BEA WebLogic Enterprise

Glossary

BEA WebLogic Enterprise 4.2
Document Edition 4.2
July 1999

Copyright

Restricted Rights Legend

Trademarks or Service Marks

Glossary

Document Edition	Date	Software Version
4.2	July 1999	BEA WebLogic Enterprise 4.2

Preface

Purpose of This Document

This document contains a list of terms and definitions used in the BEA WebLogic Enterprise (sometimes referred to as WLE) online information set.

Note: Effective February 1999, the BEA M3 product is renamed. The new name of the product is BEA WebLogic Enterprise (WLE).

Who Should Read This Document

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

How This Document Is Organized

Terms listed in the *Glossary* appear alphabetically. Each term is followed by its definition.

How to Use This Document

This document, *Glossary*, is designed primarily as an online, hypertext document. If you are reading this as a paper publication, note that to get full use from this document you should access it as an online document via the Online Documentation CD for the BEA WebLogic Enterprise 4.2 release.

The following sections explain how to view this document online, and how to print a copy of this document.

Opening the Document in a Web Browser

To access the online version of this document, open the following file:

```
\doc\wle\v42\index.htm
```

Note: The online documentation requires Netscape Communicator version 4.0 or later, or Microsoft Internet Explorer version 4.0 or later.

Printing from a Web Browser

You can print a copy of this document from the Web browser. Before you print, make sure that the *Glossary* is displayed and *selected* in your browser.

The Online Documentation CD also includes Adobe Acrobat PDF files of all of the online documents. You can use the Adobe Acrobat Reader to print all or a portion of each document. On the CD Home Page, click the PDF Files button and scroll to the entry for the document you want to print.

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.

`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 *`
	`.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`*`]...`

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

. . .	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.

Related Documentation

The following sections list the documentation provided with the BEA WebLogic Enterprise software, related BEA publications, and other publications related to the technology.

BEA WebLogic Enterprise Documentation

The BEA WebLogic Enterprise information set consists of the following documents:

Installation Guide

C++ Release Notes

Java Release Notes

Getting Started

Guide to the University Sample Applications

Guide to the Java Sample Applications

Creating Client Applications

Creating C++ Server Applications

Creating Java Server Applications

Administration Guide

Using Server-to-Server Communication

C++ Programming Reference

Java Programming Reference

Java API Reference

JDBC Driver Programming Reference

System Messages

Glossary (this document)

Technical Articles

Note: The Online Documentation CD also includes Adobe Acrobat PDF files of all of the online documents. You can use the Adobe Acrobat Reader to print all or a portion of each document.

BEA Publications

Selected BEA TUXEDO Release 6.5 for BEA WebLogic Enterprise version 4.2 documents are available on the Online Documentation CD.

To access these documents:

1. Click the Other Reference button from the main menu.
2. Click the TUXEDO Documents option.

Other Publications

For more information about CORBA, Java, and related technologies, refer to the following books and specifications:

Cobb, E. 1997. *The Impact of Object Technology on Commercial Transaction Processing*. VLDB Journal, Volume 6. 173-190.

Edwards, J. with DeVoe, D. 1997. *3-Tier Client/Server At Work*. Wiley Computer Publishing.

Edwards, J., Harkey, D., and Orfali, R. 1996. *The Essential Client/Server Survival Guide*. Wiley Computer Publishing.

Flanagan, David. May 1997. *Java in a Nutshell*, 2nd Edition. O'Reilly & Associates, Incorporated.

Flanagan, David. September 1997. *Java Examples in a Nutshell*. O'Reilly & Associates, Incorporated.

Fowler, M. with Scott, K. 1997. *UML Distilled, Applying the Standard Object Modeling Language*. Addison-Wesley.

Gamma, E., Helm, R., Johnson, R., and Vlissides, J. 1995. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Professional Computing Series.

Jacobson, I. 1994. *Object-Oriented Software Engineering: A Use Case Driven Approach*. Addison-Wesley.

