BEA WebLogic Enterprise

Glossary

WebLogic Enterprise 5.0
Document Edition 5.0
December 1999

Copyright

Restricted Rights Legend

Trademarks or Service Marks

Glossary

Document Edition	Date	Software Version
5.0	December 1999	BEA WebLogic Enterprise 5.0

About This Document

This document contains a list of terms and definitions used in the BEA WebLogic Enterprise (WLE) online documentation.

What You Need to Know

This document is intended for all users of the BEA WebLogic Enterprise software.

e-docs Web Site

The BEA WebLogic Enterprise product documentation is available on the BEA corporate Web site. From the BEA Home page, click the Product Documentation button or go directly to the "e-docs" Product Documentation page at http://e-docs.beasys.com.

How to Print the Document

You can print a copy of this document from a Web browser, one file at a time, by using the File—>Print option on your Web browser.

A PDF version of this document is available on the WebLogic Enterprise documentation Home page on the e-docs Web site (and also on the documentation CD). You can open the PDF in Adobe Acrobat Reader and print the entire document (or a portion of it) in book format. To access the PDFs, open the WebLogic Enterprise documentation Home page, click the PDF Files button, and select the document you want to print.

If you do not have the Adobe Acrobat Reader, you can get it for free from the Adobe Web site at http://www.adobe.com/.

Related Information

For more information about CORBA, Java 2 Enterprise Edition (J2EE), BEA TUXEDO, distributed object computing, transaction processing, C++ programming, and Java programming, see the WLE Bibliography in the WebLogic Enterprise online documentation.

Contact Us!

Your feedback on the BEA WebLogic Enterprise documentation is important to us. Send us e-mail at **docsupport@beasys.com** if you have questions or comments. Your comments will be reviewed directly by the BEA professionals who create and update the WebLogic Enterprise documentation.

In your e-mail message, please indicate that you are using the documentation for the BEA WebLogic Enterprise 5.0 release.

If you have any questions about this version of BEA WebLogic Enterprise, or if you have problems installing and running BEA WebLogic Enterprise, contact BEA Customer Support through BEA WebSupport at www.beasys.com. You can also contact Customer Support by using the contact information provided on the Customer Support Card, which is included in the product package.

When contacting Customer Support, be prepared to provide the following information:

- Your name, e-mail address, phone number, and fax number
- Your company name and company address
- Your machine type and authorization codes
- The name and version of the product you are using
- A description of the problem and the content of pertinent error messages

Documentation Conventions

The following documentation conventions are used throughout this document.

Convention	Item
boldface text	Indicates terms defined in the glossary.
Ctrl+Tab	Indicates that you must press two or more keys simultaneously.
italics	Indicates emphasis or book titles.

Convention	Item

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

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

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

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

Convention	Item

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

. . .	Indicates the omission of items from a code example or from a syntax line. The vertical ellipsis itself should never be typed.

Glossary

Access Control List (ACL)

A list of which entities are allowed to invoke which objects and methods.

Access Decision object

The object within the WebLogic Enterprise security infrastructure that enforces the checking of authorized access before a request to the target object is delivered.

ACID properties

The essential characteristics of transaction processing systems:

Atomicity: All changes that a transaction makes to a database are made permanent; otherwise, all changes are rolled back.

Consistency: A successful transaction transforms a database from a previous valid state to a new valid state.

Isolation: Changes that a transaction makes to a database are not visible to other operations until the transaction completes its work.

Durability: Changes that a transaction makes to a database survive future system or media failures.

activation

The process of preparing an object for execution. When using EJBs, the operation where a container reloads the state of a bean from persistent storage and makes the bean available again.

activation policy

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

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

Active object

A running instance of an object interface.

See also **Active Object Map**, **CORBA object**, **enterprise bean**, **object ID (OID)**, **object reference**, **Portable Object Adapter (POA)**, **servant**, and **WebLogic Enterprise (WLE) client application**.

Active Object Map

A table maintained by a POA and the TP Framework, or the EJB container that maps the association of object IDs to servants.

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

ActiveX

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

AdminAPI

The combination of the management information base (MIB) and programmatic interfaces for accessing and manipulating system and run-time information.

API

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

applet

An application written in Java to run within a Web browser that is compatible with Java.

application

In the WebLogic Enterprise system, a single computer program designed to do a certain type of work.

See also **BEA WebLogic Enterprise (WLE) system**.

