* fashion.

CMIP

See **Common Management Interface Protocol (CMIP)**.

columnar object

A MIB *leaf* object—that is, a MIB object that does not have any objects below it in the OID tree—which can have zero or more instances. A columnar object represents one column in a table.

COM

See **Component Object Model (COM)**.

COM view

(*CORBA)* A representation of an object that conforms to the Component Object Model (COM) standards, including implementations of all necessary interfaces.

See also **Component Object Model (COM)**, **interface**, and **object**.

command-line interface

A style of user interface that allows user interaction by entering commands at a system prompt.

commit

- Complete a transaction so that changes are recorded and stable. Protected resources are released.
- The declaration or process of making a transaction's updates and messages visible to other transactions. When a transaction commits, all its effects become public and durable. After commitment, the effects of a transaction cannot be reversed automatically.

Common Management Interface Protocol (CMIP)

An protocol for network management defined by ISO standards.

Common Object Request Broker Architecture

See **CORBA**.

compact disc-read only memory (CD-ROM)

A disk from which data is read optically by laser and on which the data cannot be modified except under special conditions.

component

Part of an application.

Component Object Model (COM)

(*CORBA)* A collection of services that let software components interoperate in a networked environment.

See also **COM view**, **conversation**, *and* **object**.

concurrency

The simultaneous execution of more than one function or process.

concurrent

Pertaining to the occurrence of two or more activities within a given interval of time. Concurrent processes can alternately use shared common resources.

configuration

The set of hardware, hardware options, software, and software setup on a computer or on a network.

configuration set

The name or the number used to reference a particular configuration in a configuration partition. Each configuration set describes the services to be used when the configuration is active.

configure

To customize hardware and software for a computer or for a network.

connection

A half-duplex communication channel between processes.

connection-oriented communication

Communication between two BEA Tuxedo system processes over a connection.

consistent state

A condition in which shared data is correct and valid.

constructor

(*CORBA)* A pseudo-method that creates an object. In Java, constructors are instance methods with the same name as their class. Java constructors are invoked using the `new` keyword.

See also **class**, **instance**, **Java**, **metadata interface**, *and* **object**.

conversation

A dialog over a connection.

conversational

Communication characterized by one or more messages exchanged by a requestor and a server such that the server remains dedicated to the communication until the termination of the exchange.

conversational communication

See **conversational**.

conversational server

A server that offers services that require a connection to have a conversation with the requester. The conversation follows an application-established protocol. A conversational service must conform to the startup and termination rules of BEA Tuxedo system services.

conversational service

A service routine that is invoked by means of conversational communication from a client program. When the connection is established and the service is invoked, the client and service exchange data in a manner specific to the application. When the service returns, the connection ends.

CORBA

(CORBA) Common Object Request Broker Architecture. A multivendor standard published by the Object Management Group for distributed object-oriented computing.

CORBA callback object

(CORBA) A CORBA object supplied as a parameter in a client application's invocation on a target object. The target object can make invocations on the callback object either during the execution of the target object or at some later time (even after the invocation on the target object has been completed). A callback object might be located inside or outside a BEA Tuxedo domain.

See also **client application**, **CORBA object**, *and* **BEA Tuxedo domain**.

CORBA client stub

When using CORBA objects, a file created by the IDL compiler when you compile an application's OMG IDL statements. The client stub contains code that is generated during the client application build process. The client stub maps OMG IDL operation definitions for an object type to the methods in the server application that the BEA Tuxedo domain calls when it invokes a request. The code is used to send the request to the CORBA server application.

When using OMG IDL, a C++ or Java programming language class created by the compiler and used transparently by the client ORB during object invocation. The remote object reference held by the client points to the client stub. This stub is specific to the IDL interface from which it was generated, and contains the information needed for the client to invoke a method on the CORBA object defined in the IDL interface.

See also **metadata interface**, **OMG IDL**, **skeleton**, *and* **BEA Tuxedo domain**.

CORBA domain

A collection of CORBA or ATMI servers, services, interfaces, machines, and associated resource managers defined by a single `UBBCONFIG` (ASCII version) or `TUXCONFIG` (binary version) configuration.

See also **TUXCONFIG file** *and* **UBBCONFIG file**.

CORBA foreign client application

(*CORBA*) A client application that is implemented on an ORB that is not a product of BEA Systems, Inc.

See **ORB**.

CORBA interface

(*CORBA)* A set of operations and attributes. A CORBA interface is defined by using OMG IDL statements to create an interface definition. The definition contains operations and attributes that can be used to manipulate an object.

See also **attribute**, **interface**, **object**, **OMG IDL**, *and* **operation**.

CORBA native client application

(CORBA) A client application that invokes operations defined in OMG IDL statements to talk to CORBA server applications. Relative to the CORBA domain to which the server applications belong, a client application is either native (that is, local) or remote. Remote and native client applications are the same. Their requests are handled differently and transparently, depending on whether or not the applications are collocated on a machine that is running in the CORBA domain. CORBA native client applications are always collocated on a machine in the CORBA domain.

See also **CORBA domain**, **CORBA foreign client application**, **CORBA remote client application**, **CORBA server application,** *and* **OMG IDL**.

CORBA object

(*CORBA)* An entity that complies with the CORBA standard upon which operations are performed. An object is defined by its interface.

See also **interface**, **object**, *and* **operation**.

CORBA ORB

(*CORBA)* Any Object Request Broker (ORB) that complies with the CORBA standard. A CORBA ORB is a communications intermediary between client and server applications that are distributed across a network. The ORB used in BEA Tuxedo applications is a CORBA ORB.

See also **conversation**.

CORBA remote client application

(*CORBA)* A client application that invokes operations defined in OMG IDL statements to talk to remote CORBA server applications using IIOP. Remote and native client applications are the same. Their requests are handled differently and transparently, depending on whether or not the

applications are collocated on a machine that is running in the CORBA domain. CORBA remote client applications are typically not located on a machine that is running in the CORBA domain.

See **IIOP**, **OMG IDL**, **CORBA domain**, **CORBA foreign client application**, *and* **CORBA native client application**.

CORBA server application

(CORBA) A program that performs a task requested of it by a client application and that is written to be used with the BEA Tuxedo CORBA software.

See **local factory**.

CORBA TP framework

(CORBA) Run-time library of default implementations that the CORBA server application build procedure links to the server application executable image. The Transaction Processing (TP) Framework consists of a set of convenience functions that make it easy for you to write code that does the following:

Initializes the server application and executes startup and shutdown routines.

Ties the server application to CORBA domain resources.

Manages objects, bringing them into memory when needed, flushing them from memory when no longer needed, and managing reading and writing of data for persistent objects.

Performs object housekeeping.

See **CORBA domain** *and* **CORBA server application**.

CORBAfacilities

(*CORBA)* The adopted OMG Common Facilities. Common Facilities provide horizontal end-user-oriented frameworks that are applicable to most applications, and are defined in OMG IDL.

See also **OMG IDL**.

CORBAservices

(*CORBA)* A set of system services for objects that were developed for the programmer. These services, defined in OMG IDL by the OMG, can be used to create objects, control access to objects, track objects and object references, and control the relationship between types of objects. Programmers can call object service functions instead of writing and calling their own private object service functions.

See also **CORBA object**, **CORBAservices Life Cycle Service**, **CORBAservices Naming Service**, **CORBAservices Object Transaction Service (OTS)**, **CORBAservices Security Service**, **object**, **object reference**, *and* **OMG IDL**.

CORBAservices Life Cycle Service

(*CORBA)* The CORBAservice that defines conventions for creating, deleting, copying, and moving objects.

See also **CORBAservices** *and* **object**.

CORBAservices Naming Service

(*CORBA)* The CORBAservice that provides the ability to associate a name to an object relative to a naming context.

See also **CORBAservices** *and* **object**.

CORBAservices Object Transaction Service (OTS)

(*CORBA)* The CORBAservice that provides transaction semantics to ensure the integrity of data in the system.

See also **CORBAservices**.

CORBAservices Security Service

(*CORBA)* The CORBAservice that defines identification and authentication of principals, authorization and access control, security auditing, security of communication between objects, nonrepudiation, and administration of security information.

See also **authentication**, **authorization**, **Callbacks Wrapper object**, *and* **object**.

core class

(*CORBA)* A public class (or interface) that is a standard member of the Java platform. The intent is that the Java core classes, at a minimum, are available on all operating systems on which the Java platform runs.

See also **class**, **interface**, *and* **Java**.

credentials

Information that describes the security attributes (identity and/or privileges) of a user or other principal. Credentials are claimed through authentication or delegation and are used by access control.

See also **authentication**.

Credentials object

(*CORBA)* The object that holds the security attributes of a principal. These security attributes include the principal's authenticated or unauthenticated identities. The Credentials object also contains information for establishing security associations. The Credentials object provides methods to obtain the security attributes of the principals it represents.

See also **attribute**, **metadata interface**, *and* **object**.

cryptography

The art of protecting information by transforming it (encrypting it) into an unreadable format, called ciphertext. Only those who possess a secret key can decipher (or decrypt) the message into plaintext.

See also **ciphertext** *and* **plaintext**.

CSI

The API for the BEA TOP END system.

current context

(*ATMI*) Clients may initialize to multiple contexts; however, at any given time, in any particular thread, only one of these contexts may be the current context.

(*CORBA)* A special type of ORB object that is used to communicate between a user application and a specialized built-in service.

See also **CORBA ORB**, **object**, **SecurityCurrent**, *and* **TransactionCurrent**.

Custom GUI element

(*ATMI)* A Java GUI class that communicates with JoltBeans. The means of communication can be JavaBeans events, methods, or properties offered by JoltBeans.

D

daemon

A system process that runs in the background.

DASD

See **Direct Access Storage Device (DASD)**.

database

A collection of interrelated or independent data items stored together without redundancy to serve one or more applications.

database management system (DBMS)

A program or set of programs that lets users structure and manipulate the data in the tables of a database. A DBMS ensures privacy, recovery, and integrity of data in a multi-user environment.

data-dependent routing

- Routing that directs a request to be processed by a particular group based on the value in a data field of the message.
- A mechanism in the BEA Tuxedo system in which a service request is mapped to a specific server group based on a value contained in a designated field in the data buffer.

Data Encryption Standard (DES)

A symmetric key algorithm adopted by the U.S. government in 1976 as a standard encryption-decryption system for unclassified data in the United States. Several types of DES are offered, including DES-CBC and two-key triple-DES.

- DES-CBC is a 64-bit block cipher run in Cipher Block Chaining (CBC) mode. It provides 56-bit keys; 8 parity bits are stripped from the full 64-bit key.
- Two-key triple-DES is a 128-bit block cipher run in Encrypt-Decrypt-Encrypt (EDE) mode. Two-key triple-DES provides two 56-bit keys (in effect, a 112-bit key).

data independence

The ability to request data by a high-level data-management method without concern as to how the data is stored or retrieved.

data transfer protocol

A set of rules for transforming data of a particular buffer type from one representation into another.

DB2

IBM relational database.

DBBL

See **Distinguished Bulletin Board Liaison (DBBL)**.

DBMS

See **database management system (DBMS)**.

DDE

See **Dynamic Data Exchange (DDE)**.

DDE conversation

The sending and receiving of DDE messages between a client application and a server application.

deadlock

- Unresolved contention for the use of a resource.
- An error condition in which processing cannot continue because each of the two elements of the process is waiting for an action or a response from the other.

decoding

The conversion of encoded data back to the native format.

See also **encoding**.

decryption

The process of restoring encrypted data to its original form.

decryption private key

An algorithm that reverses the work of the encryption algorithm.

default

The value assumed by a program if a value is not supplied by the user.

default context

A BEA Tuxedo application association that is referenced by subsequent ATMI calls if `tpsetctxt()` is not called. The default context may differ from thread to thread. This term is often used interchangeably with application context.

See also **application context**.

deferred synchronous communication

A form of asynchronous communication in which one piece of software can send a message to another piece of software, and then continue to work and retrieve the reply to the message at some later time.

deployment

The process of placing an application in a distributed environment and making the application available for use. Deployment can include such tasks as installation, configuration, and administration of various parts of the application.

DES

See **Data Encryption Standard (DES)**.

design document

The document written by the system integrator that explains the overall design of the application or the framework to be built.

design pattern

A document that encapsulates, in a structured format, solutions to design problems. Design patterns are guides to good design practices.

See also **Client Data Caching design pattern** *and* **Process-Entity design pattern**.

desktop client

(*CORBA)* A CORBA client application that operates on a Microsoft desktop platform, such as Windows XP. Desktop client applications use the Component Object Model (COM) and communicate with the BEA Tuxedo domain.

