BEA Liquid Data for WebLogic™

Concepts Guide

Version 8.1
Document Date: July 2003
Revised: December 2003
Part Number: 886-002005-003

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.

Contents

About This Document

1. Introducing Liquid Data

2. Liquid Data Architecture

3. Liquid Data Standards and Features

4. Liquid Data Quick Start

Index

About This Document

This document provides an introductory overview of the BEA Liquid Data for WebLogic™ product. This document covers the following topics:

- Chapter 1, "Introducing Liquid Data," provides a general introduction to Liquid Data, including a summary of key capabilities and user roles associated with a Liquid Data implementation.
- Chapter 2, "Liquid Data Architecture," provides an overview of the Liquid Data architecture and product components.
- Chapter 3, "Liquid Data Standards and Features," introduces concepts that you should understand in order to use Liquid Data effectively.
- Chapter 4, "Liquid Data Quick Start," provides an overview of the key tasks to start using Liquid Data for data access and aggregation.

What You Need to Know

This document provides a general introduction to the BEA Liquid Data for WebLogic product. You should read it to learn the basics of using Liquid Data.

This document does not require any particular expertise, but a familiarity with the following topics helps: data integration and aggregation concepts, business and technical knowledge about the environment in which you will install and use Liquid Data, the BEA WebLogic® Platform™, and query concepts.

e-docs Web Site

BEA product documentation is available on the BEA corporate Web site. From the BEA Home page, click on Product Documentation or go directly to the "e-docs" Product Documentation page at http://e-docs.bea.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 Liquid Data 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 Liquid Data 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 in general about Java and XQuery, refer to the following sources.

- The Sun Microsystems, Inc. Java site at:

  `http://java.sun.com/`

- The World Wide Web Consortium XML Query section at:

  `http://www.w3.org/XML/Query`

For more information about BEA products, refer to the BEA documentation site at:

`http://edocs.bea.com/`

Contact Us!

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

In your e-mail message, please indicate that you are using the documentation for the BEA Liquid Data for WebLogic 1.0 release.

If you have any questions about this version of Liquid Data, or if you have problems installing and running Liquid Data, contact BEA Customer Support through BEA WebSupport at **www.bea.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.

`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`

Convention	Item
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. *Example*s: 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.

`...`	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.

CHAPTER 1

Introducing Liquid Data

This chapter provides a high level overview of BEA Liquid Data for WebLogic. It contains the following sections:

- What is BEA Liquid Data for WebLogic?
- Liquid Data Capabilities
- Integration with WebLogic Workshop

What is BEA Liquid Data for WebLogic?

BEA Liquid Data for WebLogic is a data access and aggregation product for *Information Visibility*, allowing a real-time unified view of disparate enterprise data. Information visibility is essential to enable greater supply-chain visibility, a unified view of customer information, and better decision-making.

BEA Liquid Data for WebLogic provides a cost effective, standard way to rapidly aggregate and expose logical views from any number of heterogeneous sources (including Web services, databases, flat files, XML files, applications and Web sites). This enables developers to re-use information across applications without dealing with the complexity of the underlying data.

One of the greatest challenges for IT organizations is that today's technologies do not adequately address the complexity inherent in aggregating data from distributed systems. BEA Liquid Data for WebLogic allows IT departments to easily aggregate data, in real time, from many sources within and outside of the enterprise, and tailor it for different business users, providing information visibility anywhere in the enterprise.

BEA Liquid Data for WebLogic is an easy-to-use product that simplifies the development of applications and portals that require aggregated, real-time information from disparate or heterogeneous sources. BEA Liquid Data for WebLogic returns results in XML for easy consumption by Web applications.

BEA Liquid Data for WebLogic delivers flexible, reusable information without the high cost and time consuming custom coding of queries and data transformations required with other products and solutions. Developers access BEA Liquid Data for WebLogic as simple Business Information Services created and published by a company's domain or data architects. Additionally, developers have the power and flexibility of XQuery, a standards-based query language for XML documents, to directly access the distributed data sources, which are represented as integrated logical views.

BEA Liquid Data for WebLogic builds on existing IT infrastructure and leverages XML standards and BEA WebLogic Platform—the industry-leading application infrastructure—for easy and secure deployment in existing IT environments. BEA Liquid Data for WebLogic can be deployed as a standalone solution or within the framework of a larger project.

Liquid Data Capabilities

BEA Liquid Data for WebLogic provides the following capabilities:

Real Time Data Access

- **Universal data access**—Using XML translators and optimized XML queries, BEA Liquid Data for WebLogic can retrieve data from legacy and client-server applications, enterprise applications, relational databases, Web Services, flat files and XML files, and other data sources, both inside and outside of corporate firewalls. BEA Liquid Data for WebLogic returns results in XML for easy consumption by Web applications.
- **Abstract Data Views**—A virtual abstraction layer aggregates distributed data sources as an integrated, logical view. For developers, these logical views of aggregated data sets—or data views—can be thought of as a single, virtual database. Data views allow easy access to shared business entities without needing to deal with the underlying complexity of varying data structures, semantics, and access methods.