Mowbray, Thomas J. and Malveau, Raphael C. (Contributor). 1997. *CORBA Design Patterns*, Paper Back and CD-ROM Edition. John Wiley & Sons, Inc.

Orfali, R., Harkey, D., and Edwards, J. 1997. *Instant CORBA*. Wiley Computer Publishing.

Orfali, R., Harkey, D. February 1998. *Client/Server Programming with Java and CORBA*, 2nd Edition. John Wiley & Sons, Inc.

Otte, R., Patrick, P., and Roy, M. 1996. *Understanding CORBA*. Prentice Hall PTR.

Rosen, M. and Curtis, D. 1998. *Integrating CORBA and COM Applications*. Wiley Computer Publishing.

Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F., and Loresen, W. 1991. *Object-Oriented Modeling and Design*. Prentice Hall.

The Common Object Request Broker: Architecture and Specification. Revision 2.2, February 1998. Published by the Object Management Group (OMG).

CORBAservices: Common Object Services Specification. Revised Edition. Updated: November 1997. Published by the Object Management Group (OMG).

Contact Information

The following sections provide information about how to obtain support for the documentation and software.

Documentation Support

If you have questions or comments on the documentation, you can contact the BEA Information Engineering Group by e-mail at **docsupport@beasys.com**. (For information about how to contact Customer Support, refer to the following section.)

Customer Support

If you have any questions about this version of the BEA WebLogic Enterprise product, or if you have problems installing and running the BEA WebLogic Enterprise software, 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

Glossary

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.

See also **request**.

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 nullified.

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.

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 CORBA object that is immediately ready to service an invocation by a client application. That is, the object has a servant associated with the object ID already in an Active Object Map. For the TP Framework, this means that a servant's `activate_object` method has been called or that the user code called `TP::create_active_object_reference`. For the POA, it means that the server code called one of the `POA::activate_object` methods.

See also **Active Object Map**, **CORBA object**, **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 that maps the association of object IDs to servants.

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

ActiveX

A set of technologies from Microsoft that enables software components to interact with one another in a networked environment, regardless of the language in which the components were created. ActiveX is built on the Component Object Model (COM) and includes OLE functionality, such as OLE Automation.

See also **OLE** and **OLE Automation**.

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 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 outbound IIOP

See **outbound IIOP**.

attribute

An identifiable association between an object and a value.

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

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**.

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**.

BEAWrapper Callbacks API

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

See also **application programming interface (API)**, **callback object**, **Callbacks Wrapper 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 an application into the WebLogic Enterprise domain and provides initial object references to that application. Every 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 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 **method**.

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**, and **WebLogic Enterprise (WLE) domain**.

Callbacks Wrapper object

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

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

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**, **method**, and **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**.

client stub

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 WebLogic Enterprise domain calls when it is invoking a request. The code is used to send the request to the server application.

See also **method**, **OMG IDL**, **request**, and **WebLogic Enterprise (WLE) domain**.

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 `//` or `/*...*/`.

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

Component Object Model (COM)

The object model used on Microsoft platforms. COM is different from CORBA in many ways. For example, there are differences in the mechanisms by which objects are referenced, and in the process by which objects are created.

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**, **method**, and **object**.

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)**.

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 Naming 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 Naming Service

The CORBAservice that provides the ability to associate a name to an object relative to a naming context. A naming context is an object that contains a set of name associations in which each name is unique. To resolve a name is to determine the object associated with the name in a given context.

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 **CORBAservices** 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 where the Java platform runs.

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

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**, **method**, 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**, and **SecurityCurrent**.

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**.

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**.

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 during program execution that prevents the program from continuing normally (usually an error). Java supports exceptions with the `try`, `catch`, and `throw` keywords.

See also **Java**.

factory

Any CORBA object that returns an object reference to other 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 **factory**, **object reference**, **request**, and **BEA WebLogic Enterprise (WLE) software**.

factory finder

The object that locates the factories that an application needs. Both client applications and server applications can use a factory finder. `Tobj::FactoryFinder` is a WLE factory finder.

