



# BEALiquid Data for WebLogic™

## Glossary

Release: 8.1  
Document Date: July 2003  
Revised: July 2003

# Copyright

Copyright © 2003 BEA Systems, Inc. All Rights Reserved.

## Restricted Rights Legend

This software and documentation is subject to and made available only pursuant to the terms of the BEA Systems License Agreement and may be used or copied only in accordance with the terms of that agreement. It is against the law to copy the software except as specifically allowed in the agreement. This document may not, in whole or in part, be copied, photocopied, reproduced, translated, or reduced to any electronic medium or machine readable form without prior consent, in writing, from BEA Systems, Inc.

Use, duplication or disclosure by the U.S. Government is subject to restrictions set forth in the BEA Systems License Agreement and in subparagraph (c)(1) of the Commercial Computer Software-Restricted Rights Clause at FAR 52.227-19; subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013, subparagraph (d) of the Commercial Computer Software--Licensing clause at NASA FAR supplement 16-52.227-86; or their equivalent.

Information in this document is subject to change without notice and does not represent a commitment on the part of BEA Systems. THE SOFTWARE AND DOCUMENTATION ARE PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND INCLUDING WITHOUT LIMITATION, ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. FURTHER, BEA Systems DOES NOT WARRANT, GUARANTEE, OR MAKE ANY REPRESENTATIONS REGARDING THE USE, OR THE RESULTS OF THE USE, OF THE SOFTWARE OR WRITTEN MATERIAL IN TERMS OF CORRECTNESS, ACCURACY, RELIABILITY, OR OTHERWISE.

## Trademarks or Service Marks

BEA, Jolt, Tuxedo, and WebLogic are registered trademarks of BEA Systems, Inc. BEA Builder, BEA Campaign Manager for WebLogic, BEA eLink, BEA Liquid Data for WebLogic, BEA Manager, BEA WebLogic Commerce Server, BEA WebLogic Enterprise, BEA WebLogic Enterprise Platform, BEA WebLogic Express, BEA WebLogic Integration, BEA WebLogic Personalization Server, BEA WebLogic Platform, BEA WebLogic Portal, BEA WebLogic Server, BEA WebLogic Workshop and How Business Becomes E-Business are trademarks of BEA Systems, Inc.

All other trademarks are the property of their respective companies.

# Glossary

## **access control list**

Used to authenticate users and manage access to network services. The WebLogic implementation of ACL is based on the `java.security.acl` package. Each entry in an ACL contains a set of permissions associated with a particular principal.

*See also* **security**.

## **ad-hoc query**

An ad-hoc query is any [hand-coded query](#) or [Builder-generated query](#) that is not stored in the BEA Liquid Data for WebLogic™ server repository but rather is passed to the Liquid Data server on the fly.

## **Administration Console**

Web-based administration tool that an administrator uses to configure and monitor WebLogic Server. Liquid Data extends the Administration Console so that administrators can configure the Liquid Data server via a Liquid Data node in the left pane.

## **Administration Server**

In each WebLogic [domain](#), one WebLogic Server instance acts as the Administration Server—the server instance that configures, manages, and monitors all other server instances and resources in the domain. In a [multi-node deployment](#) or [clustered deployment](#), the Administration Server has administrative control over the domain. In a [clustered deployment](#), the Administration Server usually hosts the [Server Repository](#). In a [standalone deployment](#), the sole WebLogic Server instance is the Administration Server by default.

## **administrative privileges**

In a Liquid Data deployment, members of the `LDAdmin` [group](#) have administrative-level access to Liquid Data tasks such as running queries, configuring data sources, managing the [Server Repository](#), designing data views and queries in the [Data View Builder](#), and assigning [access control lists](#) to Liquid Data resources.

*See also* **security**, [system administrator](#).

## **aggregation**

Query operation in which a single value is produced from a set of input values, such as the `count`, `max`, `sum`, or `average` functions.

## **analytical query**

A query that performs interim processing steps that produce [intermediate result sets](#), such as complex [joins](#) or [aggregation](#). In contrast, simple queries perform basic operations, such as retrieving a customer list or an employee's profile, without producing intermediate result sets.

## **application developer**

Application developers create applications that use the data views and stored queries, created by the [data architects](#), to access real-time information. Application developers create [Enterprise JavaBeans \(EJB\)](#), [Java Server Page \(JSP\)](#), or [Web service](#) client applications to access data views and stored queries, and they can write [custom functions](#) in Java.

*See also* **system administrator**.

## **Application Integration**

A [JCA](#) standards-based integration broker developed by BEA Systems, Inc. for inter- and intra-enterprise integration. Application integration is a component of WebLogic Integration.