See **Component Object Model (COM)**, **conversation**, *and* **application**.

dialog

A process of sending and receiving information.

digital certificate

An electronic file used to identify individuals and resources over networks such as the Internet. A digital certificate securely binds the identity of an individual or resource, as verified by a trusted third party known as a Certificate Authority, to a particular public key. Because no two public keys are ever identical, a public key can be used to identify its owner.

BEA Tuxedo public key security recognizes certificates that comply with X.509 version 3.0.

digital signature

A digital code, attached to an electronically transmitted message, that uniquely identifies the sender and that can, therefore, be used to authenticate the identity of the sender. When a message is authenticated, verification is provided that: (1) the message is genuine; (2) the message has arrived exactly as it was sent; and (3) the message has come from the stated source.

Digital signatures are especially important for electronic commerce and are a key component of most authentication schemes. The recipient of signed data can use a digital signature to prove to a third party that the signature was, in fact, generated by the signatory. When such proof is provided, the signed data is protected by *nonrepudiation*: the signatory cannot, at a later time, deny authorship of the signature.

digital signature algorithm

An algorithm that transforms a message of any length to a digital signature in such a way that it is computationally infeasible to (1) find two messages with the same digital signature, (2)

produce a message from a given, predetermined digital signature, or (3) find the digital signature of a given message without knowledge of the sender's private key. Typically, a digital signature algorithm is implemented by computing a message digest on the message, then encrypting the message digest with the sender's private key.

An example of a digital signature algorithm is DSA.

DII

See **Dynamic Invocation Interface (DII)**.

Direct Access Storage Device (DASD)

A disk, disk drive, or group of disks or drives on an IBM machine.

Distinguished Bulletin Board Liaison (DBBL)

A BEA Tuxedo administrative process that runs on the MASTER node of the application and communicates with BBLs to coordinate updates to the bulletin board.

Distinguished Name (DN)

(*CORBA)* A distinguished name (DN) is an entry in the Directory Information Tree (DIT) that uniquely identifies an object in an X.500 directory.

See also **object**.

distributed application

An application that is separated into two or more parts (such as a client and a server) on different computers that communicate through a network.

distributed computing

An application design and implementation strategy that separates an application into units that are executed on different computers and communicate through a network. For example, an application can be separated into three distributed units: a user interface unit, a processing unit, and a storage unit.

distributed object

(*CORBA)* An object that can live anywhere on a network. Distributed objects are packaged as independent pieces of code that can be accessed by remote clients via method invocations. The language and compiler used to create distributed objects are totally transparent to the clients. Clients do not need to know where the distributed object resides or what operating system executes on it.

Distributed Program Interface (DPI)

The Distributed Program Interface (DPI) protocol extension to SNMP agents. Permits end-users to dynamically add, delete or replace variables in the local MIB without recompiling the SNMP agent, by creating a subagent that communicates with the agent via the SNMP-DPI protocol.

distributed transaction

A transaction involving multiple transaction managers. In a distributed transaction environment, a client application may send requests to several servers resulting in resource updates at multiple resource managers. To complete the transaction, the transaction manager for each participant (client, servers, and resource managers) must be polled to coordinate the commit process for each participant within its domain.

distributed transaction processing (DTP)

A form of processing in which multiple application programs update multiple resources (such as databases) in a coordinated manner. Programs and resources can reside on one or more computers access a network.

DLL

See **dynamic link library (DLL)**.

domain

See **application**.

Domain Configuration (`DMCONFIG`) File

The file that describes the relationship between the local domain (the domain in which the `DMCONFIG` file resides) and remote domains (any other domains). There is one `DMCONFIG` file per domain. The `DMCONFIG` file contains domain information for BEA Tuxedo domains.

See also **Domains**.

domain gateway

A highly asynchronous multitasking server provided by the BEA Tuxedo system to process service requests to and from remote domains. The gateway makes access to services on a remote domain (and service requests from a remote domain) transparent to both the application programmer and the user.

domain gateway group

See **gateway group**.

Domains

A BEA Tuxedo system component that provides a framework for interoperability among multiple domains. The framework consists of the following:

- An extended client/server model that hides the heterogeneity of different computers and application programs, as well as the location of application programs.
- A centralized administration system that allows application administrators to control all cooperating machines as a single application.

Domains-level failback

The restoration of message traffic to a primary remote domain. The TDomain gateway always tries to use the primary domain or the highest-level alternate remote domain defined for a service. When these domains become unavailable (due to circuit failure or other reasons), the gateway transfers message traffic to a lower-priority alternate remote domain, and periodically checks the availability of the primary remote domain and the highest-level alternate remote domain. When possible, the gateway restores message traffic to the primary remote domain or the highest-level remote domain.

See **domain gateway** *and* **TDomain gateway**.

Domains-level failover

The transfer of message traffic to an alternate remote domain when a primary remote domain fails.

See **remote domain**.

dotted-decimal notation

A convention for expressing IP addresses, consisting of a series of four decimal numbers (0-255), separated by periods. Example: 123.205.23.99.

DPI subagent

See **Distributed Program Interface (DPI)**.

DSA

Digital Signature Algorithm. An algorithm used to generate *digital signatures*. DSA is defined in US FIPS 186.

DSI

See **Dynamic Skeleton Interface (DSI)**.

DTP

See **distributed transaction processing (DTP)**.

dual-paired connections outbound IIOP

See **outbound IIOP**.

dynamic argument

A type of argument for which a method can allocate or extend the existing storage.

Dynamic Data Exchange (DDE)

A form of communication available on Microsoft Windows platforms that allows applications to exchange information through a series of messages. Two applications sending and responding to DDE messages are said to have a DDE conversation.

dynamic datatype

A datatype whose memory size is not known when the code is compiled; a dynamic datatype's memory size is known only when the code is executed.

Dynamic Invocation Interface (DII)

(*CORBA)* An API that allows a CORBA client to either perform invocations on an object whose signature may be unknown at compile time, or a deferred synchronous invocation. If an object's signature is unknown, the client locates the object and uses the Interface Repository to obtain information about the object's signature and constructs an invocation with the proper parameters. The client can then issue the invocation and receive the response. DII is distinguished from the static invocation interface in which a client performs a synchronous invocation using client stubs. DII also allows a client to issue a request and to not block until the request is completed. The client checks for a response at a later time.

See **Dynamic Skeleton Interface (DSI)**.

dynamic link library (DLL)

A collection of functions grouped into a load module that is dynamically linked with an executable program at run time for a Microsoft Windows application.

Dynamic Skeleton Interface (DSI)

(*CORBA)* An API that provides a way to deliver requests from an ORB to an object implementation. DSI is used at compile time when the ORB has no knowledge of the object implementation. As the server-side analog to the client-side DII, DSI lets the application programmer examine the parameters of an incoming request to determine a target object and method.

See **Dynamic Invocation Interface (DII)**.

E

electronic commerce

The practice of buying and selling goods and services over the Internet.

encoding

The conversion of architecture-specific data into a format that can be transmitted between different architectures, such as XDR encoding.

encryption

The process of algorithmically scrambling data to prevent (or hinder) unauthorized disclosure, while still preserving access to the original data by authorized users. To read an encrypted file, a recipient must have access to a secret key or password that enables the recipient to decrypt it. Unencrypted data is called plaintext; encrypted data is referred to as ciphertext.

See **ciphertext** *and* **plaintext**.

encryption key pair

An encryption key pair consists of the public key used to encrypt information and a private key used to decipher the information.

environmental object

(*CORBA)* Any support object that provides independence from the underlying environment (for example, independence from the operating system). The Bootstrap object is an environmental object.

See also **Bootstrap environmental object** *and* **object**.

environment variable

A string of specific value that controls a certain attribute of an application. Environment variables are made available to the application as it begins.

event

An indication to a BEA Tuxedo system process of the occurrence of a particular state or condition, for example, disconnection, transaction request mode, connection request, and so forth.

Event Broker/Monitor

A BEA Tuxedo system component that monitors the occurrence of defined system and application events and notifies subscribers when an event is detected.

event posting

Notification by the BEA Tuxedo system (or by an application) to the Event Broker/Monitor that a defined event has occurred.

event subscription

A request made to the Event Broker/Monitor to be notified when a specific event is detected.

exception

(*CORBA)* An event that occurs during program execution that prevents the program from continuing normally (usually an error). C++ supports exceptions with the `try`, `catch`, and `throw` keywords. There are two categories of exceptions: system and user-defined.

In C++, system exceptions inherit from `CORBA::System_Exception` and user-defined exceptions inherit from `CORBA::User_Exception`.

extensibility

The ease with which a system adapts to meet new requirements. Extensibility also includes the ability to add or change a function or data (data type, file format, database schema, or information model) without:

- Requiring changes to existing functions, data, and interfaces.
- Introducing unwanted side effects, such as degradation of performance, reliability, maintainability, portability, and so on.

Exterior Gateway Protocol (EGP)

A protocol used to advertise the set of networks that can be reached within an autonomous system. EGP enables this information to be shared with other autonomous systems.

External Data Representation (XDR)

A canonical data format defined by Sun Microsystems and used to transfer data between heterogeneous hardware nodes.

F

factory

(*CORBA)*

- Any distributed CORBA object that returns an object reference to other distributed CORBA objects. A factory is located in a server application.
- An interface used by a client to obtain an object reference to a CORBA object. Object references to factories are obtained by the client using an object reference to a Factory

Finder interface. The Factory Finder interface is advertised by the system and is made available to the client as part of client bootstrap processing.

See also **CORBA object**, **object reference**, *and* **application**.

factory-based routing

(*CORBA)* A feature of the BEA Tuxedo software that permits the routing of requests on a CORBA object reference to a specific server group based on criteria supplied at the time the object reference is created by a factory.

See also **factory** *and* **object reference**.

factory finder

(*CORBA)* A CORBA object that locates the factories that an application needs. Both client applications and server applications can use a factory finder. A factory finder object provides an implementation of the CORBAservices `COSLifeCycle.FactoryFinder` interface, as well as the BEA `Tobj.FactoryFinder` interface.

See also **factory**, **local factory**, **object**, **client application**, *and* **server application**.

`factory_finder.ini` file

(*CORBA)* The FactoryFinder configuration file for domains. This file is parsed by the `TMFFNAME` service when it is started as a Master NameManager. The file contains information used by NameManagers to control the import and the export of object references for factory objects with other domains.

See also **domain**, **factory**, *and* **object reference**.

failback

Restoration of message traffic to a higher-priority circuit. The `BRIDGE` process always tries to use the highest-priority circuit defined for the node; when traffic is flowing on a lower-priority circuit, whether due to circuit failure or just non-availability, the `BRIDGE` periodically checks higher-priority circuits to find one that is usable. When a higher priority circuit becomes available again, the data flow is returned to it. This mechanism is called *failback*.

failover

(*ATMI and CORBA*) Seamless transfer of message traffic to a lower-priority circuit on the occasion of the failure of a higher-priority circuit. Some operating system and hardware bundles transparently detect a problem on one network card and replace it with another. When this replacement is done quickly, application-level TCP virtual circuits have no indication that a fault has occurred.

In the BEA Tuxedo system, data flows over the highest available priority circuit. If all network groups have the same priority, data travels over all networks simultaneously. If all circuits at the current priority fail, data is sent over the next lower priority circuit. This is called “failover.”

(*Jolt*) A failure prevention mechanism that works as follows. If the current Jolt Relay Adapter (JRAD) fails to respond to a connection request, the Jolt Relay (JRLY) is enabled to connect to another available JRAD. The Jolt client proves a list of JRLY addresses to which the JRAD attempts connection in a round-robin fashion.

field

- In a record, a specified area used for a particular category of data.
- An area within a segment that is the smallest referable unit of data.
- Any designated portion of a segment.
- A way of addressing a single item of data in a database table.
- An area of a window where data displays.

Field Manipulation Language (FML)

(*ATMI*) A set of C language functions for defining and manipulating storage structures called field buffers. Cooperating processes can send and receive data in fielded buffers.

(*Jolt*) An interface for maintaining buffers with field/value pairs; specifically, the 16-bit version of this interface.

field table

(*ATMI*) A file that consists of FML field names and their identifiers. The field table enables users to refer to fields by logical names rather than by system field identifiers.

FML

See **Field Manipulation Language (FML)**.