Application Assembler

A domain expert who composes applications that use EJBs. The application assembler works with the deployment descriptor and client-view contract of the EJB. The application assembler does not have any knowledge of the implementation of an EJB except the interfaces of the EJB that define the contract between the client and the server.

application code

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

application-controlled deactivation

A feature of the WebLogic Enterprise software that you can use 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 programming interface (API)

The verbs and environment that exist at the application level to support a particular system software product. A set of well-defined programming interfaces (that is, entry points, calling parameters, and return values) by which one software program uses the services of another.

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**.

ATMI

Application to Transaction Monitor Interface. The application interface to the BEA TUXEDO system that includes transaction routines, message handling routines, service interface routines, and buffer management routines.

attribute

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 also **CORBA object** and **object**.

authentication

The process of determining whether someone or something is, in fact, who or what it is declared to be.

authorization

The granting of authority, which includes granting access based on access rights.

BEA ActiveX Client

The component of the WebLogic Enterprise software that provides interoperability between a WLE domain and the ActiveX object system. The ActiveX Client translates into ActiveX methods the interfaces of CORBA objects that are located in the WLE domain.

The ActiveX Client has two components: the BEA Application Builder and the Object Bridge.

See also **ActiveX**, **BEA Application Builder**, **Object Bridge**, and **WebLogic Enterprise (WLE) domain**.

BEA Administration Console

A Java applet that you can download into your Internet browser and use to remotely administer WebLogic Enterprise client and server applications and BEA TUXEDO systems. The BEA Administration Console offers a convenient graphical user interface (GUI) for performing your system administration tasks.

See also **BEA TUXEDO system**, **WebLogic Enterprise (WLE) client application**, and **WebLogic Enterprise (WLE) server application**.

BEA Application Builder

The component of the WebLogic Enterprise software that creates ActiveX bindings for CORBA interfaces.

See also **ActiveX**, **BEA WebLogic Enterprise (WLE) software**, **binding**, and **CORBA**.

Bean-managed persistence

For entity beans, using handwritten SQL (or some other handwritten persistence mechanism) within the bean to manage the storage and retrieval of its state information.

Bean Provider

An application programmer who produces enterprise bean classes, remote and home interfaces, and descriptor files, and package them in an EJB `.jar` file.

BEA TUXEDO application

One or more BEA TUXEDO domains cooperating to support a single business function.

See also **BEA TUXEDO domain**.

BEA TUXEDO domain

A collection of servers, services, and associated resource managers defined by a single `UBBCONFIG` or `TUXCONFIG` configuration file.

See also **`TUXCONFIG` file**, **`UBBCONFIG` file**, and **WebLogic Enterprise (WLE) domain**.

BEA TUXEDO system

The BEA TUXEDO software as the customer receives it from BEA Systems, Inc.

BEA WebLogic Enterprise (WLE) software

The BEA WebLogic Enterprise product as the customer receives it from BEA Systems, Inc.

See also **BEA WebLogic Enterprise (WLE) system**.

BEA WebLogic Enterprise (WLE) system

The BEA WebLogic Enterprise software and the hardware on which the WebLogic Enterprise software is running.

See also **BEA WebLogic Enterprise (WLE) software**.

BEA Wrapper Callbacks API

An application programming interface designed specifically to simplify the implementation of callback objects for WebLogic Enterprise 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**.

binding

In the BEA ActiveX Client, the association of the interface of a CORBA object to another object system, such as an ActiveX object system.

See also **ActiveX**, **BEA ActiveX Client**, **CORBA object**, and **object**.

Bootstrap environmental object

The object that brings a CORBA application into the WebLogic Enterprise domain and provides initial object references to that application. Every CORBA client or server application that interacts with a WebLogic Enterprise domain needs a Bootstrap environmental object.

See also **environmental object**, **object**, **object reference**, and **WebLogic Enterprise (WLE) domain**.

bootstrapping

The process of setting up an application to interact with CORBA objects that are located within the WebLogic Enterprise domain.

See also **Bootstrap environmental object**, **CORBA object**, and **WebLogic Enterprise (WLE) domain**.

business object

An application-level component that can be used in unpredictable combinations. 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**.

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

An object implemented to support callbacks on CORBA joint client/server applications using the BEAWrapper Callbacks API.

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

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 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**.

Certificate-based 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 also **authentication** and **certificate**.

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 Socket Layer (SSL)**.

class

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**, **Java interface**, **metadata interface**, and **object**.

client

Any code that invokes an operation on a distributed object.

client application

See **WebLogic Enterprise (WLE) client application**.

Client Data Caching design pattern

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**.

CMP

See **container-managed persistence (CMP)**.

COM

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

comment

In an application, explanatory text that is ignored by the compiler. In Java applications, comments are delimited using a double forward slash (`//`) or a forward slash, followed by an asterisk, explanatory text, an asterisk, and a forward slash (for example, `/*sample code*/`).