## **Application View**

A business-level interface to the data in a packaged application such as Siebel, PeopleSoft, or SAP. Liquid Data allows you to dynamically access the data in these applications by treating them as *Application View* data sources. Application Views are defined and implemented primarily through the Application Integration (AI) component of Liquid Data.

## **Builder-generated query**

A query that was generated by the Data View Builder. Includes source and target schemas and associated query design tools provided by the Data View Builder.

*See also* **hand-coded query**.

**cache**

Location where Liquid Data stores information about commonly executed stored queries for subsequent, efficient retrieval, thereby enhancing overall system performance. Liquid Data provides the following kinds of caching: [query plan cache](#) and [result set cache](#).

**cache policy**

In the [result set cache](#), configuration settings that determine when the cached results expire for individual stored queries.

**clustered deployment**

Deployment architecture in which a group of servers work together to provide an application platform that is more powerful and reliable than a single server. A cluster appears to its clients as a single server but it is, in fact, a group of servers acting as one. If properly designed and configured, a cluster can provide both availability and scalability. It is possible to dynamically add new processes and machines to a cluster to handle increased load without shutting down the cluster. It is also possible to remove individual servers from the cluster, periodically, in order to perform maintenance. In a cluster, one WebLogic Server instance is the [Administration Server](#) that controls the domain, and all other WebLogic Server instances are [Managed Servers](#).

*See also* **standalone deployment**, [multi-node deployment](#).

**custom function**

User-defined functions that performed specialized tasks. The Liquid Data provides a set of standard [XQuery](#) functions for use in creating data views and queries. In addition, users can create custom functions, which are implemented in Java code, declared in a custom functions library definition (CFLD) file (in XML format), and then configured in the [Administration Console](#). Once configured, custom functions show up as functions available for use in the Data View Builder or in any [hand-coded query](#).

**custom function description**

Configuration settings, defined in the [Administration Console](#), that Liquid Data uses to find the set of [custom function](#) declarations in the custom functions library definition (CFLD) file.

**data architect**

Data architects know about the desired business entities to be created and the data sources that are required. Data architects tend to be subject matter experts and have a deep understanding of the data, underlying schema, and relationships across the various data sources. They create the data views and stored queries used by the Application Developers, using either the Data View Builder tool or creating hand-coded [XQuery](#) queries.

*See also* **application developer**, [system administrator](#).

### **data view**

Represents the result of a query that provides a filtered view on the data and information being queried. As such, the data view (query result) is dynamic—it will continue to reshape itself based on any changes that occur in the data it is querying. Data views can be re-used as data sources by saving them in the [Server Repository](#).

### **Data View Builder**

Liquid Data graphical user interface (GUI) tool for constructing valid [XQuery](#) requests on various types of data sources to be used in Liquid Data powered data integration systems and applications. Liquid Data users can use the Data View Builder drag-and-drop schemas and tools to define target schemas for query results, map source data to target schemas, define source conditions, and build and test runtime queries.

### **data source**

A source of information, such as a database or a file, against which the [Liquid Data Server](#) can query and return a result. Liquid Data can access various types of data sources including relational databases, [Web services](#), [Application Views](#), [data views](#), and XML files.

### **data source description**

A group of configuration settings that Liquid Data uses to access a particular data source. Liquid Data requires a configured data source description before it can retrieve information from the data source. You use the Data Sources tab in the Liquid Data node of the [Administration Console](#) to create, edit, and remove data source descriptions and assign [access control lists](#). The information stored in a data source description varies by [data source type](#).

### **data source type**

One of the following types of data sources that Liquid Data supports: relational databases (JDBC data sources), XML files, [Web services](#), [Application Views](#), and [data views](#).

### **deploy**

Process of installing and configuring Liquid Data components in a production environment in a WebLogic [domain](#). Deployment architectures include [standalone deployments](#), [multi-node deployments](#), and [clustered deployments](#).

**document object model (DOM)**

A W3C standard naming scheme and hierarchy for representing an XML document in a way that provides a programmable interface for dynamically accessing and updating XML. Scripts can be employed to dynamically update the content, structure, and style of an XML document.

**document type definition (DTD)**

A model for defining the structure (grammar and syntax) of a document in a markup language such as SGML, HTML, or XML. Defines what content can exist in the document. Users construct their own DTDs based on the requirements, context, and elements of their particular business arena. The DTD used determines the underlying structure and metadata of documents produced. The structure and metadata are processed and/or displayed by Web browsers and applications. DTDs are part of the [World Wide Web Consortium \(W3C\)](#) XML specification.