FML buffer

(*ATMI*) A buffer of self-describing data items accessed through the field manipulation language API.

foreign access path

A physical connection between a native BEA Tuxedo system node and a foreign node. At least one gateway server must reside on the BEA Tuxedo node.

foreign client application

(*CORBA)* A client application that is implemented on an ORB that is not a product of BEA Systems, Inc. Although the client is implemented on a Microsoft product, the ORB is provided by BEA Systems, Inc.

See **ORB**.

foreign node

A node in the network that does not have access to the configuration's bulletin board, or that cannot execute the full complement of BEA Tuxedo system software.

format independence

The ability to send data to a device without having to be concerned with the format in which the data is displayed. The same data may appear in different formats on different devices.

framework

The software environment tailored to the needs of a specific domain. Frameworks include a collection of software components that programmers use to build applications for the domain the framework addresses. Frameworks can contain specialized APIs, services, and tools, which reduce the knowledge a user or programmer needs to accomplish a specific task.

G

gateway

For the BEA Tuxedo system, any communication mechanism between different environments (for example, between native and foreign nodes). A software program that allows dissimilar systems to communicate and exchange information. A gateway normally handles communication between systems and performs all necessary protocol translation such that the end applications communicate transparently.

gateway group

A collection of processes that provide communication services to and from remote domains. The group consists of the following: `GWADM`, the gateway administrative server, and a gateway process, for example, `GWTDOMAIN`.

gateway server

A server process, resident on a native BEA Tuxedo system node, that communicates with one or more foreign machines.

General Inter-ORB Protocol (GIOP)

(*CORBA*) A standard for communication between independent CORBA Object Request Broker (ORB) implementations. GIOP was developed by the Object Management Group (OMG). GIOP is an abstract protocol that forms the basis for specific protocols that map the GIOP standard to individual transport layers. For example, IIOP maps the GIOP standard to the TCP/IP transport layer.

See also **CORBA ORB** and **IIOP**.

GIOP

See **General Inter-ORB Protocol (GIOP)**.

global transaction

The BEA Tuxedo system name for a transaction in which multiple servers or multiple resource manager interfaces are used and that is coordinated as an atomic unit of work. A global transaction may be composed of several local transactions, in each of which a single resource manager is accessed.

See also **resource manager (RM)**.

Global Transaction Identifier (GTRID)

A data structure, the value of which uniquely identifies a global transaction.

graphical user interface (GUI)

A high-level interface that uses windows and menus with graphic symbols instead of typed system commands to provide an interactive environment for a user. The BEA Administration Console, available through the World Wide Web, enables an authorized user to configure and control a BEA Tuxedo application.

group

A collection of servers or services on a machine, often associated with a resource manager. A group is an administrative unit used for booting, shutting down, and migrating servers and services.

See also **MIB group**.

GUI

See **graphical user interface (GUI)**.

H

handler

A request that originates on a remote computer. Handlers are registered in the local BEA Tuxedo bulletin board as client programs.

See also **Workstation Handler (WSH)**.

hierarchical database

A database organized in the form of a tree structure that predetermines the access paths to data stored in the database. DL/I, IMS, and SQL/DS are hierarchical database managers.

hierarchy

In a database, a tree of segments beginning with the root and proceeding downward to dependent segment types. No segment type can be dependent on more than one other segment type.

high-level language

A programming language.

host

A computer that is attached to a network and provides services other than acting as a communication switch.

host computer

The primary or controlling computer in a data communication system.

hypertext markup language (HTML)

The language used for writing pages for the World Wide Web.

I

ICF

See **Implementation Configuration File (ICF)**.

ident string

See **identification string**.

identification string

Portions of a file that get expanded by RCS and SNMP Agent utilities to contain file and identification information. If compiled, these strings are placed into object file functions, where the information is made available.

IDL

See **OMG IDL**.

idl compiler

(*CORBA*) A tool that takes an OMG IDL interface and produces C++ programming language interfaces and classes that represent the mapping from the IDL interface to the C++ programming language.

See also **OMG IDL**.

IDL interface

(*CORBA)* A declaration in OMG IDL of an interface to a CORBA object. The interface declaration contains IDL operations and attributes. The OMG IDL interface declaration is used to generate stubs and skeletons for BEA Tuxedo CORBA objects.

See also **CORBA object**, **interface**, **OMG IDL**, and **skeleton**.

IDL parameter

(*CORBA)* One or more objects the client passes to an IDL operation when it invokes the operation. Parameters may be declared as `in` (passed from client to server), `out` (passed from server to client), or `inout` (passed from client to server and then back from server to client).

idltojava compiler

(*CORBA*) A tool that takes an OMG IDL interface and produces Java programming language interfaces and classes that represent the mapping from the IDL interface to the Java programming language. The resulting files are `.java` files.

See also **OMG IDL**.

IIOP

(*CORBA*) Internet Inter-ORB Protocol. The standard protocol defined by the CORBA specification for interoperation between Object Request Brokers (ORBs), which was written by the Object Management Group (OMG). The IIOP enables two or more Object Request Brokers (ORBs) to cooperate to deliver requests to an object.

See also **CORBA ORB** *and* **object**.

IIOP Handler (ISH)

(*CORBA)* A BEA Tuxedo system process that handles all IIOP communication between a remote application and target CORBA objects.

See also **Jolt Server Handler (JSH)** *and* **Workstation Handler (WSH)**.

IIOP Listener (ISL)

(*CORBA)* A BEA Tuxedo system process that listens for incoming IIOP connections from remote applications. After a connection is established, the Listener hands off the connection to the IIOP Handler.

See also **Jolt Server Listener (JSL)** *and* **Workstation Listener (WSL)**.

IIOP Listener/Handler

(*CORBA)* The feature of the BEA Tuxedo software that enables client applications to communicate with the BEA Tuxedo domain, and the reverse. The IIOP Listener/Handler receives a request from a client application via the IIOP protocol, and then sends that request to the appropriate server application within the BEA Tuxedo domain. It also receives a request from a server application in the BEA Tuxedo domain and sends the request to a server outside the domain.

See also **IIOP**, **client application**, **domain**, *and* **server application**.

(*Jolt*) A process that receives a client request, which is sent using the IIOP, and delivers that request to the appropriate server application.

implementation code

(*CORBA)* The method code that you write that satisfies a client application's request on a specific object. The interface defines the operation and is implemented in the method.

See also **interface**, **metadata interface**, and **object**.

Implementation Configuration File (ICF)

(*CORBA)* A file that describes the implementation attributes of BEA Tuxedo C++ server applications. The ICF file is input to the IDL compiler when generating skeletons for BEA Tuxedo C++ server applications.

See also **skeleton** and **server application**.

implementation file

(*CORBA)* The file that contains, among other data, method declarations for each operation defined in your OMG IDL statements. You need to implement the method with your business logic. When you build the server application, you provide this implementation file to the BEA Tuxedo build procedure.

See also **implementation code**, **metadata interface**, **OMG IDL**, **operation**, *and* **server application**.

inactive server

A server that is not currently available to process requests.

incoming connection

A connection to the local gateway that is initiated by a domain gateway on a remote domain.

information hiding

A software design technique in which a piece of code contains only the information it needs to do its job.

infrastructure

A common underlying computing base. The infrastructure is a set of components (fundamental services) that support another higher-level set of components in a given system. The higher-level components are typically more directly associated with providing the specific function of the overall system.

initial naming context

(*CORBA)* When using CORBA objects, the `NamingContext` object returned by a call to the method `orb.resolve_initial_references("NameService")`. It is an object reference to the CosNaming Service registered with the ORB. The initial naming context can be used to create other `NamingContext` objects.

See also **naming context**.

instance

(*CORBA)* A particular realization of an abstraction or template, such as a class of objects or a computer process.

instantiate

(*CORBA)* To create an instance by defining one particular variation of an object within a class, giving it a name and locating the object in some physical place.

instrumentation

Facilities that provide access to the attributes of managed resources, to retrieve or modify values of these attributes. Access to managed resources used by agents to respond to management requests.

integration

The ability of applications to share information or to process independently by requesting services and satisfying service requests. In a well-integrated system, all of the parts have a purpose, and the parts combine effectively to achieve the purpose of the overall system.

interaction model

A description of how the clients and servers in a distributed application or application framework work with each other.

interactive

Pertaining to an application in which each entry entails a response from a system or program, as in an inquiry system or airline reservation system. An interactive system may also be conversational, implying a continuous dialog between the user and the system.

interactive interface

A system facility that controls how different users see and work with the system by means of user profiles. When signing on, the interactive interface makes available those parts of the system authorized by the profile. The interactive interface has sets of selection and data entry panels through which users communicate with the system.

interface

See **IDL interface**.

Interface Repository

(*CORBA)* An online database that contains the definitions of the interfaces that determine the CORBA contracts between client and server applications.

See **conversation** *and* **IDL interface**.

internationalization

A mechanism that allows customization of a system's text messages and data formats into an application's language and format of choice.

International Standards Organization (ISO)

An international organization whose membership includes standards and research groups from various nations. ISO establishes standards for computer network communications and many other technologies.

Internet

The world's largest network, the Internet is based on the TCP/IP protocol suite and is universally accessible.

Internet Inter-ORB Protocol (IIOP)

See **IIOP**.

Internet Protocol Address (IP address)

A numeric value that uniquely identifies a node in a TCP/IP network. IP addresses are usually expressed in *dotted decimal notation*, a series of four decimal numbers (0-255), separated by periods. Example: 123.205.23.99.

interoperability

The ability to exchange requests between entities.

Interoperable Object Reference (IOR)

(*CORBA)* The entity that associates a collection of tagged profiles with object references. An ORB must create an IOR (from an object reference) whenever an object reference is passed across ORBs.

See also **CORBA ORB** *and* **object reference**.

intranet

A set of internal company or group-specific networks protected by firewalls and connected by IP routers. An intranet appears to its users as a single network.

invocation

(*CORBA)* The process of performing a method call on a distributed object, with or without knowledge of the object's location on the network. CORBA Static invocation, which uses a client stub for the invocation and a server skeleton for the service being invoked, is used when the interface of the object is known at compile time. CORBA Dynamic invocation must be used if the interface is not known at compile time.

See also **CORBA callback object** *and* **skeleton**.

invocation access policy

(*CORBA)* The security policy that controls whether a client application may invoke a method on the target object as specified in the request.

See also **metadata interface** *and* **policy**.

IP address

See **Internet Protocol Address (IP address)**.

ISO

See **International Standards Organization (ISO)**.

J

JAR files (.jar)

(*CORBA*) Java ARchive files. A file format used for aggregating many files into one file.

See also **Java**.

Java

An object-oriented programming language developed by Sun Microsystems, Inc. A *write once, run anywhere* programming language.

JavaBeans

(*CORBA)* A specification developed by Sun Microsystems that defines how Java objects interact. An object that conforms to this specification is called a JavaBean. The JavaBean can be used by any application that understands the JavaBeans format. JavaBeans can be developed only in Java, but can run on any platform.

Java Development Kit (JDK)

A software development environment for writing applets and applications in Java.

See also **applet** and **Java**.

Javadoc

(*CORBA)* A tool from Sun Microsystems, Inc. that generates API documentation in HTML format from comments in Java source code. The *Java API Reference* document is formatted by the Javadoc tool.

See also **application programming interface (API)**.

Java Runtime Environment (JRE)

A subset of the Java Development Kit that is suitable for redistribution and sufficient for end users. The JRE consists of the Java Virtual Machine (JVM), the Java core classes, and supporting files.

JDK

See **Java Development Kit (JDK)**.

joint client/server application

An application that executes code that acts as the starter for some business actions, and also executes method code for invocations on objects.

See also **native joint client/server application**.

JoltBeans

(*ATMI)* JavaBeans components that are used in Java development environments to construct Jolt clients. JoltBeans consist of two sets of JavaBeans: JoltBeans toolkit and Jolt-aware AWT beans.

JoltBeans toolkit

(*ATMI)* A JavaBeans-compliant interface to BEA Jolt. The toolkit includes the JoltServiceBean, JoltSessionBean, and JoltUserEventBean.

Jolt Class Library

(*ATMI)* A set of Java classes that allows the user to write Java programs to access BEA Tuxedo services.

Jolt Relay (JRLY)

(*ATMI)* A standalone program that routes Jolt messages from Jolt clients to the Jolt Server Listener (JSL) or Jolt Server Handler (JSH) via the Jolt Relay Adapter (JRAD). Jolt Relay is *not* a BEA Tuxedo server or BEA Tuxedo client.

Jolt Relay Adapter (JRAD)

(*ATMI)* A BEA Tuxedo application server that does not include any BEA Tuxedo services. It requires command-line arguments in order to work with the JSL and the BEA Tuxedo system. The JRAD may or may not be located on the same BEA Tuxedo host machine and server group to which the JSL server is connected.