See also **application** and **Java**.

Common Object Request Broker Architecture

See **CORBA**.

Component Object Model (COM)

A collection of services that let software components interoperate in a networked environment.

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

COM view

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**.

constructor

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**.

container

A system that functions as the infrastructure for EJBs. Multiple EJBs can be deployed in the same container. The container is responsible for managing transactions, the state of an EJB, resource pooling, deployment, administration, naming, and other object services.

container-managed persistence (CMP)

For entity beans, allowing the container to take care of managing the state information of the entity bean.

CORBA

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

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

CORBA callback object

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 WebLogic Enterprise domain.

See also **client application**, **CORBA object**, **RMI client stub**, and **WebLogic Enterprise (WLE) 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 WLE 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**, **RMI client stub**, and **WebLogic Enterprise (WLE) domain**.

CORBAfacilities

The adopted OMG Common Facilities. Common Facilities provide horizontal end-user-oriented frameworks that are applicable to most domains, and are defined in OMG IDL.

See also **OMG IDL**.

CORBA interface

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 object

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

Any Object Request Broker (ORB) that complies with the CORBA standard. A CORBA ORB is a communications intermediary between client and server applications that typically are distributed across a network. The WebLogic Enterprise ORB is a CORBA ORB.

See also **CORBA**.

CORBAservices

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 Object Transaction Service (OTS), CORBAservices Security Service**, **object**, **object reference**, and **OMG IDL**.

CORBAservices Life Cycle Service

The CORBAservice that defines conventions for creating, deleting, copying, and moving objects.

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

CORBAservices Object Transaction Service (OTS)

The CORBAservice that provides transaction semantics to ensure the integrity of data in the system.

See also **CORBAservices**, **Java Transaction API (JTA)**, and **Java Transaction Service (JTS)**.

CORBAservices Security Service

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

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

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**.

Current

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**.

decryption private key

An algorithm that reverses the work of the encryption algorithm.

deployable JAR file

A JAR file that contains container-specific classes and interfaces that are used by the container to manage EJBs at runtime. A deployable JAR file also contains the context of the `ejb.jar`.

Deployer

A person who has expertise at a specific operational environment (for example, mapping security roles to user groups and accounts) who is responsible for the deployment of EJBs. The Deployer uses tools supplied by BEA to perform deployment tasks.

deployment descriptor

An XML file that specifies two broad categories of information about an EJB: the structure of the EJB and its external dependencies, and how the EJB is composed into a larger application deployment unit. The types of information specified in an EJB's deployment descriptor include (but is not limited to): the bean's name, its class, its home and remote interfaces, the bean's type (entity versus session), persistence information, security roles, and transaction attributes.

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

A client application that operates on a Microsoft desktop platform, such as Windows NT or Windows 95. Desktop client applications use the Component Object Model (COM) and communicate with the WebLogic Enterprise domain by using the ActiveX Client to translate between COM and CORBA.

See also **BEA ActiveX Client**, **Component Object Model (COM)**, **CORBA**, and **WebLogic Enterprise (WLE) domain**.

digital signature

An electronic signature that can be used to authenticate the identity of the sender of a message. A digital signature can also be used to ensure that the original content of a message that has been sent is unchanged.

DII

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

DSI

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

Distinguished Name (DN)

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 object

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.

domain

See **BEA TUXEDO domain** and **WebLogic Enterprise (WLE) domain**.

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 and for WebLogic Enterprise domains.

See also **BEA TUXEDO domain** and **WebLogic Enterprise (WLE) domain**.

dual-paired connections outbound IIOP

See **outbound IIOP**.

Dynamic Invocation Interface (DII)

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

Dynamic Skeleton Interface (DSI)

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

encryption

The conversion of data into a form, called a cipher, that cannot be easily understood by unauthorized people.

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.

EJB home interface

An EJB component interface that allows clients to look up and/or create EJBs.

EJB remote interface

Defines the business methods callable by a client. The remote interface extends the `javax.ejb.EJBobject` interface.

EJB roles

The EJB specification defines six distinct roles in the application development and deployment cycle of an EJB. Note that all the EJB roles may be performed by one person. BEA is an EJB Server and an EJB container provider.

See also **Application Assembler** and **Deployer**.

EJB JAR file

Used by EJB tools for packaging EJBs with their declarative information. The EJB JAR file is intended to be processed by application assembly and deployment tools.

enterprise bean

A software component that describes the general bean implementation. An enterprise bean can be either a session bean or an entity bean.