**domain**

A collection of servers, services, interfaces, machines, and associated resource managers defined by a single configuration file. Liquid Data can be deployed in various types of WebLogic domains, including WebLogic Platform domains, WebLogic Server domains, WebLogic Integration domains, WebLogic Portal domains, and WebLogic Workshop domains.

**EJB**

*See Enterprise JavaBeans (EJB).*

**Enterprise JavaBeans (EJB)**

Java API that defines a component architecture for multi-tier client/server systems. Specifically, the EJB specifies an architecture for the development and deployment of object-oriented, distributed, enterprise-level applications. Applications written using the EJB architecture are scalable, transactional, and secure.

**extensible markup language (XML)**

Subset of SGML that is rapidly becoming a universal standard for defining, validating, and sharing data formats and documents. Because XML is text-based (that is, it is not written in binary format), and it uses syntax rather than binary markers to organize data, it can be deployed across heterogeneous and potentially incompatible systems and platforms. Its extensibility derives from markup symbols that are unlimited and self-defining, unlike those of [HTML](#). Like HTML, XML can describe how a file is displayed. Unlike HTML, XML enables you to specify how a file is displayed. XML is a crucial component of the Liquid Data solution.

**final result set**

Result set that Liquid Data returns to the client application that submitted the query request, in contrast to an [intermediate result set](#) that the Liquid Data server create temporarily while processing an [analytical query](#).

**fixed query**

An unparameterized query.

*See also* **parameterized query**.

**group**

A set of users that share some characteristics. An [access control list](#) can assign permissions to a group. Since all permissions are positive in the WebLogic compatibility [realm](#), giving a permission to a group is the same as giving the permission to each [user](#) who is a member of the group.

*See also* **security**.

**hand-coded query**

A query coded in [XQuery](#) syntax that was not generated by the Data View Builder.

*See also* **Builder-generated query**.

**Hint**

For join operations, specifies which [join](#) algorithm to use when running a query on the [Liquid Data Server](#).

**HTML**

*See* **Hypertext Markup Language (HTML)**.

**Hypertext Markup Language (HTML)**

The set of symbols in a file that governs the format of that file when it is displayed on a World Wide Web browser.

**internationalization (I18N)**

The tailoring of a software application to the customs and languages of specific locales, without recompilation or the use of binaries other than the originals.

**intermediate result set**

Temporary result sets produced for interim processing steps when Liquid Data executes an [analytical query](#), in contrast to the [final result set](#) that Liquid Data returns to the client that requested the query.



**Internet**

The worldwide computer network of smaller, distributed networks that communicate via the TCP/IP protocols.

**JDBC**

*See* **Java Database Connectivity (JDBC)**

**JCA**

*See* **Java Connector Architecture (JCA)**

**Java Connector Architecture (JCA)**

The Java Connector Architecture defines a standard architecture for connecting the J2EE platform to heterogeneous enterprise information systems. JCA defines a set of functionality that application service vendors must provide, and which back-end system vendors (for example, SAP, PeopleSoft, Siebel, Oracle, and third-party connector developers) can use to plug in to J2EE.

**Java Database Connectivity (JDBC)**

Java Database Connectivity is a Java specification for access to relational databases. JDBC is published by Sun Microsystems, Inc. For more information, see <http://java.sun.com/products/jdbc/index.html>.

**Java Server Page (JSP)**

A J2EE component that extends the Servlet class, and allows for rapid server-side development of HTML interfaces that can be co-mingled with Java.

**Java Management Extensions (JMX)**

Basis of the Liquid Data administration API implemented through WebLogic Server. JMX, which is published by Sun Microsystems, Inc, is the standard API for management applications.

**JMX**

*See* **Java Management Extensions (JMX).**

**join**

Query operation that relates data across data sources. Typically, a join is an equality condition between two data sources, where each has a unique identifier in its own database. There are two types of join operations based on equality of matching fields (or columns):

inner joins combine data from two data sources only if values in the joined fields match, while outer joins behave like inner joins but they also include data that does not match the join condition.

## **JSP**

*See **Java Server Page (JSP)**.*

## **Liquid Data**

BEA data integration product that allows you to access and aggregate data from multiple sources in real time using the [XQuery](#) standard. Key components of Liquid Data include the [Liquid Data Server](#), [Server Repository](#), [Data View Builder](#), [Query Processor](#), and Liquid Data [Enterprise JavaBeans \(EJB\)](#) and [Java Server Page \(JSP\)](#) APIs.

## **Liquid Data Server**

Data access and aggregation server that runs as deployed application in the BEA WebLogic Server environment. The Liquid Data server accesses various information sources to process client requests. Client request are handled in the server's [Query Processor](#) engine.