Jolt Repository

(*ATMI)* A subsystem in Jolt that provides primitive services and storage for the service definitions.

Jolt Server Handler (JSH)

(*ATMI)* A program that runs on a BEA Tuxedo server machine to provide a network connection point for remote clients. The JSH works with the Jolt Server Listener (JSL) to provide client connectivity with the BEA Tuxedo system.

See **IIOP Handler (ISH)** *and* **Workstation Handler (WSH)**.

Jolt Server Listener (JSL)

(*ATMI)* A program that supports clients on an IP/port combination. The JSL works with the Jolt Server Handler (JSH) to provide client connectivity to the backend of the Jolt system. The JSL is administered by the same tools used to manage any resource within a BEA Tuxedo environment.

See **IIOP Listener (ISL)** *and* **Workstation Listener (WSL)**.

Jolt WAS for Servlet

(*ATMI)* Renamed BEA Tuxedo-JSE Connectivity.

Jolt WAS for WebLogic

(*ATMI)* Renamed BEA Tuxedo-WebLogic Connectivity.

journaling

The recording of information in any journal (including the system log) for possible subsequent processing by the user. The primary purpose of journaling is to enable forward recovery of the data sets. A data set can be reconstructed by applying transactions in the journal against a previous version of the data set. Journaling can be used for any other user-defined purpose, such as auditing, accounting, or performance analysis.

JRAD

See **Jolt Relay Adapter (JRAD)**.

JRE

See **Java Runtime Environment (JRE)**.

JREPSVR

A BEA Tuxedo server that provides services to access the Jolt Repository storage. It provides support for the Jolt Runtime Environment and minimum editing and query functions.

JRLY

See **Jolt Relay (JRLY)**.

JSH

See **Jolt Server Handler (JSH)**.

JSL

See **Jolt Server Listener (JSL)**.

K

Kerberos protocol

The private key authentication protocol developed as part of Project Athena at the Massachusetts Institute of Technology (MIT).

Kerberos security

The security system that provides authentication, mutual authentication, and protection against replay and sequencing attacks.

keyword

- A symbol that identifies a parameter.
- A part of a command operand that consists of a specific character string.

L

LAN

See **Local Area Network (LAN)**.

LAN partition

The failure of a LAN connecting the machines of an application, resulting in a loss of message communication between the machines. A partitioned site is one that no longer has access to the master node.

lazy connection

A connection between a domain gateway and a remote domain that is not established until the remote domain receives a request for a remote service. A lazy connection keeps initialization overhead low for configurations involving many domains.

When a domain gateway server is booted, no connections are made to any remote domains. All remote services are assumed to be available and are advertised in the BEA Tuxedo bulletin board. When the first request for a service in a particular remote domain is made, the gateway server receives the request and tries to establish the connection. If the connection is made, the request flows to the remote domain and the connection remains active. If the connection fails, the client receives a failure message and the service remains advertised.

LDAP

See **Lightweight Directory Access Protocol**.

legacy application

(*ATMI*) An existing application, usually based on a relatively old release of the BEA Tuxedo system that must be modified or wrapped before it can be used by a BEA Tuxedo domain.

See also **domain**, **Workstation Handler (WSH)**, *and* **wrapper**.

Life Cycle Service

See **CORBAservices Life Cycle Service**.

Lightweight Directory Access Protocol

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.

See also **certificate** and **X.509**.

link-level encryption (LLE)

The encryption of messages moving over network links. Encryption is performed just before data is transmitted on the network, and decryption is performed just after data is received.

LLE operates over Workstation client, domain gateway, bridge, and administrative network links. It employs a symmetric key encryption technique (specifically, RC4), which uses the same key for encryption and decryption.

link-level failover in Domains

Link-level failover is a mechanism that ensures that an alternate network link becomes active when a primary link fails.

listener

See **Workstation Listener (WSL)**.

Listener/Handler

See **IIOP Handler (ISH)**, **IIOP Listener (ISL)**, **Jolt Server Handler (JSH)**, **Jolt Server Listener (JSL)**, **Workstation Handler (WSH)**, *and* **Workstation Listener (WSL)**.

LMID

See **logical machine ID (LMID)**.

load balancing

The practice of distributing service requests among all the servers in a given domain to achieve the most efficient handling of those requests. Specifically, the system identifies the server currently doing the smallest amount of work and sends requests to that server's queue for processing.

When a service request is routed to a domain gateway, the gateway implements two algorithms, a load-balancing algorithm and a data-dependent routing algorithm, to find the proper remote domains to which the request should be sent. Load-balancing and data-dependent routing

algorithms are based on the remote service table entries and remote domain table entries in the gateway-shared memory.

local

In data communication, pertaining to devices that are accessed directly (that is, without use of a telecommunication line).

local application names

The DDE Listener uses the application name supplied in the DDE Initiate message to determine if the client is looking for a local or a remote DDE application. The syntax for a local DDE application includes only the name of the application. For example, if the client is looking for Microsoft Excel on the local computer, the application name would be EXCEL.

Local Area Network (LAN)

A high-speed network that spans a limited distance, such as a building or a cluster of buildings. LANs can be connected to wide area networks (WANs) with bridge devices.

local domain

View of an application (that is, a subset of the application's services) that is available to other domains. Because a *local domain* is always represented by a *domain gateway group*, the terms are used interchangeably.

local factory

(*CORBA)* A factory object that exists in the local domain that is made available to remote domains through a BEA Tuxedo factory finder.

See also **factory**, **factory finder**, *and* **remote factory**.

local gateway

A specific gateway group (for example, `GWADM` and `GWTDOMAIN`) within a local BEA Tuxedo application. Multiple local gateways may be running within a single BEA Tuxedo application.

local node

The computer that is connected to a user's workstation.

local service

A service of a local domain that is available to remote domains through a domain gateway group.

local system

In a multisystem environment, the system on which an application program is being executed. A local application can process data from databases located on either the same (local) system or another (remote) system.

local transaction

A local resource manager transaction that is active on behalf of a global transaction.

See also **ACID properties**, **CORBA ORB,** *and* **global transaction**.

locality-constrained object

(*CORBA)* A CORBA object that cannot be invoked outside the address space in which it exists. Any attempt to pass a reference outside the address space of such an object, or any attempt to externalize an object supporting the interface using `CORBA::ORB::object_to_string`, results in the `CORBA::MARSHAL` system exception being raised.

location transparency

The ability to define a resource so that its name implies no specific network address or physical location.

log file

A message file that describes events that occur during an operation. Log files are updated frequently during an operation and are useful for tracing system operations and errors.

logical machine ID (LMID)

The logical name assigned (in the configuration file) to a processing element used in a transaction manager application.

M

makefile

A file, referenced by the make command, that tells the make command how to create each of the files needed to generate a complete program. The makefile contains a list of source files, object files, and dependency information.

managed object

A software entity, defined within the Management Information Base (MIB), that represents a feature of a managed resource (such as a process, a piece of hardware, or system performance attribute) and is controlled through a management infrastructure, such as the BEA Tuxedo TMIB, on behalf of a management console.

See also **Management Information Base (MIB)** *and* **managed resource**.

managed resource

The physical resource whose attributes are represented by *managed objects* in a *management information base.* A managed resource can be a software entity such as an application or queue, or a hardware device, such as an interface card or hub.

management framework

A system that provides a unified view of hardware and software resources on distributed systems and enterprise-wide networks to help network or system administrators to manage and control these resources.

Management Information Base (MIB)

(*ATMI* and *CORBA*) A BEA Tuxedo system component that provides a complete definition of the classes and their attributes that make up the BEA Tuxedo system. The BEA Tuxedo System Management Information Base comprises a generic MIB and a specific MIB for each major component, such as Domains and Workstation. Configuration and administration of the BEA Tuxedo system can be done programmatically by using the ATMI to set or change the value of an attribute.

See also **Tuxedo MIB**.

management station

The machine on which the SNMP manager application runs.

man-in-the-middle attack

An attack where an enemy inserts a machine into a network, and then captures, possibly modifies, and retransmitts all messages between two parties.

mapping

(*ATMI*) The process of associating local values or entities with values or entities that are meaningful on remote systems.

(*CORBA*) In CORBA, the relationship between OMG IDL statements and the programming language code that results when the OMG IDL statements are compiled. For example, a C++ IDL compiler maps OMG IDL statements into C++ language bindings.

See **OMG IDL**.

marshal

The process of packing data into a stream of bytes so that the data can be shipped across a network to another computer.

mask

An SNMP means of hiding selected SNMP traps, so that alarms are generated only for specified instances.

master agent

The single point of contact for the SNMP manager on a managed node. The master agent receives requests from the SNMP manager and contacts the appropriate subagents to fulfill the requests.

master node

The `MASTER` node for an application as designated in the `RESOURCES` section of the configuration file. It contains the master copy of the `TUXCONFIG` binary configuration file. Administration of the running system is done from the `MASTER` node.

MD5

Message Digest 5. An algorithm defined in RFC 1321 that takes as input a message of arbitrary length and produces as output a 128-bit *message digest*, or *hash value*, of the input. The MD5 algorithm is intended for digital signature applications, where a large file must be compressed in a secure manner before being encrypted with a private key under a public-key cryptosystem such as PKCS.

message

A formula for sending data and values across applications. Messages may contain statistical or status information about application processes or instructions for the recipient. They consist of a header, containing message ID data, and a body containing user-defined information.

See also **request**.

message catalog

With respect to internationalization, a file or storage area containing program messages, command prompts, and responses to prompts for a particular native language, territory, and codeset.

message definition block

The total body of data that makes up a message definition. It includes, for example, the command name, the subsystem name, and the internal and external recommendations.

message digest

The representation of text in the form of a single string of digits, created with a formula called a *one-way hash function*. Encrypting a message digest with a private key creates a digital signature, which is an electronic means of authentication.

message digest algorithm

A method of reducing a message of any length to a string of a fixed length, called the *message digest* or *hash value*. Message digest algorithms have the property that it is computationally infeasible to find a message that corresponds to a given message digest, or to find two different messages that produce the same message digest. Examples of message digest algorithms are MD5 and SHA-1.

metadata interface

(*CORBA)* An interface that accesses data about data; descriptive information about a particular object.

method

(*CORBA)* In object-oriented programming, a programmed procedure that is defined as part of a class and included in any object of that class. A class (and thus an object) can have more than one method. A method in an object can have access only to the data known to that object, which ensures data integrity among the set of objects in an application. A method can be reused in multiple objects.

See **Callbacks Wrapper object** *and* **operation**.

MIB

See **Management Information Base (MIB)**.

MIB group

Ancestor object of MIB objects within the OID (or registration) tree. A MIB group may contain other MIB groups, or it may contain scalar or tabular objects.

middleware

A set of services for building distributed client/server applications, such as services for locating other programs in the network, establishing communication with those programs, and passing information between applications. Middleware services can also be used to resolve disparities between different computing platforms and to provide a uniform authorization model in networks that comprise multiple vendors and multiple operating systems.

migrate

Relocates a server or group of servers from one LMID to another. Migration must be planned for and specified in the configuration file.

model

A paradigm in which details have been abstracted for the sake of simplification.

modeling

A design technique used in developing architecture, simulations, and computer systems.

module

A piece of code that contains information on a particular topic and on the topic's related interfaces. For example, code that describes a banking withdrawal operation could be stored in a module.

MOPS

Management operations per second.

MP model

A BEA Tuxedo application configuration that runs on more than one computer. The multiple machines that make up an MP configuration might include two or more uniprocessors, one or more multiprocessors, or a combination of uniprocessors and multiprocessors.

multicontexted process

A BEA Tuxedo process that is connected to more than one application and/or has more than one connection to the same application.

multidomain client

A BEA Tuxedo client that is associated with more than one BEA Tuxedo applications.

multiple listening addresses

When addresses are available on separate networks, even if one virtual circuit is disrupted, another circuit can continue undisturbed. Only a failure on all configured networks makes reconnection of the `BRIDGE` processes impossible. For example, when a high-priority network fails, its load can be switched to an alternate network with a lower priority. When the higher-priority network returns to service, the network load returns to it.

Multiple Virtual Storage (MVS)

One of IBM's principal mainframe operating systems. MVS/XA: Extended Architecture. MVS/ESA: Enterprise Systems Architecture.

multiprocessor

A computer that has more than one processing element, each with its own private memory.

multiprogramming

The concurrent execution of two or more computer programs.

multithreading

Use of a process by several transactions.

multithreaded CORBA server application

(*CORBA*) An application design to use multiple, independent threads. In general, this provides concurrency within an application and can improve overall throughput. Using multiple threads enables applications to be structured efficiently with threads servicing several independent tasks in parallel.