Enterprise JavaBeans (EJB)

An API specification for building scalable, distributed, component-based, multitiered applications. EJBs leverage and extend the JavaBeans component model to provide a rich, object-oriented transactional environment for developers creating enterprise applications.

entity bean

A long-lived bean that represents data in an underlying data store and can be shared by multiple clients.

environmental object

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**.

exception

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

In Java, system exceptions inherit from `org.omg.CORBA.SystemException` (which is a `java.lang.RuntimeException`); user-defined exceptions inherit from `org.omg.CORBA.UserException` (which is a `java.lang.Exception`).

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

factory

Any distributed CORBA object that returns an object reference to other distributed CORBA objects. A factory is located in the server application.

See also **CORBA object**, **object reference**, and **WebLogic Enterprise (WLE) server application**.

factory-based routing

A feature of the WebLogic Enterprise software that permits the routing of requests on an object reference to a specific server group based on criteria supplied at the time the object reference is created by a factory.

See also **BEA WebLogic Enterprise (WLE) software**, **factory**, and **object reference**.

factory finder

The 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**, **WebLogic Enterprise (WLE) client application**, and **WebLogic Enterprise (WLE) server application**.

`factory_finder.ini` file

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**.

foreign client

See **WebLogic Enterprise (WLE) foreign client application**.

garbage collection

The automatic detection and freeing of memory that is no longer in use. The Java run-time system performs garbage collection so that programmers never explicitly free objects.

General Inter-ORB Protocol (GIOP)

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

A transaction that can execute in more than one server, accessing data from more than one resource manager. A global transaction may be composed of several local transactions, each accessing a single resource manager.

See also **resource manager**.

home interface

See **EJB home interface**.

ICF

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

IDL

See **OMG IDL**.

idl compiler

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

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 WebLogic Enterprise CORBA objects.

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

IDL parameter

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

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

Internet Inter-ORB Protocol. A protocol specified 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

A WLE system process that handles all IIOP communication between a remote application and target WLE objects.

IIOP Listener

A WebLogic Enterprise 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.

IIOP Server Listener/Handler

The feature of the WebLogic Enterprise software that enables client applications to communicate with the WebLogic Enterprise 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 WebLogic Enterprise domain. It also receives a request from a server application in the WebLogic Enterprise domain and sends the request to a server outside the domain.

See also **BEA WebLogic Enterprise (WLE) software**, **IIOP**, **WebLogic Enterprise (WLE) client application**, **WebLogic Enterprise (WLE) domain**, and **WebLogic Enterprise (WLE) server application**.

implementation

A class that defines the behavior for all operations and attributes of the supported interface. There may be many implementations of a single interface.

implementation code

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)

A file that describes the implementation attributes of WebLogic Enterprise C++ server applications. The ICF file is input to the IDL compiler when generating skeletons for WebLogic Enterprise C++ server applications.

See also **skeleton** and **WebLogic Enterprise (WLE) server application**.

implementation file

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 WebLogic Enterprise build procedure.

See also **implementation code**, **metadata interface**, **OMG IDL**, **operation**, and **WebLogic Enterprise (WLE) server application**.

initialContext

An object used to access a service provider using JNDI, which creates a context for the provider.

See also **Java Naming and Directory Interface (JNDI)**.

initial naming context

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 COS Naming Service registered with the ORB. The initial naming context can be used to create other `NamingContext` objects.

See also **naming context**.

instance

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

instantiate

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.

interface

See **IDL interface** and **Java interface**.

Interface Repository

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

See also **CORBA**, **IDL interface**, and **Java interface**.

Interoperable Object Reference (IOR)

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**.

invocation

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

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**.

JAR files (.jar)

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.

Java 2 Enterprise Edition (J2EE)

An environment for developing and deploying enterprise applications. The J2EE platform consists of a set of services, APIs, and protocols that provide the functionality for developing multitiered, Web-based applications.

Java Database Connectivity (JDBC)

An industry standard for database-independent connectivity between Java and a wide range of databases. The JDBC provides a call-level API for SQL-based database access.

See also **Java**.

JDBC Connection Pooling

A mechanism to maintain a cache of reusable database connections. The JDBC 2.0 Standard Extension API defines interfaces between the connection pooling module and the database connections.

Java Development Kit (JDK)

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

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

Javadoc

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 IDL

The classes, libraries, and tools that make it possible to use CORBA objects from the Java programming language. The main components of Java IDL are an ORB, a factory finder, and the idltojava compiler.

Java interface

A declaration used in the Java language to define an abstract interface. Since Java does not have multiple inheritance, a Java class can implement one or more interfaces to provide mix-in functionality.

See also **IDL interface**.