## **Managed Server**

In a [multi-node deployment](#) or [clustered deployment](#), any WebLogic Server instance in the domain that is not the [Administration Server](#).

## **minus**

Query operation that returns all instances of some named value that exists in one data source but not another. For example, in an A minus B query, the query would return all instances of a named value that exists in A but no in B. The minus operation does not explicitly exist in the [XQuery](#) or Data View Builder, but it can be simulated by moving sequentially through the A data source and determining whether a named value in A is found in B.

## **multi-node deployment**

Deployment architecture that distributes Liquid Data and WebLogic Platform software components across multiple server machines. A multi-node deployment allows you to run the various software components on dedicated servers, distributing resource contention across machines and optimizing system performance. In a multi-node deployment, one WebLogic Server instance is the [Administration Server](#) that controls the domain, and all other WebLogic Server instances are [Managed Servers](#).

*See also **standalone deployment**, [clustered deployment](#).*

**packaged application**

Any type of application that is purchased rather than developed. Such applications contain reusable business processes that represent best-of-breed business models, and do not require a full-scale development effort. Examples are PeopleSoft or Siebel.

**parameterized query**

Query with parameterized content. Before a parameterized query is executed, clients must specify the type and value of each name parameter.

*See also* **fixed query**.

**query**

A request for information from one or more data sources. In Liquid Data, queries are expressed using the [World Wide Web Consortium \(W3C\) XQuery](#) standard.

*See also* [ad-hoc query](#), [stored query](#), [fixed query](#), [parameterized query](#).

**query operation**

Operation that a query performs, such as a [join](#), [aggregation](#), [union](#), or [minus](#).

**query plan**

Compiled query. Before a query is run, Liquid Data compiles the XQuery code into an executable query plan. When the query executes, the query plan is sent to the data source for processing.

**query plan cache**

Memory-based [cache](#) that stores the [query plans](#) of executed stored queries for the purpose of enhancing overall system performance.

**Query Processor**

Liquid Data Server engine that parses, re-writes, optimizes, and executes the queries received from client requests for information from multiple types of data sources.

**realm**

Domain for a set of security attributes. The realm organizes security information and defines its range of operations. A realm has its own idea of principals and permissions. Particular security domains are reflected in Java as realm instances. A realm determines how a user is authenticated and retrieves access to WebLogic resources. Liquid Data uses the WebLogic Server compatibility realm for security.

*See also* **security**.

**result set**

The data returned from an executed query. There are two types of result sets: [intermediate result sets](#) are temporary result sets that the [Query Processor](#) generate while processing an [analytical query](#), while [final result sets](#) are returned to the client application that requested the query in the form of [XML](#) data.

**result set cache**

Database-based [cache](#) that stores the results of executed queries for the purpose of enhancing overall system performance.

*See also* **cache policy**.

**role**

An organizational identity that defines a set of allowable actions for an authorized principal.

*See also* **security**.

**schema**

*See* **XML schema**

**scope**

For a query, scope helps clarify which part of a data view is the focal point for a particular condition in a query, affecting the placement of a "where" clause in the [XQuery](#) generation. In most cases, scope is implicit and the query generator can determine what the desired result should be, while in some cases the client might need to communicate their objectives explicitly, such as in the [Data View Builder](#).

**security**

Set of mechanisms available to prevent access to, corruption of, or theft of data. Liquid Data uses the WebLogic Server compatibility security mechanisms to define groups, users, and access control to Liquid Data resources.

*See also* **realm**, [access control list](#), [role](#), [group](#), [user](#).

**Server Repository**

Central file system storage for [stored query](#) files, [data views](#), [source schemas](#), [target schemas](#), [XML](#) files, [data source descriptions](#), [custom function descriptions](#), [custom function libraries](#), and generated [Web services](#). You configure the server repository using the Repository tab in the Liquid Data node of the [Administration Console](#).

## Simple Object Access Protocol (SOAP)

An extensible, platform-independent, XML-based protocol that allows disparate applications to exchange messages over the Web. SOAP can be used to invoke methods on servers, [Web services](#), application components, and objects in a distributed, heterogeneous environment. SOAP-based Web services are one of the data sources Liquid Data supports.

### source schema

XML schema that describes the shape (structure and legal elements) of the source data—that is, the data to be queried. The Liquid Data server runs queries against source data and returns the result of a query in the form in which you define for *target data*.

*See also* **target schema**, [XML schema](#).

### standalone deployment

Deployment architecture in which the Liquid Data software is installed on a single standalone WebLogic Server. This design is the simplest to set up and it is the one suggested for use in a development environment or in a production environment in which Liquid Data is the primary application and failover protection is not the top priority.