See also **CORBA server application**, **thread**, *and* **threading**

mutual authentication

The process whereby each side of an intended communication proves its identity to the other. Frequently, this is a prerequisite for the establishment of a secure association between a client and a target. Mutual authentication ensures that both parties can perform a secure transaction.

See also **authentication**.

MVS

See **Multiple Virtual Storage (MVS)**.

N

name binding

(*CORBA)* The association of a name with an object reference. Name bindings are stored in a naming context.

name resolution

(*CORBA)* The process of converting a name to an object reference.

name server

(*CORBA)* A software component of the BEA Tuxedo system that transparently maps service names to physical addresses so that users can communicate with services by name rather than by internal identifier.

namespace

(*CORBA)* A collection of naming contexts that are grouped together.

naming context

(*CORBA)* An object that contains a set of name associations in which each name is unique.

See **CORBAservices Naming Service**.

Naming Service

See **CORBAservices Naming Service**.

native client

See **client**.

native client application

(*CORBA*) A client application that invokes operations defined in OMG IDL statements to talk to CORBA server applications. Relative to the BEA Tuxedo domain to which the server applications belong, a client application is either native (that is, local) or remote. Remote and native client applications are the same. Their requests are handled differently and transparently, depending on whether or not the applications are co-located on a machine that is running in the BEA Tuxedo domain. Native client applications are always co-located on a machine in the BEA Tuxedo domain.

See **OMG IDL**, **foreign client application**, *and* **remote client application**.

native joint client/server application

(*CORBA*) A joint client/server application that is located within a BEA Tuxedo domain. C++ native joint client/server applications are built with the `buildobjclient` command. The BEA Tuxedo software does not support Java native joint client/server applications.

See **joint client/server application**.

native node

A machine in a BEA Tuxedo configuration that contains the full complement of BEA Tuxedo software, and that has access to the same bulletin board as all other native nodes in the configuration (that is, it is part of the administrative domain of the application).

network

- An interconnected group of nodes.
- The assembly of equipment through which connections are made between data stations.
- The communication path used to communicate with a server.

Network Agent (NA)

BEA TOP END term that is equivalent to the BEA Tuxedo term **Workstation Handler (WSH)**.

Network Interface (NI)

BEA TOP END term that is equivalent to the BEA Tuxedo term **Bootstrap environmental object**.

Node Manager (NM)

BEA TOP END term that is equivalent to the BEA Tuxedo term **Bulletin Board Liaison (BBL)**.

network provider

The protocol used at the transport level and below to communicate data across a network. Network providers are typically accessed from programs through a transport interface. Examples of network providers are TCP/IP and StarLAN.

NLS

The UNIX system network listener service.

node

A point on a network. The term is also used to refer to a computer (for example, a single instantiation of the UNIX operating system) that participates in a BEA Tuxedo system application. It is possible, however, to have more than one node in a multiprocessor system.

nonblocking mode

An asynchronous style of message delivery, that does not require a program to wait for an action to complete before proceeding.

nonmaster node

Any node of a BEA Tuxedo application that is not designated the `MASTER` node.

nonpartitioned

A term used to identify portions of a partitioned network that continue to be able to communicate with the DBBL on the master node.

***n*-tier client/server**

An application development approach that distributes application logic across three or more environments: the desktop computer, one or more application servers, and a database server. The main advantage of the *n*-tier client/server model is that it extends the benefits of client/server architecture to the enterprise level. Other advantages include added manageability, scalability, security, and higher performance.

O

object

(*CORBA)* An entity defined by its state, behavior, and identity. These attributes (also known as properties) are defined by the object's object system.

See **CORBA object** *and* **Remote Method Invocation (RMI)**.

object activation

(*CORBA*) The process of making a CORBA object ready to accept invocations from client applications. The object needs to have its methods and state available in memory.

When using CORBA objects, the association of an object ID to a servant in the Active Object Map of a POA and the TP Framework. When a CORBA object is activated, the TP Framework reloads the state of the CORBA object from persistent storage and makes the object available to fulfill requests from client applications.

See **active object map**, **metadata interface**, **object deactivation**, **object ID (OID)**, **object reference**, **Portable Object Adapter (POA)**, *and* **servant**.

Object Bridge

(*CORBA*) Software from Visual Edge Software, Ltd. that provides a framework for object system interoperability.

object by value

A method of passing an object by value as opposed to by reference. A description of the state of an object is sent to a receiver. The receiver of an object passed by value instantiates a new instance of that state with a separate identity from that of the sender. Once the object is passed by value, no relationship is assumed to exist between the two instances.

object deactivation

(*CORBA*) In CORBA, the removal of the association of an object ID to a servant in the Active Object Map of a POA and the TP Framework. The result of object deactivation is that no client invocation on an object reference that contains this object ID can be satisfied without first performing object activation.

In EJB, the removal of the association of an object ID to an instance in the Active Object map by the EJB container. The result of object deactivation is that no client invocation on an object reference that contains this object ID can be satisfied without first performing object activation.

See **active object map**, **object activation**, **object ID (OID)**, **object reference**, **passivation**, *and* **Portable Object Adapter (POA)**.

object handle

(*CORBA*) Identifies the object in a portable way. The handle can be serialized, which allows you to store the object handle and then use it at a later time, possibly in a different process or in a different system, or by another bean or object.

object ID (OID)

See **object identifier (OID)**.

object identifier (OID)

(*CORBA)* A unique number assigned to each object in the MIB. These OIDs fall into specific categories and form a tree. When the agent accesses a specific object, it traverses the OID tree in the MIB file to find the object. An OID identifies an object by specifying a unique path to the object from the root of the OID tree.

object implementation

(*CORBA)* The code you write that implements the operations defined for an interface.

See also **interface**.

object interface

(*CORBA)* The interface to an object, as defined in an application's OMG IDL statements. The object interface identifies the set of operations and attributes that can be performed on an object. For example, the interface for a teller object identifies the types of operations that can be performed on that object, such as withdrawals, transfers, and deposits.
`Tobj::TransactionCurrent` is an example of an object interface provided by the BEA Tuxedo software.

See **CORBA object**, **OMG IDL**, *and* **operation**.

Object Management Group (OMG)

(*CORBA)* An international organization that establishes industry guidelines and object management specifications to provide a common framework for object-oriented application development. The OMG Common Object Request Broker Architecture specifies the CORBA object model.

object model

(*CORBA)* The model that reflects as objects the overall design of an application or system.

object reference

(*CORBA)* An identifier that uniquely specifies an instance of an object within a distributed ORB system.

Object Request Broker

See **CORBA ORB**.

object system

(*CORBA)* A software system that stores, manipulates, and uses a collection of objects according to a set of system-specific standards. An object system specifies how information is exchanged

between objects, and how objects are implemented in accordance to an object model, such as CORBA COM, EJB, and RMI.

See **Component Object Model (COM)**, **conversation**, *and* **object model**.

Object Transaction Service

See **CORBAservices Object Transaction Service (OTS)**.

octet

- A byte that consists of eight bits.
- A byte composed of eight binary elements.

OID

See **object identifier (OID)**.

OLE

(*CORBA)* Object Linking and Embedding.

OLTP

See **online transaction processing (OLTP)**.

OMG IDL

(*CORBA*) Object Management Group Interface Definition Language. A definition language specified by the OMG for describing an object's interface (that is, the characteristics and behavior of an object, including the operations that can be performed on the object).

See also **operation**.

online transaction processing (OLTP)

- A form of data processing in which users at terminals or workstations send messages to application programs, which update databases in real time.
- The execution of units of work in a performance-critical environment that appears to the user as immediate; real-time; usually having internal recoverability, history-keeping, and consistency-assurance features.

open framework

A software infrastructure in which developers can easily remove and replace software components in a *plug-and-play* fashion.

open system

A system that implements specified common standards across different computer vendors. Implementing open system standards for communication allows computers from different vendors to communicate with each other.

Open Systems Interconnect Commitment, Concurrency, and Recovery (OSI CCR)

- This protocol is an ISO standard (ISOIIEC 9804) for services and protocols that are used to commit or roll back global transaction branches.
- A software implementation of the ISOIIEC 9804 standard.

Open Systems Interconnection (OSI)

A consortium that facilitates communication among different types of computer systems.

Open Systems Interconnect Transaction Processing (OSI TP)

- An ISO standard (ISOIIEC 10026-2) for services and protocols that are used to establish dialogs and pass messages between clients and service routines on different computers.
- A software implementation of the ISOIIEC 10026-2 standard.

operation

(*CORBA)* An action that can be performed by an object. For example, you can request several operations on a file object, including opening, closing, reading, and printing.

See also **object**.

ORB

See **CORBA ORB**.

ORBMain module

(*CORBA)* The main procedure of the BEA Tuxedo server application process. The BEA Tuxedo software provides the ORBMain module. You do not modify this module. The server application build procedure automatically builds the ORBMain module into the server application process. The ORBMain module is provided by the `buildobjserver` command for servers using the TP Framework. Note that joint client/server applications must provide their own main procedure and must use the `-P` switch on the `buildobjclient` command.

OSI

See **Open Systems Interconnection (OSI)**.

OSI CCR

See **Open Systems Interconnect Commitment, Concurrency, and Recovery (OSI CCR)**.

OSI TP

See **Open Systems Interconnect Transaction Processing (OSI TP)**.

OTS

See **CORBAservices Object Transaction Service (OTS)**.

outbound IIOP

(CORBA) A feature of the BEA Tuxedo software that supports client callbacks. Outbound IIOP adds the outbound half-gateway to the ISL/ISH.

The BEA Tuxedo system supports the following three types of outbound IIOP:

1. asymmetric outbound IIOP

Outbound IIOP, via a second connection, to joint client/server applications that are not connected to an ISH. This feature of the BEA Tuxedo software is supported for GIOP 1.0, GIOP 1.1, and GIOP 1.2 client applications, server applications, and joint client/server applications.

2. bidirectional outbound IIOP

Outbound IIOP to a remote joint client/server application that is connected to an ISH. The outbound callback reuses the same connection initially used by the joint client/server for inbound calls. This feature is supported only for BEA Tuxedo C++ GIOP 1.2 client applications, server applications, and joint client/server applications.

3. dual-paired connections outbound IIOP

Outbound IIOP to a remote joint client/server application that is connected to an ISH. Unlike bidirectional outbound IIOP, the outbound callback uses a second connection that is separate from the connection initially used by the joint client/server application for inbound calls. This feature of the BEA Tuxedo software is supported for GIOP 1.0, GIOP 1.1, and GIOP 1.2 client applications, server applications, and joint client/server applications.

outgoing connection

A connection from the local gateway that has been generated as a result of one of the following: an automatic retry of the connection, an initial request to a remote domain, or a `dmadmin(1) connect` command sequence issued by the administrator.

out-of-band data

Data delivered by the BEA Tuxedo system outside the normal client/server communications channels supported by the BEA Tuxedo system.

P

parallel data circuits

Parallel data circuits enable data to flow simultaneously on more than one circuit. When you configure parallel data circuits, network traffic is scheduled over the circuit with the largest network group number (NETGRPNO). When this circuit is busy, the traffic is scheduled automatically over the circuit with the next lower network group number. When all circuits are busy, data is queued until a circuit is available.

partition

A state in which one or more active nodes of a networked application are unable to contact other active nodes because of a problem such as a LAN failure.

pass phrase

A string of alphanumeric and other characters, usually provided by a human being to provide identity. A pass phrase is typically longer than a password. It should contain more than one word, with mixed uppercase and lowercase letters, plus punctuation characters. A pass phrase should be easy to remember, but harder for an intruder to guess than a single password.

passivation

The deactivation of a bean's state. In the case of a stateful session bean or an entity bean, passivation typically involves writing the bean's state data to durable storage. This state can be restored at a later time during reactivation. For stateful session and entity beans, passivation causes object deactivation.

See also **object deactivation**.

persistent object

An object that exists independently of the process within which its object reference was created.

See **object reference** *and* **transient object**.

PID

See **process ID (PID)**.

PIDL (Pseudo-IDL)

(CORBA) The interface definition language for describing a CORBA pseudo-object. Each language mapping, including the mapping from IDL to the C++ or Java programming language, describes how pseudo-objects are mapped to language-specific constructs. PIDL mappings may or may not follow the rules that apply to mapping regular CORBA objects.