Java Naming and Directory Interface (JNDI)

A standard extension to the Java platform, providing Java technology-enabled applications with a unified interface to multiple naming and directory services in the enterprise. As part of the Java Enterprise API set, JNDI enables seamless connectivity to heterogeneous enterprise naming and directory services.

Java Native Interface (JNI)

A standard Java interface that allows Java code to call C or C++ functions, or C or C++ functions to call Java code.

Java Runtime Environment (JRE)

A subset of the Java Development Kit for end users and programmers who want to redistribute the JRE. The JRE consists of the Java Virtual Machine (JVM), the Java core classes, and supporting files.

See also **core class** and **Java Virtual Machine (JVM)**.

JavaServer

A server process provided by the WLE system that loads the JVM and then your object classes (CORBA, Java, or EJB). You configure the JavaServer in the application's `UBBCONFIG` file and start the JavaServers with the `tmboot` command.

See also **Java Virtual Machine (JVM)**.

Java Transaction API (JTA)

A high-level application transaction interface and a Java mapping to XA. Allows an application to control user transaction boundaries. The XA interface allows an external transaction manager to control transaction boundaries for operations performed by multiple resource managers using the two phase commit X/Open XA protocol. The API is defined in the `javax.transaction` package.

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

Java Transaction Service (JTS)

The Sun Microsystems, Inc. Java Transaction API that specifies a standard mapping of the OMG Object Transaction Service (OTS). The JTS defines a low-level transaction management specification intended for vendors who provide the transaction system infrastructure required to support the application run-time environment. CORBA Java client applications use JTS.

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

Java Virtual Machine (JVM)

The part of the Java Runtime Environment responsible for interpreting Java bytecodes.

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

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**.

JNDI

See **Java Naming and Directory Interface (JNDI)**.

JRE

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

JTA

See **Java Transaction API (JTA)**.

JVM

See **Java Virtual Machine (JVM)**.

LDAP

See **Lightweight Directory Access Protocol (LDAP)**.

legacy application

An existing application that needs to be modified or wrapped so that it can be used by the WLE domain.

See also **WebLogic Enterprise (WLE) domain**, **wrap**, and **wrapper**.

Life Cycle Service

See **CORBAservices Life Cycle Service**.

Lightweight Directory Access Protocol (LDAP)

A specialized database that can be physically distributed across multiple systems for access by many applications within an enterprise. 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**.

Listener/Handler

See **IIOP Handler**.

local factory

A factory object that exists in the local domain that is made available to remote domains through a WebLogic Enterprise factory finder.

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

locality-constrained object

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.

local transaction

A transaction that accesses a single database or file and is controlled by the resource manager responsible for performing concurrency control and atomicity of updates at that database.

See also **ACID properties**, **CORBA ORB**, and **WebLogic Enterprise (WLE) server application**.

Management Information Base (MIB)

A BEA WebLogic Enterprise system component that provides a complete definition of the object classes and their attributes that together comprise the BEA WebLogic Enterprise system.

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

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 also **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.

metadata interface

An interface that accesses data about data; descriptive information about a particular object. In the EJB world, metadata are most often encountered in the EJBMetaData interface and in the JDBC DatabaseMetaData and ResultMetaData interfaces.

method

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 re-used in multiple objects.

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

MIB

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

mutual authentication

The process whereby each side of an intended communication provides 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**.

name binding

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

namespace

A collection of naming contexts that are grouped together.

naming context

An object that contains a set of name associations in which each name is unique.

See also **Java Naming and Directory Interface (JNDI)**.

Naming Service

See **Java Naming and Directory Interface (JNDI)**.

native client application

See **WebLogic Enterprise (WLE) native client application**.

native joint client/server application

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

See also **joint client/server application** and **WebLogic Enterprise (WLE) domain**.

object

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

See also **CORBA object**, **Enterprise JavaBeans (EJB)**, and **RMI object**.

object activation

The process of making an object or EJB ready to accept invocations from client applications. The object or EJB needs to have its methods and sate 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. The result of object activation is that an invocation can be made immediately on a servant to service a client invocation of a method on an object reference.

When using EJBs, the act of restoring a stateful bean instance's state relative to its EJB object. The EJB container associates an object ID to an instance of the EJB in the active object map. The result of object activation is that an invocation can be made immediately on a method to service a client invocation of a method on an object reference.

See also **Active Object Map**, **metadata interface**, **object deactivation**, **object ID (OID)**, **object reference**, **Portable Object Adapter (POA)**, and **servant**.