*See also* **multi-node deployment**, [clustered deployment](#).

### stored query

A query that has been saved to the Liquid Data repository. There is a performance benefit to using a stored query because its [query plan](#) is always cached in memory along with the query result. With an [ad-hoc query](#), however, the query plan and result are not cached. In addition, caching of query results for a stored query is configurable through the Cache tab on the Liquid Data node in the [Administration Console](#).

## Structured Query Language (SQL)

The standard, structured language used for communicating with relational databases. Database programmers use SQL queries to retrieve information and modify information in relational databases. In order to be able to access different types of data sources dynamically, Liquid Data employs the XML-based XQuery language as a layer on top of platform-dependent query systems such as SQL.

### system administrator

System administrators install, deploy, configure, and maintain the Liquid Data server. In addition to standard WebLogic Server administration tasks, an administrator uses the Liquid Data node in the [Administration Console](#) to perform configuration and monitoring tasks.

*See also* **application developer**, [data architect](#).

**target schema**

XML schema that describes the shape (structure and legal elements) of the output data—that is, the result of a query. The [Liquid Data Server](#) runs queries against source data and returns the result of a query the form in which you define for *target data*.

See also **source schema**, [XML schema](#).

**temporal criteria**

Query operation involving date- or time-based criteria, such as a date range or an expiration time.

**UDDI**

See **Universal Description, Discovery, and Integration (UDDI)**.

**union**

Query operation in which results are retrieved from two or more data sources but, unlike a [join](#), no condition exists across data sources. A union query is equivalent to concatenating two or more subordinate queries and pooling the query results into a single output.

**Universal Description, Discovery, and Integration (UDDI)**

Specification for distributed Web-based information registries maintained by the World Wide Web Consortium (W3C). The specification defines a method for publishing and researching information about [Web services](#). UDDI is defined primarily to support [Web services description language \(WSDL\)](#). The main component of UDDI is the UDDI business registration, an XML file that describes a business entity and its Web services. Programs and programmers use the UDDI Business Registry to locate information about services.

See also: <http://www.uddi.org>.

**user**

In WebLogic Server, an individual *principal*, which is an entity that can be authenticated by an authentication mechanism deployed in an enterprise. A principal is identified using a principal name and authenticated using authentication data. An access control list is often used to define the set of permissions within a realm. An individual's permissions override any permissions that are granted to a group of which the user is a member.

In a Liquid Data deployment, a security [role](#) designed to grant users limited access to such Liquid Data tasks as running queries, browsing data sources, and browsing the [Server Repository](#).

**value-added service**

Logic that a trading partner adds to the services provided by Liquid Data. Credit checking and shipping are examples of value-added services.

**W3C**

*See* **World Wide Web Consortium (W3C)**.

**WebLogic Server**

The platform upon which Liquid Data is built. WebLogic Server is the #1 application server on the market, providing the scalability, reliability, and security needed to run a global, networked environment—and all in one simple and extensible architecture. WebLogic Server incorporates comprehensive standards-based services for the development, deployment, and management of distributed Web-based applications.

**Web service**

Business functionality made available by one company, usually through an Internet connection, for use by another company or software program. Web services are a type of service that can be shared by, and used as components of, distributed Web-based applications. Web services communicate with clients (both end-user applications or other Web services) through XML messages that are transmitted by standard Internet protocols, such as HTTP. Web services endorse standards-based distributed computing. Currently, popular Web Service standards are [Simple Object Access Protocol \(SOAP\)](#), [Web services description language \(WSDL\)](#), and [Universal Description, Discovery, and Integration \(UDDI\)](#).

**Web services description language (WSDL)**

Specification for an XML-based grammar that defines and describes a [Web service](#). A WSDL is necessary if two different online systems need to communicate without human intervention.

**World Wide Web Consortium (W3C)**

An international software standards organization consisting of various workgroups focused on developing interoperable technologies for the Internet and World Wide Web, including XQuery and XML. For more information see <http://www.w3.org/>.

**WSDL**

*See* **Web services description language (WSDL)**.

**XQuery**

[World Wide Web Consortium \(W3C\)](#) standard XML Query language for retrieving data from multiple types of XML data sources.

**XML**

*See extensible markup language (XML)*

**XML schema**

A structured model for describing the structure, content, and semantics of XML documents based on custom rules. Replacement for DTDs. Unlike DTDs, XML schemas are written in XML data syntax and provide more support for standard data types and other data-specific features. Liquid Data users can create source and target XML schemas with the Data View Builder.