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.

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 **See also CORBA ORB andresource managerAn 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.**.

ICF

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

identifier

The name of an item in a Java application.

IDL

See **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**.

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 the proper object.

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

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 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**, **method**, **object**, and **request**.

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**, **method**, **OMG IDL**, **operation**, and **WebLogic Enterprise (WLE) server application**.

instance

An object instance in C++ or Java. Object instances are used as servants for CORBA objects in the WebLogic Enterprise product.

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 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 **method**, **policy**, and **request**.

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 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**.

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 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 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 provided by the WebLogic Enterprise system for Java server applications. You start the WebLogic Enterprise JavaServer in the application's `UBBCONFIG` file before starting your application's Java servers. The server that loads the JVM.

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

JavaSoft

A business unit of Sun Microsystems, Inc.

Java Transaction API (JTA)

The API that defines a high-level transaction management specification for resource managers and transactional applications that are deployed in a distributed transaction system.

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

Java Transaction Service (JTS)

The Sun Microsystems, Inc. Java interface for transaction services, based on the 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.

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**.

JRE

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

JTA

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

JVM

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

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**.

Listener/Handler

See **IIOP Server Listener/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**.

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** and **See also CORBA ORB andresource managerAn 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.**.

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.

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**.

method

A method of a C++ or Java class. User-written methods of C++ or Java classes provide implementation of IDL operations for Web logic Enterprise distributed objects.

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

MIB

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

naming context

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

See also **CORBAservices Naming Service**.

Naming Service

See **CORBAservices Naming Service**.

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 WebLogic Enterprise domain. C++ native joint client/server applications are built with the `buildobjclient` command. The WebLogic Enterprise 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**.

object activation

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.

See also **Active Object Map**, **method**, **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

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.

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

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 model

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

object reference

An identifier that associates an object definition with an instance of the object, such as a bank account number or an employee identification number.

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 and COM.

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 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 version 4.2 software that supports client callbacks. In the version 4.0 and 4.1 releases of the WebLogic Enterprise software, the ISL/ISH was an inbound half-gateway. 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 version 4.2 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 version 4.2 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.

persistent object

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

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

POA

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

policy

See **activation 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**.

Principal Authenticator object

The object that is visible to the application that is 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 **object**.

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**.

ReceivedCredentials object

The object that represents the secure association to the application. The Received Credentials 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 joint client/server applications

A 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**, **domain**, **joint client/server application**, **native joint client/server application**, **WebLogic Enterprise (WLE) client application**, and **WebLogic Enterprise (WLE) TP Framework**.

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**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**.

SecurityCurrent

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

See also **object**.

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**, **method**, **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 version 4.2 (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 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 version 4.2 software that allows WebLogic Enterprise applications to invoke CORBA objects and handle invocations from those CORBA objects (referred to as callbacks). The CORBA objects can be either inside or outside of a WebLogic Enterprise domain.

singleton object

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

skeleton

The WebLogic Enterprise Object Request Broker (ORB) component that is specific to the object interface and that assists an Object Adapter in passing requests to particular methods.

The skeleton is produced by the IDL compiler and is used at run time by the WLE ORB to invoke specific methods to satisfy requests.

See also **CORBA ORB**, **method**, and **object interface**.

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.

stateless application

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

stroid

An object ID represented as a string.

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

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

The 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**, **method**, and **See also CORBA ORB andresource managerAn 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.**.

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 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**.

`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.

UserTransaction interface

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

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

UserTransaction environmental object

The 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 an 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**, **request**, 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., such as Netscape Navigator. 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 domain

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

WLE client application

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

WLE foreign client application

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

WLE native client application

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

WLE remote client application

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

XML

Extensible Markup Language. A language written by the World Wide Web Consortium (W3C) organized by Sun Microsystems, Inc. to put SGML on the World Wide Web.