Object Bridge

Software from Visual Edge Software, Ltd. that provides a framework for object system interoperability.

object deactivation

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 also **Active Object Map**, **object activation**, **object ID (OID)**, **object reference**, **passivation**, and **Portable Object Adapter (POA)**.

object handle

Identifies the object in a portable way. The handle can be serialized, which allows you 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)

A value that uniquely identifies a distributed object of a given interface.

object implementation

The code you write that implements the operations defined for an interface.

See also **interface**.

object interface

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 WebLogic Enterprise software.

See also **BEA WebLogic Enterprise (WLE) software**, **CORBA object**, **OMG**

IDL, and **operation**.

Object Management Group (OMG)

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

The model that reflects as objects the overall object-oriented design of an application or system.

object reference

An identifier that uniquely specifies an instance of an object within a distributed ORB system.

Object Request Broker

See **CORBA ORB**.

object system

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 also **Component Object Model (COM)**, **CORBA**, and **object model**.

Object Transaction Service

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

OID

See **object ID (OID)**.

OLE

Object Linking and Embedding. The part of ActiveX that supports object linking and embedding.

See also **ActiveX**.

OLE Automation

A technology that Microsoft provides as a way to manipulate ActiveX objects from outside the application that defines them.

See also **ActiveX** and **application**.

OMG IDL

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**.

operation

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

The main procedure of the WebLogic Enterprise server application process. The WebLogic Enterprise 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.

See also **BEA WebLogic Enterprise (WLE) software** and **WebLogic Enterprise (WLE) server application**.

OTS

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

outbound IIOP

A feature of the WebLogic Enterprise software that supports client callbacks. Outbound IIOP adds the outbound half-gateway to the ISL/ISH.

The WebLogic Enterprise product supports three types of outbound IIOP, as follows:

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 WebLogic Enterprise software is supported for GIOP 1.0, GIOP 1.1, and GIOP 1.2 client applications, server applications, and joint client/server applications.

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 WebLogic Enterprise C++ GIOP 1.2 client applications, server applications, and joint client/server applications.

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 WebLogic Enterprise software is supported for GIOP 1.0, GIOP 1.1, and GIOP 1.2 client applications, server applications, and joint client/server applications.

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 also **object reference** and **transient object**.

PIDL (Pseudo-IDL)

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.

POA

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

policy

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

Portable Object Adapter (POA)

A run-time library of functions that are built in to the 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 WebLogic Enterprise server application procedure automatically builds the POA into the server application. The WebLogic Enterprise TP Framework automatically handles all the WLE server application interactions with the POA. Note that joint client/server applications interact directly with the POA.

See also **CORBA object, object reference, state**, **WebLogic Enterprise (WLE) server application**, and **WebLogic Enterprise (WLE) TP Framework**.

pragma

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.

Principal Authenticator object

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.

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**.

Pseudo-IDL

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

pseudo-object

An object similar to a CORBA object in that it is described in ID; 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.

publish

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

ReceivedCredentials object

The object that represents the secure association to the application. The ReceivedCredentials object contains the properties of that association.

See also **object**.

remote client application

See **WebLogic Enterprise (WLE) remote client application**.

remote factory

A factory object that exists in a remote domain that is made available to the application through a WebLogic Enterprise factory finder.

See also **factory**, **factory finder**, **local factory**, and **WebLogic Enterprise (WLE) domain**.

remote interface

The interface that the client uses to interact with an EJB on a server.

See also **RMI remote interface**.

remote joint client/server applications

A CORBA joint client/server application that is located outside a WebLogic Enterprise domain. The remote joint client/server application does not use the WebLogic Enterprise 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 also **CORBA ORB**, **WebLogic Enterprise (WLE) client application**, and **WebLogic Enterprise (WLE) TP Framework**.

Remote Method Invocation (RMI)

A Java-specific API for accessing remote objects.

request

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

See also **CORBA ORB** and **WebLogic Enterprise (WLE) server application**

resource manager

An interface and associated software that provide 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 entities accessed and controlled within a global transaction.

See also **transaction manager**.

request-level interceptor

A user-written application that is inserted into the invocation path between the client and server components of a WebLogic Enterprise 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.

RMI callback object

A remote object created by an RMI client and sent to a server. The remote object is essentially the client interface, and what is passed to the server is actually a reference to the object. The server can then invoke methods on this object as if it were local using the object reference (which is equivalent to a client stub).

RMI client stub

A generated proxy class used by the clients of a remote object. The stub marshals the invoked method name and its arguments for the client, forwards these to the remote object, and unmarshals the returned results for the client. An RMI client stub is generated by running the `weblogic.rmi` compiler on the fully qualified package names of compiled class files that contain remote object implementations, like `my.package.MyImpl`.