PKCS-7

See **Public-Key Cryptography Standard 7 (PKCS-7)**.

plaintext

In cryptography, text that is not encrypted.

platform

The combination of hardware, operating system, and windowing system software that supports an application.

POA

See **Portable Object Adapter (POA)**.

policy

See **activation policy**, **SecurityCurrent**, *and* **transaction policy**.

polling

(*CORBA)* An activity in which a manager interrogates an agent at periodic intervals, checking to determine whether a managed object value has crossed a specified threshold. The agent reports the values of specified managed objects.

portability

The ease with which developers can move an application from one platform to another without costly reengineering.

Portable Object Adapter (POA)

(*CORBA)* A run-time library of functions that are built into the CORBA server application executable image. The POA creates and manages object references to all objects used by the application. In addition, the POA manages object state and provides the infrastructure for the support of persistent objects and the portability of object implementations between different ORB products.

The BEA Tuxedo server application procedure automatically builds the POA into the server application. The BEA Tuxedo TP Framework automatically handles all the BEA Tuxedo server application interactions with the POA. Note that joint client/server applications interact directly with the POA.

***See* CORBA object, object reference, state**, *and* **WebLogic Express**.

port number

The entity on a TCP/IP host that identifies a logical communications channel and distinguishes one connection from another. A TCP/IP server "listens" for incoming connection requests at a designated port. A TCP/IP client initiates a connection with the server by specifying the host's IP address and the server's designated port number.

pragma

(*CORBA)* A directive to an IDL compiler to perform specific operations when compiling an IDL file. For example, the pragma Prefix affects the Interface Repository ID for an IDL interface.

precommit

The process (used by resource managers to make data recoverable) of copying data changes to stable storage.

primary remote domain

The remote domain with the highest priority. It is used whenever it is available.

principal

(ATMI) For security purposes, an authenticated user.

(CORBA) A user or programmatic entity with the ability to use the resources of a system.

Principal Authenticator object

(*CORBA)* The object that is visible to the application responsible for the creation of Credentials for a given principal. A user or principal that requires authentication, but has not been authenticated, uses the Principal Authenticator object.

See also **authentication** *and* **object**.

private key

An encryption/decryption key known only to the party or parties that exchange secure messages.

private MIB

A MIB that is defined under a private MIB directory.

procedure

A sequence of instructions (to a computer) for performing a particular task.

process ID (PID)

A unique number that identifies a process.

Process-Entity design pattern

The design pattern that can be used to increase performance in situations where a client application needs to interact with database records stored on a server machine.

See also **Client Data Caching design pattern** *and* **design pattern**.

profile

A set of information about a client or a user. The profile provides information the server might require to recognize the client or the user.

protocol

- A set of rules that govern the format and timing of messages sent and received over a communication link. TCP/IP is an example of a network protocol.
- The set of *rules* followed by two systems to communicate and exchange information.

provider

The communications product supplying networking facilities through level 4 of the OSI communications protocol.

Pseudo-IDL

See **PIDL (Pseudo-IDL)**.

pseudo-object

(*CORBA)* An object similar to a CORBA object in that it is described in IDL however, unlike a CORBA object, it cannot be passed using its object reference, nor can it be narrowed or stringified.

The DII interfaces are examples of pseudo-objects, although the DII interfaces are implemented as libraries and are more accurately described in OMG specifications as pseudo-objects with IDL interfaces. The IDL for pseudo-objects is called PIDL, indicating that a pseudo-object is being defined.

See also **PIDL (Pseudo-IDL)**.

public key

A value provided by a certificate authority that, combined with a private key, can be used to encrypt and decrypt messages.

public key algorithm

An algorithm for encrypting or decrypting data with a public or private key. A private key is typically used to encrypt a message digest; in such an application, the public key algorithm is

called a *message digest encryption algorithm*. A public key is typically used to encrypt a content-encryption key, or session key; in such an application, the public key algorithm is called a *key-encryption algorithm*. An example of a public key algorithm is RSA.

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.

public key encryption

A technique that uses a pair of asymmetric keys for encryption and decryption. Each pair of keys consists of a public key and a private key. The public key is made public by distributing it widely. The private key is never distributed; it is always kept secret.

public key security

BEA Tuxedo security built on the public key encryption technology. Uses public key encryption to establish end-to-end digital signing and data privacy between BEA Tuxedo application clients and servers. Complies with the PKCS-7 standard.

publish

The act of pushing a structured event into the event channel to make the event available to subscribers.

Q

queue

A simple data structure for managing the time-staged delivery of requests to servers. Queued elements may be sorted in some order of priority. Clients insert items in the queue and servers remove items from the queue, as soon as possible, in batch, or periodically.

R

RC2

Rivest's Cipher 2. A variable key-size block cipher with a key size range of 40 to 128 bits. It is faster than DES and is exportable with a key size of 40 bits. A 56-bit key size is allowed for foreign subsidiaries and overseas offices of United States companies. In the United States, RC2 can be used with keys of virtually unlimited length, although BEA Tuxedo public key security restricts the key length to 128 bits.

RC4

Rivest's Cipher 4. A variable key-size stream cipher with byte-oriented operations. RC4 is a symmetric or private key system that is about 10 times faster than DES and is exportable with a key size of 56 bits. In the United States, RC4 can be used with keys of virtually unlimited length, although BEA Tuxedo link-level encryption restricts the key length to 128 bits.

ReceivedCredentials object

(*CORBA)* The object that represents the secure association to the application. The ReceivedCredentials object contains the properties of that association.

See also **object**.

RECONNECT client

A Jolt client whose network connection can be torn down after being idle for a specific amount of time, but whose user context in BEA Tuxedo remains active.

record

Input or output data as it exists outside the local or remote BEA Tuxedo regions, or on different kinds of systems.

recover

A request from a coordinator or a participant to complete an identified transaction.

recovery

In transaction systems, after a failure, the ability to restore the system to the most recently committed, and therefore consistent, state. In distributed systems, recovery may involve resynchronizing several distributed components. Once a system has been recovered, processing can resume, and transactions aborted as a result of the failure can be resubmitted.

relational database

A database accessed and organized according to the relationships between data items. Relationships are expressed by means of tables that allow the accessing of items with matching attributes. The access path is determined at the time of access.

relative OID

(*CORBA)* An *object identifier* (OID) that specifies a path to a managed object only relative to some node in the OID tree below root.

reliability

The extent to which a system (or part of a system) produces the correct output on repeated trials (without unintended side effects), while meeting the performance specification.

remote

Describes a service or a computer that is available to a client over the network.

remote client

See **client**.

remote client application

See **CORBA remote client application**.

remote domain

(*ATMI)* View of an application that is accessed through a local domain gateway group.

remote factory

(*CORBA)* A factory object that exists in a remote domain that is made available to the application through a BEA Tuxedo factory finder.

See also **factory**, **factory finder**, **local factory**, *and* **CORBA domain**.

remote file sharing (RFS)

A UNIX system capability that provides access to remote files over a network.

remote gateway

The functionality of a specific gateway group within a remote BEA Tuxedo application.

remote joint client/server applications

(*CORBA)* A CORBA joint client/server application that is located outside a BEA Tuxedo domain. The remote joint client/server application does not use the BEA Tuxedo TP Framework and requires more direct interaction between the client application and the ORB. Remote joint client/server applications are built with the `buildobjclient` command or with the Java client application commands.

See **CORBA ORB**, **client application**, *and* **WebLogic Express**.

Remote Method Invocation (RMI)

(*CORBA)* A Java-specific API for accessing remote objects.

remote node

Any computer in the network other than the computer to which the user's workstation is connected.

remote procedure call (RPC)

A local procedure call that is executed in a non-local program or address space. Enables application logic to be split between a client and a server in the way that best uses available resources.

remote service

A service of a remote domain that is made available to the local application through a domain gateway group.

remote service fan-out

A configuration in which a remote service that is imported from multiple remote domains is advertised only once by a local domain gateway in the BEA Tuxedo bulletin board. A remote service can be fanned-out by a gateway server that imports the same service name from multiple remote domains. Fan-out is achieved through the gateway shared memory.

remote service name

The name (between 1 and 16 characters long) of a service offered by a remote system that can be accessed through a remote gateway.

request

A message sent by a client that identifies an operation to be performed. In a CORBA environment, the message is sent to the Object Request Broker and is relayed to the appropriate server application, which fulfills the request.

See also **client**, **server**, *and* **CORBA ORB**.

requester

- A process that receives messages from clients, converts these messages to a common internal form, determines the appropriate server or servers for the transaction request, and forwards the request to a server.
- A generic term for a client or server that is acting as a client.

request-level interceptor

(*CORBA)* A user-written application that is inserted into the invocation path between the client and server components of a BEA Tuxedo application, and that is invoked automatically by the ORB on each object invocation. A request-level interceptor allows services such as security or monitoring components to be added to an object invocation, either at the client or the server end. Request-level interceptors facilitate the use of third-party security plug-in software.

requestor

A process that receives messages from clients, converts these messages to a common internal form, determines the appropriate server or servers for the transaction request, and forwards the request to a server.

request/response communication

Communication characterized by a single request matched to a single response message. When this type of communication is used, a client requests a task, and a server performs the task and sends a response to the client. Request/response communication may be conducted synchronously or asynchronously.

request/response server

A server that offers request/response services.

request/response service

A service initiated by a request from a client. The service routine receives a single request and provides (at most) a single reply. A request/response service is handled like a procedure and has the following properties: it is executed until completion, it does not have any dialog with the requestor, and it sends back a return-value to the requester. For a requester, the execution of a request/response service can be synchronous or asynchronous.

Requests For Comments (RFC)

Documents in which Internet standards, as approved by the Internet Architecture Board (IAB), are published.

resource manager (RM)

An interface and associated software that provides access to a collection of information and processes; for example, a database management system. Resource managers provide transaction capabilities and permanence of actions; they are the entities accessed and controlled within a global transaction.

See also **transaction manager (TM)**.

resource manager instance

A particular instance or occurrence of a resource manager. There may be many occurrences or instances of the same or different resource managers within a global transaction, each managing different data. Each resource manager instance is considered to be autonomous, in full control of local access (for both local and global transactions), administration, and so forth.

response time

The elapsed time between entering an inquiry or request and receiving a response.

RFC

See **Requests For Comments (RFC)**.

RM

See **resource manager (RM)**.

roll back

To terminate a transaction in such a way that all resources updated within a transaction revert to their original (that is, pretransactional) state.

rollback

The event that ends a transaction and nullifies or undoes all changes to resources that were specified during that transaction.

RPC

See **remote procedure call (RPC)**.

RSA

A public key algorithm invented by Rivest, Shamir, and Adleman. RSA is widely employed for digital signatures and encryption. The RSA algorithm is asymmetric: a private key is closely held by its owner, while a corresponding public key may be widely disseminated.

RTQ

Component of the BEA TOP END system that is equivalent to the /Q component of the BEA Tuxedo system.

run-time trace

A BEA Tuxedo feature that enables users to monitor application-to-application transactions and, if necessary, troubleshoot distributed applications under development or during production. Also, it allows users to pinpoint problems to any hardware, operating system, network, or application code.

S

scalability

The extent to which developers can apply a solution to problems of different sizes. Ideally, a solution should work well across the entire range of complexity. In practice, however, there are usually simpler solutions for problems of lower complexity.

scalar object

A MIB *leaf* object—that is, a MIB object that does not contain other MIB objects below it in the OID tree—that can have only one instance.

SCM

See **Service Control Manager (SCM)**.

scope

(*ATMI*) To use a class to enforce a particular use for an application.

Secure Sockets Layer (SSL)

A transport-level technology developed by Netscape for authentication and data encryption between two communicating applications, based on the use of public key technology.

See also **authentication**.

security

The protection of information from unauthorized modification or disclosure and the protection of resources from unauthorized use.

SecurityCurrent

(*CORBA)* The object that provides access to the security features of the system.

See also **object**.

security policy

(*CORBA)* A set of security rules for an application that are defined and enforced by a security administrator. A security policy has a collection of users (or principals) and uses a well-defined authentication protocol for authenticating users. In addition, groups may be used to simplify the setting of security rules.

When using EJBs, a security policy defines the set of permissions in Java that determine which operations in a JVM are accessible.

security principal

An entity that is known to, and can be authenticated by, the security system.

Security Service

See **CORBAservices Security Service**.

Security Service Provider Interface (SSPI)

(*CORBA*) An interface that allows security components provided by multiple vendors to be integrated with the BEA Tuxedo Security Service.

servant

(*CORBA*) The instance of the class that implements the interface defined in an application's OMG IDL statements. A servant contains the method code that implements the operations of one or more CORBA objects.

See **CORBA object**, **implementation code**, **instance**, **metadata interface**, **OMG IDL**, *and* **operation**.

servant factory

(*CORBA*) A feature of BEA Tuxedo Java server applications for automatically instantiating servants. Unlike BEA Tuxedo C++ servers, Java servers do not need to provide a callback for instantiating servants.

servant pooling

(*CORBA*) A feature of the BEA Tuxedo (C++) software that gives your BEA Tuxedo server application the opportunity to keep a servant in memory after the servant's association with a specific object ID has been broken.

See **object identifier (OID)** *and* **servant**.

server

A software program that provides a service in a client/server architecture by performing the following steps:

1. Receives a request from a client (or another serving playing the role of a client) for a particular service.

2. Dispatches a service routine to fulfill the request.

3. Sends a response to the original requestor, indicating whether the requested service has been performed successfully and conveying the results of the service that has been performed.

See also **client**.

server abstraction

Applications combine their service routines with the BEA Tuxedo system `main()` in building a server process. The BEA Tuxedo system's `main()` provides server initialization and termination, and receives incoming requests and dispatches them to service routines. All of this processing is transparent to applications.

server application

(*CORBA)* A program, written for use with the BEA Tuxedo software, that performs a task requested of it by a client application.

See **local factory**.

Server Description File

(*CORBA)* The file within which you assign the default CORBA activation and transaction policies for the interfaces implemented in your Java server application. This XML file also contains a server declaration, which includes the name of the server implementation class and the name of the server descriptor file. You can also identify the Java class files that comprise the server application's Java ARchive (`.jar`) file.

See **JAR files (.jar)**.

server group

See **group**.

server ID

An identifier for a single server. No two servers can operate at the same time with the same server ID.

Server object

(*CORBA)* The object that performs server application initialization functions, creates one or more servants, and performs server application shutdown and cleanup procedures.

See **servant**.

server-to-server communication

(*CORBA)* The feature of the BEA Tuxedo software that allows applications to invoke distributed objects and handle invocations from those distributed objects (referred to as callbacks). The CORBA or RMI objects can be either inside or outside a BEA Tuxedo domain.

service

- An application routine available for request by a client in the system.
- A module of application code that carries out a service request.

service code

The name associated with a service offered remotely through a tlisten process.

Service Control Manager (SCM)

A Windows 2000 control panel applet that provides an interface for the interactive user to control Windows 2000 services.

service request

A request initiated by a requester process that asks for the invocation of a service.

service routine

An application module that performs one or more specific services on behalf of clients. The structure of service routines (the mechanism by which they are called and terminated) is defined by the XATMI interface specification.

SERVICES section

The section of the configuration file in which services are defined.

servlet

A Java replacement for CGI. Servlets are Java classes that are invoked by a Web server in response to HTTP requests. Servlets generate Hypertext Markup Language (HTML) as their output.

(*Jolt*) An applet that runs on a server. This term usually refers to a Java applet that runs in a Web server environment. This is analogous to a Java applet that runs in a Web browser environment.

session bean

(*Jolt*) A nonpersistent object that implements some business logic running on the server. A session bean can be thought of as a logical extension of the client that runs on the server. A session bean is not shared among multiple clients.

session key

Used with the symmetric key algorithm in which encryption and decryption involve the same key: the session key. Data encrypted with a session key can only be decrypted with the same session key.

SHA-1

Secure Hash Algorithm 1. An algorithm specified in the Secure Hash Standard that takes as input a message of any length greater than 264 bits and produces as output a 160-bit *message digest,* or *hash value,* of the input. It is slightly slower than MD5, but the larger message digest makes it more secure against brute-force collision and inversion attacks. The SHA-1 algorithm is intended for digital signature applications, where a large file must be compressed in a secure manner before being encrypted with a private key under a public-key cryptosystem such as PKCS.

shared connection transaction

A transaction that appears to span processes by using a single, shared connection to a single database.

SHM model

A BEA Tuxedo application that runs entirely on a single computer, even if that computer is a symmetric multiprocessor.

simple events

The BEA Systems, Inc. proprietary events interface. As the name implies, this interface is designed to be simple to use.

Simple Network Management Protocol (SNMP)

A de facto standard network management protocol developed by the Internet community.

single threading

The complete execution of a program. Processing of one transaction is completed before another transaction is started.

singleton object

(*CORBA)* An object that can appear only once in a process address space.

skeleton

(*CORBA)* A public abstract class generated by an IDL compiler that provides the ORB with information required to dispatch method invocations to servant objects. A server skeleton, like a client stub, is specific to the IDL interface from which it is generated. A server skeleton is the server-side analog to a client stub. Client stubs and skeletons are used by ORBs in static invocation.

SMUX

Stands for SNMP multiplexing. A protocol for master agent/subagent communication defined by RFC 1227.

SMUX subagent

The SNMP Multiplexing (SMUX) protocol allows the creation of subagents that communicate with the agent and resolve management operations for specific objects in the MIB module.

SNMP

See **Simple Network Management Protocol (SNMP)**.

SNMP agent

An agent that uses the SNMP protocol to exchange data with a system manager.

socket

An endpoint of communication to which a name may be bound. The socket interface is a network access method supported by the BEA Tuxedo system. The logical end point of a TCP/IP connection. An application accesses a TCP/IP connection through a socket.

socket descriptor

A TCP/IP-assigned number that uniquely identifies a socket and TCP/IP connection. An application must specify the socket descriptor in TCP/IP API calls to identify the socket/connection.

socket ID

See **socket descriptor**.

socket number

See **socket descriptor**.

SQL

Structured Query Language. A non-procedural language for defining and accessing relational databases. SQL has become the industry standard database language.

standard MIB

A MIB developed as a standard by the Internet community. Examples are MIB I and MIB II.

StarLAN

An AT&T LAN product.

state

(ATMI) The situation of a conversation from the point of view of one of the participating transactions. The conversation state determines the commands that a transaction can validly issue. The state of each transaction changes dynamically in the course of conversation.

(CORBA) A description of the current situation of an object. State is typically described in memory.

stateful application

An application that retains state information in memory after a service or an operation has been performed.

stateful session bean

(*CORBA)* A bean that preserves information about the state of its conversation with the client. This conversation may consist of several calls that modify the conversation state.

stateless application

An application that flushes state information from memory after a service or an operation has been performed.

stateless session bean

(*CORBA)* A bean that does not save information about the state of its conversation with the client.

STRING buffer

(ATMI) A data structure that is an array of non-null characters terminated by the null character. It is a self-describing buffer.

stroid

(*CORBA)* An object ID represented as a string.

See **object identifier (OID)**.

Structured Event

(*CORBA)* A COS Structured Event as defined by the CORBAservices Notification Service. A Structured Event contains a Fixed header, Variable header, Filterable body parts, and a Remaining body.

Structured Query Language

See **SQL**.

subagent

A component of the master agent protocol that fulfills requests and replies to the master agent.

subroutine

(ATMI) A sequenced set of instructions that can be used in one or more programs and at one or more points in a program. The execution of a subroutine is usually invoked by a call.

subscribe

The act of registering to receive structured events.

subscriber

An application program that subscribes to an event or set of events, and declares what action should take place when an event is posted.

symmetric key algorithm

An algorithm for encrypting or decrypting data with the same key, known as a *session key*. A random number generator creates a new session key for each communication, which makes it difficult for a would-be attacker to reuse previous communications.

Examples of symmetric key algorithms are DES, RC2, and RC4.

synchronization

A coordinated commitment control process between communicating transactions that ensures that all logically related updates to recoverable resources are completed or that all are backed out.

synchronous

- Pertaining to an event that happens, exists, or arises at precisely the same time as another event.
- Pertaining to an operation that occurs regularly or predictably with regard to the occurrence of a specified event in another process; for example, the calling of an input/output routine that receives control at a precoded location in a program.

synchronous communication

A method of transmitting data using a timing signal. With this form of communication, when one piece of software sends a message to another piece of software, the originating software must wait until the service provider completes the request before it can continue.

synchronous process

A process that cannot be executed independently of other processes. When a request is processed synchronously, the client must wait until the service provider completes the request before continuing.

system

BEA TOP END term that is equivalent to Tuxedo domain.

***See* Tuxedo domain.**

system administration

- The preparation of an instance of a system for use in a particular setting or installation.

- Changing the system as the installation changes. This term is often used in a way that includes system operations as well.

system administrator

See **administrator**.

system manager

The part of a network management system that requests data from an agent and takes actions based on that data.

system operations

Tasks that need to be performed regularly for a system. Examples are backing up and restoring data and logs, monitoring the system for error conditions, and so forth.

T

task

One execution of a transaction.

TCP/IP

See **Transmission Control Protocol/Internet Protocol (TCP/IP)**.

TDomain gateway

The domain gateway that handles communication between two BEA Tuxedo domains.

See also **domain gateway**.

terminal

- A computer monitor.
- A point in a system or communication network at which data can either enter or leave.

thread

The basic unit of program execution. A process can have several threads running concurrently. Each thread can be performing a different job, such as waiting for events or performing a time-consuming task that the program does not need to complete before the program continues. Generally, when a thread has finished performing its task, the thread is suspended or destroyed.

See also **worker thread**.

threading

An operating system capability that allows a single process to be divided into multiple entities, each of which executes independently while sharing a common address space.

thread-per-object concurrency model

(*CORBA*) A threading model supported by a multithreaded CORBA server application. In the thread-per-object concurrency model, each active object in the server process is associated with a single thread at any one time. Each request for an object establishes an association between a dispatch thread and the object.

See also **multithreaded CORBA server application**.

thread-per-request concurrency model

(*CORBA*) A threading model supported by a multithreaded CORBA server application. In the thread-per-request concurrency model, each request from a client is processed in a different thread of control.

See also **multithreaded CORBA server application**.

thread pool

(*CORBA*) A means to reduce the cost of managing threads in multithreaded CORBA server implementations. At startup and as needed, threads are created, assigned, and released to a pool of available threads where the thread waits until it is needed again to process future requests. Thread pools can be used to support any of the threading models.

See also **multithreaded CORBA server application**.

three-tier client/server architecture

An implementation of *n*-tier client/server architecture.

See also **n-tier client/server** *and* **two-tier client/server**.

tie class

(*CORBA)* Classes that are generated by the IDL compiler when the delegation-based approach to programming is used. The delegation-based approach to programming is used when the overhead of inheritance is too high or cannot be used. For example, due to the invasive nature of inheritance, implementing objects using existing legacy code might be impossible if inheritance for some global class were required.

In the delegation-based approach, the implementation does not inherit from the POA skeleton class. Instead, a wrapper class inherits from the POA skeleton and delegates upcalls to an implementation that is coded as required. This *wrapper class*, called a tie class, is generated by the IDL compiler, along with the same skeleton class used for the inheritance approach. Like the skeleton, the tie class provides a method corresponding to each OMG IDL operation for the

associated interface; however, you may need to modify the tie class to adapt it to the interface of your legacy object. The name of the generated tie class is the same as the generated skeleton class, with the addition that the string `_tie` is appended to the end of the class name.

TLI

See **Transport Layer Interface (TLI)**.

tlisten

A network-independent listener process that runs as a daemon process and provides remote service connections for other BEA Tuxedo system processes.

TLOG

See **transaction log (TLOG)**.

TM

See **transaction manager (TM)**.

TMFFNAME

(*CORBA)* A server application provided with BEA Tuxedo software that runs the FactoryFinder and supporting NameManager services that maintain a mapping of application-supplied names to object references.

See **factory finder** and **object reference**.

TMIFRSVR

(*CORBA)* A server application provided with BEA Tuxedo software for accessing the Interface Repository. The API is a subset of the Interface Repository API defined by CORBA. For a description of the Interface Repository API, see the *C++ Programming Reference.*

See also **application programming interface (API)**, **conversation**, *and* **Interface Repository**.

TMS

See **Transaction Manager Server (TMS)**.

token

An individual element in the message definition block, such as the command or the subsystem name.

TP

See **transaction processing (TP)**.

TP Framework

See **WebLogic Express**.

TP monitor

See **transaction processing monitor (TP monitor)**.

TP protocols

See **transaction processing protocols (TP protocols)**.

TPSUT

See **transaction processing service user title (TPSUT)**.

transaction

- A complete unit of work that transforms a database from one consistent state to another. In DTP, a transaction can include multiple units of work performed on one or more systems.
- A logical construct through which applications perform work on shared resources (for example, databases). The work done on behalf of the transaction conforms to the four ACID properties: atomicity, consistency, isolation, and durability.