RMI object

An object that uses an object reference to invoke methods on another object in the network as though the object existed locally on the client's Java Virtual Machine.

RMI on IIOP

The BEA WebLogic version of RMI, implemented to use the CORBA IIOP protocol. Firewalls configured to support IIOP traffic will accept WebLogic RMI on IIOP messages as standard IIOP messages. WebLogic RMI on IIOP passes serialized objects as in traditional RMI. It does not pass objects by value, which is needed for full interoperability between Java clients, EJBs, and CORBA objects.

RMI remote interface

A standard Java interface that extends the `rmi.Remote` interface. In the WLE product, a remote interface must extend either the `java.rmi.Remote` interface or the `weblogic.rmi.Remote` interface. The `rmi.Remote` interface itself contains no method signatures; it simply acts as a tag to identify remote classes. The remote interface the programmer writes (extending `rmi.Remote`) must include all method signatures needed by all remote classes that implement it.

RMI remote object

An instance of a class that implements an RMI remote interface. Every class that can be remotely invoked implements a remote interface. The remote object is sometimes referred to as an RMI server.

RMI server

See **RMI remote object**.

RMI server skeleton

The skeleton class, which exists in the RMI remote object or RMI server, unmarshals the `invokes` method and arguments on the remote object, invokes the method on the instance of the remote object, and then marshals the results for return to the client. In the WLE product, the skeleton class is generated at the same time as the client stub by running the `weblogic.rmic` compiler on the fully qualified package names of the compiled class files that contain remote object implementations. However, the generated skeleton class files can be discarded for WLE applications because the WLE infrastructure handles the unmarshalling, method invocations, and marshalling on the RMI server side.

Secure Socket 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**.

SecurityCurrent

The object that provides access to the security features of the system.

See also **object**.

security policy

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 Service

See **CORBAservices Security Service**.

servant

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

servant factory

A feature of WebLogic Enterprise Java server applications for automatically instantiating servants. Unlike WebLogic Enterprise C++ servers, Java servers do not need to provide a callback for instantiating servants.

servant pooling

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

See also **object ID (OID)**, **servant**, and **WebLogic Enterprise (WLE) server application**.

server application

See **WebLogic Enterprise (WLE) server application**.

Server Description File

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 also **JAR files (.jar)**.

Server object

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

See also **servant** and **WebLogic Enterprise (WLE) server application**.

server-to-server communication

The feature of the WebLogic Enterprise 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 of a WLE domain.

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.

session bean

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.

simple events

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

singleton object

An object that can appear only once in a process address space.

skeleton

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.

state

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 fulfilled.

stateful session bean

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 fulfilled.

stateless session bean

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

stroid

An object ID represented as a string.

See also **object ID (OID)**.

Structured Event

A COS Structured Event as defined by the CORBAservices Notification Service. Structured Events contain a Fixed header, Variable header, Filterable body parts, and a Remaining body.

subscribe

The act of registering to receive structured events.

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**.

tie class

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.

TMFFNAME

A server application provided by BEA Systems, Inc. that runs the FactoryFinder and supporting NameManager services that maintain a mapping of application-supplied names to object references.

See also **factory finder**, **object reference**, and **WebLogic Enterprise (WLE) server application**.

TMIFRSVR

A server application provided by BEA Systems, Inc. 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)**, **CORBA**, **Interface**

Repository, and **WebLogic Enterprise (WLE) server application**.

TP Framework

See **WebLogic Enterprise (WLE) TP Framework**.

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 also **operation** and **transaction manager**.

TransactionCurrent

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**, **Java Transaction API (JTA)**, **Java Transaction Service (JTS)**, and **resource manager**.

transaction manager

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 policy

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 Service

See **CORBAservices Object Transaction Service (OTS)**, **Java Transaction API (JTA)**, and **Java Transaction Service (JTS)**.

transient object

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

See also **persistent object**.

`TUXCONFIG` file

The binary version of the configuration file for a BEA WebLogic Enterprise or a BEA TUXEDO application. This file is accessed by all BEA WebLogic Enterprise and BEA TUXEDO processes for all configuration information.

See also **BEA TUXEDO application**, **`UBBCONFIG` file**, and **WebLogic Enterprise (WLE) server application**.

TUXEDO domain

See **BEA TUXEDO domain**.

TUXEDO remote client application

See **WebLogic Enterprise (WLE) remote client application**.

`UBBCONFIG` file

The ASCII version of the configuration file for a BEA WebLogic Enterprise or a BEA TUXEDO application. This is the file from which the `TUXCONFIG` file is generated.

See also **BEA TUXEDO application**, **`TUXCONFIG` file**, and **WebLogic Enterprise (WLE) server application**.

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 sponsor

The code that calls the security interfaces for user authentication.

See also **authentication**.

UserTransaction interface

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

See also **interface**, **Java Transaction API (JTA)**, **metadata interface**, and **TransactionCurrent**.

UserTransaction environmental object

An object that connects the client application to the WebLogic Enterprise transaction subsystem, wherein the client application can perform operations within the context of a transaction. The UserTransaction object exists only with Java client applications.

view

A representation of a CORBA object in a WebLogic Enterprise domain that resides in another object system, such as ActiveX.

See also **ActiveX**, **CORBA object**, **object system**, and **WebLogic Enterprise (WLE) domain**.

WebLogic Enterprise (WLE) client application

A program that was written to be used with the WebLogic Enterprise software and that requests services from other applications.

See also **BEA WebLogic Enterprise (WLE) software** and **WebLogic Enterprise (WLE) server application**.

WebLogic Enterprise (WLE) domain

A specific instance of the WebLogic Enterprise system, plus customer server applications, plus a single `UBBCONFIG` file. The system administrator uses the `UBBCONFIG` file to configure the WLE domain.

See also **BEA WebLogic Enterprise (WLE) system**, **`UBBCONFIG` file**, and **WebLogic Enterprise (WLE) server application**.

WebLogic Enterprise (WLE) foreign client application

A client application that is implemented on an ORB that is not a product of BEA Systems, Inc. The ActiveX Client component of the WebLogic Enterprise software is not a foreign client application. Although the client is implemented on a Microsoft product, the ORB is provided by BEA Systems, Inc.

See also **BEA ActiveX Client** and **ORB**.

WebLogic Enterprise (WLE) native client application

A client application that invokes operations defined in OMG IDL statements to talk to WebLogic Enterprise server applications. 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 WLE domain. WebLogic Enterprise native client applications are always co-located on a machine in the WLE domain.

See also **OMG IDL**, **WebLogic Enterprise (WLE) domain**, **WebLogic Enterprise (WLE) foreign client application**, **WebLogic Enterprise (WLE) remote client application**, and **WebLogic Enterprise (WLE) server application**.

WebLogic Enterprise (WLE) remote client application

A client application that invokes operations defined in OMG IDL statements to talk to remote WebLogic Enterprise 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 co-located on a machine that is running in the WLE domain. WebLogic Enterprise remote client applications are typically not located on a machine that is running in the WLE domain. The ActiveX Client component of the WebLogic Enterprise software is a remote client application.

See also **BEA ActiveX Client**, **IIOP**, **OMG IDL**, **WebLogic Enterprise (WLE) domain**, **WebLogic Enterprise (WLE) foreign client application**, **WebLogic Enterprise (WLE) native client application**, and **WebLogic Enterprise (WLE) server application**.

WebLogic Enterprise (WLE) server application

A program that was written to be used with the WebLogic Enterprise software and that performs a task requested of it by a client application.

See also **BEA WebLogic Enterprise (WLE) software** and **local factory**.

WebLogic Enterprise (WLE) software

See **BEA WebLogic Enterprise (WLE) software**.

WebLogic Enterprise (WLE) TP Framework

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

1) Initializes the server application and executes startup and shutdown routines

2) Ties the server application to WLE domain resources

3) 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

4) Performs object housekeeping

See also **WebLogic Enterprise (WLE) domain** and **WebLogic Enterprise (WLE) server application**.

WLE client application

See **WebLogic Enterprise (WLE) client application**.

WLE domain

See **WebLogic Enterprise (WLE) domain**.

WLE foreign client application

See **WebLogic Enterprise (WLE) foreign client application**.

WLE native client application

See **WebLogic Enterprise (WLE) native client application**.

WLE server application

See **WebLogic Enterprise (WLE) server application**.

WLE software

See **BEA WebLogic Enterprise (WLE) software**.

WLE system

See **BEA WebLogic Enterprise (WLE) system**.

WLE TP Framework

See **WebLogic Enterprise (WLE) TP Framework**.

worker thread

A thread that is scheduled to execute a request from a client application. The WebLogic Enterprise Java software uses a thread pooling model, where a pool of available worker threads is managed by the software. When the WebLogic Enterprise 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 also **thread**.

wrap

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

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

wrapper

The enclosure that is used to wrap a legacy application to make the legacy application available as an implementation to WebLogic Enterprise client applications.

See also **legacy application**, **WebLogic Enterprise (WLE) client application**, and **wrap.**

wrapper object

See **tie class**.

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 between a transaction manager and a transaction manager of the X/Open distributed transaction processing (DTP) model.

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.