See **distributed transaction processing (DTP)**.

transaction coordinator

A system software component that provides the infrastructure that guarantees the integrity and consistency of an operation and the data involved in a transaction.

See **transaction manager (TM)**.

TransactionCurrent

(*CORBA)* An object that is used to manage transactions. The TransactionCurrent object supports all of the methods of the Current object in the `CosTransactions` module. In addition, the TransactionCurrent object supports APIs to open and close the resource manager.

TransactionCurrent defines the methods that allow a client of the CORBAservices Object Transaction Service (OTS) to explicitly manage the association between threads and transactions. This object also defines methods that simplify the use of the OTS for most applications.

See also **application programming interface (API)**, **CORBAservices Object Transaction Service (OTS)**, **Credentials object**, *and* **resource manager (RM)**.

transaction log (TLOG)

The BEA Tuxedo system log that keeps track of global transactions.

transaction manager (TM)

A system software component that manages global transactions on behalf of application programs. A transaction manager coordinates commands from application programs and communication resource managers to start and complete global transactions by communicating with all resource managers that are participating in those transactions. When resource managers fail during global transactions, transaction managers help resource managers decide whether to commit or roll back pending global transactions.

See also **transaction coordinator**.

Transaction Manager Server (TMS)

A BEA Tuxedo system server process that manages the two-phase commit protocol and recovery for global transactions.

transaction policy

(*CORBA)* The policy that determines the TP Framework's or EJB container's interaction between the client request (which may be associated with a transaction) and the servant's transaction context.

transaction processing (TP)

Transaction processing consists of a set of convenient functions that make it easy for you to write code that does the following:

1. Initialize the server application and execute startup and shutdown routines.
2. Tie the server application to BEA Tuxedo domain resources.
3. Perform housekeeping functions.

transaction processing monitor (TP monitor)

A class of products that provide a transaction execution environment on top of conventional operating systems.

transaction processing protocols (TP protocols)

A set of standard protocols by which transaction processing managers on heterogeneous systems interoperate.

transaction processing service user title (TPSUT)

A value that is used to identify OSI TP endpoints that exist within one application entity.

Transaction Service

See **CORBAservices Object Transaction Service (OTS)**.

transactions per second (TPS)

A throughput rating used in conjunction with the standard transactions defined by TPC. Often the rating implies the maximum number of transactions that can be processed by the system in which 90% of the transaction's response times are less than two seconds.

transient object

(*CORBA)* An object that exists only for the lifetime of the process within which it is created.

See **persistent object**.

translation

The process of changing how intrinsic data types are represented in input data and output data (with respect to word length, byte ordering, and character encoding).

Transmission Control Protocol/Internet Protocol (TCP/IP)

The suite of standard communications protocols that defines the Internet and is supported by the transport layer interface. Originally designed for the UNIX operating system, TCP/IP software is now available for every major operating system.

transport interface

The programming interface used to access a network provider. Transport interfaces are typically network provider independent to an extent.

Transport Layer Interface (TLI)

The standard UNIX system user-level interface to data communications features as defined by level 4 of the OSI communications protocol. It is a network access method supported by the BEA Tuxedo system.

transport provider

See **network provider**.

trap

An SNMP data packet that contains information about an error that occurred with a managed object. Traps are unsolicited event notifications, that is, notifications generated by an agent on its own initiative.

TUXCONFIG file

The binary version of the configuration file for a BEA Tuxedo application. This file is accessed by all BEA Tuxedo processes for all configuration information.

See also **application**, **client**, **server**, *and* **UBBCONFIG file**.

Tuxedo

See **BEA Tuxedo system**.

Tuxedo domain

See **domain**.

Tuxedo MIB

A Tuxedo internal data structure for resources. Specifically, it is a BEA Tuxedo or WebLogic Enterprise framework component that provides a complete definition of the object classes and their attributes that together constitute the Tuxedo or WebLogic Enterprise framework. The total Tuxedo system management information base is organized into a generic MIB and component-specific MIBs for each major component. Configuration and administration of the Tuxedo or WLE framework can be done programmatically.

Tuxedo remote client application

See **CORBA remote client application**.

two-phase commit (2PC)

A method of coordinating a single transaction across more than one DBMS (or other resource manager). It guarantees data integrity by ensuring that transactional updates are committed in all of the participating databases, or are fully rolled back out of all the databases, reverting to the state prior to the start of the transaction.

two-tier client/server

An application development approach that splits an application into two parts and divides the processing between a desktop workstation and a server machine.

TX interface

The Transaction Demarcation (TX) API used by application programs to call the transaction manager. Application programs use the TX interface to define the boundaries of global transactions and direct the completion of those transactions.

type conversion

(ATMI) The process of converting an application program's data buffer or record so that the data is formatted in a manner that is suitable to a target application program.

typed buffer

(ATMI) A buffer for message communication involving data of a specific type.

See **buffer types**.

U

UBBCONFIG file

The ASCII version of the configuration file for a BEA Tuxedo application. A binary version, referred to in BEA Tuxedo documentation as the `TUXCONFIG` file, is generated from the `UBBCONFIG` file.

See also **application**, **client**, **server**, *and* **TUXCONFIG file**.

UDP

See **user datagram protocol (UDP)**.

unadvertised

A service is unadvertised when there is no service table entry for it in the BEA Tuxedo bulletin board.

uniprocessor

A computer that has only one CPU.

See also **multiprocessor**.

Universal Device List (UDL)

A system-wide list of devices (either raw disk slices or UNIX files) on which space is allocated for `TUXCONFIG` configuration tables, BEA Tuxedo transaction logs, and, possibly, databases. Its location is specified by the `TUXCONFIG` environment variable.

URL

Uniform Resource Locator. The address of a site on the Web. An example of a URL is `www.bea.com`.

use case

Text that describes how a user will interact with the application that is being designed. The use case reflects the processes the user will follow.

See also **application**.

user datagram protocol (UDP)

The TCP/IP datagram transport layer protocol.

user sponsor

The code that calls the security interfaces for user authentication.

See also **authentication**.

UserTransaction environmental object

(*CORBA)* An object that connects the client application to the BEA Tuxedo transaction subsystem, wherein the client application can perform operations within the context of a transaction. The UserTransaction object exists only with Java client applications.

UserTransaction interface

The interface that defines the methods that allow an application to explicitly manage transaction boundaries.

See also **interface**, **metadata interface**, *and* **TransactionCurrent**.

V

view

(*ATMI)* In the VIEW System Manager, a window that is displayed when you click on an icon that represents a managed host.

(*CORBA)* A representation of a CORBA object in a BEA Tuxedo domain that resides in another object system.

See also **CORBA object**, **object system**, *and* **CORBA domain**.

VIEW buffer

(ATMI) A data structure similar to a C structure. As part of defining this buffer type, a view description file is created. It is a self-describing buffer. VIEW buffers are always accompanied by VIEW definitions.

VIEW definitions

(ATMI) Descriptions of data structures that are used for input and output in the BEA Tuxedo environment.

virtual machine

The functional equivalent of a computer and its associated devices that is controlled by a user at a terminal.

virtual one-hop network

A network in which all nodes can be reached from all other nodes in exactly one transmission. This implies nothing about the physical configuration of the network, which may be a ring, star, bus, or any other valid configuration as long as it appears that all nodes are fully interconnected.

Volume Table of Contents (VTOC)

A file that contains the BEA Tuxedo system and possibly database tables.

VTOC

See **Volume Table of Contents (VTOC)**.

W

WAN

See **Wide Area Network (WAN)**.

Web GUI

See **BEA Administration Console**.

WebLogic Express

(CORBA) An implementation of JDBC for use with Java applets or applications.

WebLogic Server

A pure Java application server for assembling, deploying, and managing distributed Java applications.

Wide Area Network (WAN)

A public or private data communications system in which data is transmitted primarily over communication lines.

window

An area of a user's screen in a graphical user interface system. A window is a mechanism used by applications for interacting with a user.

wiring

(CORBA) An indication that a bean is registered as a listener of events from another bean.

WLS

See **WebLogic Server**.

worker thread

(*CORBA)* A thread that is scheduled to execute a request from a client application. The BEA Tuxedo Java software uses a thread pooling model, where a pool of available worker threads is managed by the software. When the BEA Tuxedo Java software receives a request from a client application, the software schedules, from the thread pool, an available worker thread to execute the request. When the request is complete, the worker thread returns to the thread pool. A worker thread can serve only one request at a time.

See **thread**.

Workstation

The component of the BEA Tuxedo system that allows application clients to be located on sites on which no server-side components of the BEA Tuxedo system are installed. A Workstation client does not support an administration server, an application server, or a bulletin board. All communication between such a client and the rest of the application takes place over a network.

Workstation client (WSC)

A client process that runs on a machine on which no BEA Tuxedo server software is installed. Multiple Workstation clients can run simultaneously on Windows, Windows NT, UNIX, and VMS platforms. The ATMI is available to Workstation clients.

Workstation Handler (WSH)

A process that manages one or more connections between Workstation clients and native BEA Tuxedo servers. Specifically, a Workstation Handler makes surrogate service requests, manages transactions, and returns replies. WSH processes are started and stopped by a Workstation Listener (WSL). The WSH process resides within the administrative domain of the application. Handlers are registered in the local BEA Tuxedo bulletin board as clients.

See **IIOP Handler (ISH)** and **Workstation Listener (WSL)**.

Workstation Listener (WSL)

A process that assigns Workstation Handlers (WSHs) to Workstation clients (WSCs). Once such an assignment is made, the designated WSH manages all service requests from the specified client. The WSL also manages the pool of Workstation Handlers, starting and stopping them in response to load requirements. The WSL process resides within the administrative domain of the application.

See **IIOP Listener (ISL)** and **Workstation Handler (WSH)**.

wrap

To enclose an application in a software layer to make the application available to other applications.

See also **application** *and* **wrapper**.

wrapper

(CORBA) The enclosure that is used to wrap a legacy application to make the legacy application available as an implementation to CORBA client applications.

See also **legacy application**, **client application**, and **Workstation Handler (WSH)**.

wrapper object

See **tie class**.

X

X.509

A standard that specifies the format of certificates, which provide a way to securely associate a name to a public key, providing strong authentication.

See also **authentication** and **certificate**.

XA

The bidirectional, system-level interface used for communication among applications, transactions managers, and database systems, as defined in the X/Open distributed transaction processing (DTP) model.

XATMI application service element

Software that maps primitives in the XATMI interface to the OSI TP protocol.

XATMI interface

An interface that enables application programs to use request/response communication and conversational communication during global transactions.

X_COMMON buffer

(ATMI) A non-nested C structure whose elements are any of the following C data types: short, long, or char. X_COMMON is one of three buffers that are defined in the X/Open XATMI standard. It is equivalent to the BEA Tuxedo VIEW buffer; however, X_COMMON represents only the subset of field types that are common to both the C and COBOL languages.

X_C_TYPE buffer

(ATMI) A non-nested C structure whose elements are any of the following C data types: int, short, long, char, float, double, character string, and octet array. X_C_TYPE is one of three buffers that are defined in the X/Open XATMI standard. It is equivalent to the BEA Tuxedo VIEW buffer.

XDR

See **External Data Representation (XDR)**.

XML

eXtensible Markup Language. A language developed by the World Wide Web Consortium (W3C) and organized by Sun Microsystems, Inc. Used in specifying the Server Descriptor file and the EJB deployment descriptor.

X_OCTET buffer

(ATMI) An array of bytes whose structure is defined by an application. X_OCTET is one of three buffers that are defined in the X/Open XATMI standard. It is equivalent to the BEA Tuxedo CARRAY buffer.

X/Open

The X/Open Company, Ltd., an international private consortium of vendors and users working to establish standards for open systems. BEA Tuxedo products are designed to implement X/Open standards for distributed transaction processing.

X/Open Distributed Transaction Processing (DTP) model

The distributed transaction processing model specified in standards developed by the X/Open Company, Ltd. The BEA Tuxedo architecture is based on these standards. The model defines four components of a DTP system:

- Application programs (APs) define the transaction boundaries and perform the actions that make up the transaction (typically database updates).
- Resource managers (RMs), such as database management systems, provide access to shared resources.
- Communication resource managers (CRMs) allow application programs to communicate with each other.
- Transaction managers (TMs) assign unique identifiers (XIDs) to transactions, monitor the progress of transactions, and handle transaction completion or recovery.

Y

(No terms starting with the letter "Y")

Z

(No terms starting with the letter "Z")