Share Corporate Assets

- **Reusable Views**—Information about customers, orders, and inventory is needed again and again, by a variety of groups within the organization. BEA Liquid Data for WebLogic enables views of data to be presented as services that can be customized and reused. It makes it easy to modify views when new data sources need to be added.
- **Leverage domain resources**—Data views are created by a organization's data domain expert or data architect, who can quickly examine source schemas, and identify needed columns or elements. Data views are created and published once and shared many times by developers accessing data views as simple services, making the role of data access completely transparent to the developer.

Increase Developer Productivity

- **Expose and share Web Services**—Data views can be shared as Web services with other applications and departments by creating and sharing corporate entities (like customer or product profiles). The information is automatically tagged and organized for presentation by Web applications and can be invoked by a Web service or embedded rapidly and easily within an integrated application.
- **Dramatically reduce coding**—By providing developers with a high-level query language and visual development tools, BEA Liquid Data for WebLogic replaces months of custom coding with simple, familiar queries. Developers have the power and flexibility of a standard SQL-like language called XQuery for capabilities that can be written as familiar database queries, in only a few minutes.
- **Integrate with the WebLogic Workshop development environment**—By providing developers with a Liquid Data control to use within WebLogic Workshop, you can easily use all of the WebLogic Workshop and platform resources (NetUI, XMLBeans, WebLogic Portal, and so on) to rapidly develop industrial strength application that display data from Liquid Data, as well as provide any other functionality available through the WebLogic Platform.

Standards-Based

BEA Liquid Data for WebLogic takes full advantage of BEA WebLogic Server, leveraging all the J2EE and XML standards, including new XML standards like XQuery. Adherence to these standards protects your investment.

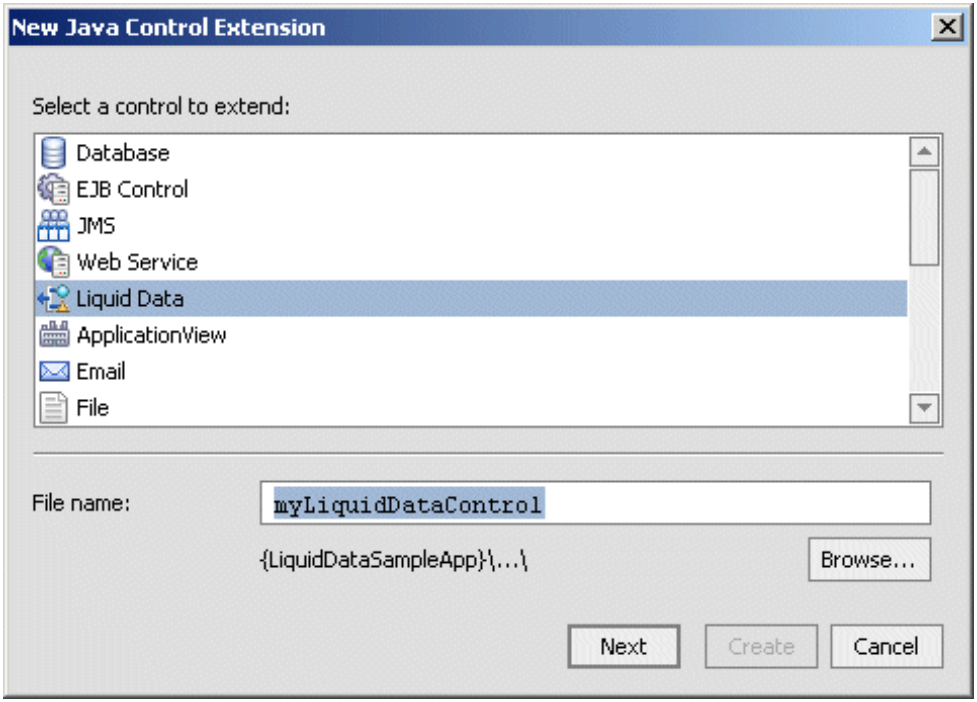
Integration with WebLogic Workshop

BEA Liquid Data for WebLogic includes a Java Control in WebLogic Workshop. The Liquid Data control provides access to stored queries and ad-hoc queries in Liquid Data, and it generates `XMLBean` classes which represent the data results sets of your queries. The `XMLBean` classes allows application developers to treat data from Liquid Data exactly as they treat data from other data providers.

Developers can take advantage of the distributed data access features of Liquid Data in conjunction with the other powerful capabilities available in the WebLogic Enterprise Platform. For example, developers can use Liquid Data with NetUI to develop web applications and portals. Also, developers can create WebLogic Integration workflows that use data from Liquid Data queries delivered by Liquid Data Controls.

After you install Liquid Data, the Liquid Data control is available with all of the other Java Controls in the WebLogic Workshop IDE.

Figure 1-1 Liquid Data Control in WebLogic Workshop



Developers use the Liquid Data control wizard, which guides them through the process of creating a control to access their queries. Once the wizard is completed, the control is automatically created (without writing any code).

For details about using WebLogic Workshop to develop Liquid Data applications, see the Liquid Data *Application Developer's Guide*.

CHAPTER 2

Liquid Data Architecture

This chapter provides an overview of the BEA Liquid Data for WebLogic architecture and describes the different roles for Liquid Data users. It includes the following sections:

- Liquid Data Architecture Components
- Liquid Data and the WebLogic Platform
- Liquid Data Roles and Responsibilities

Liquid Data Architecture Components

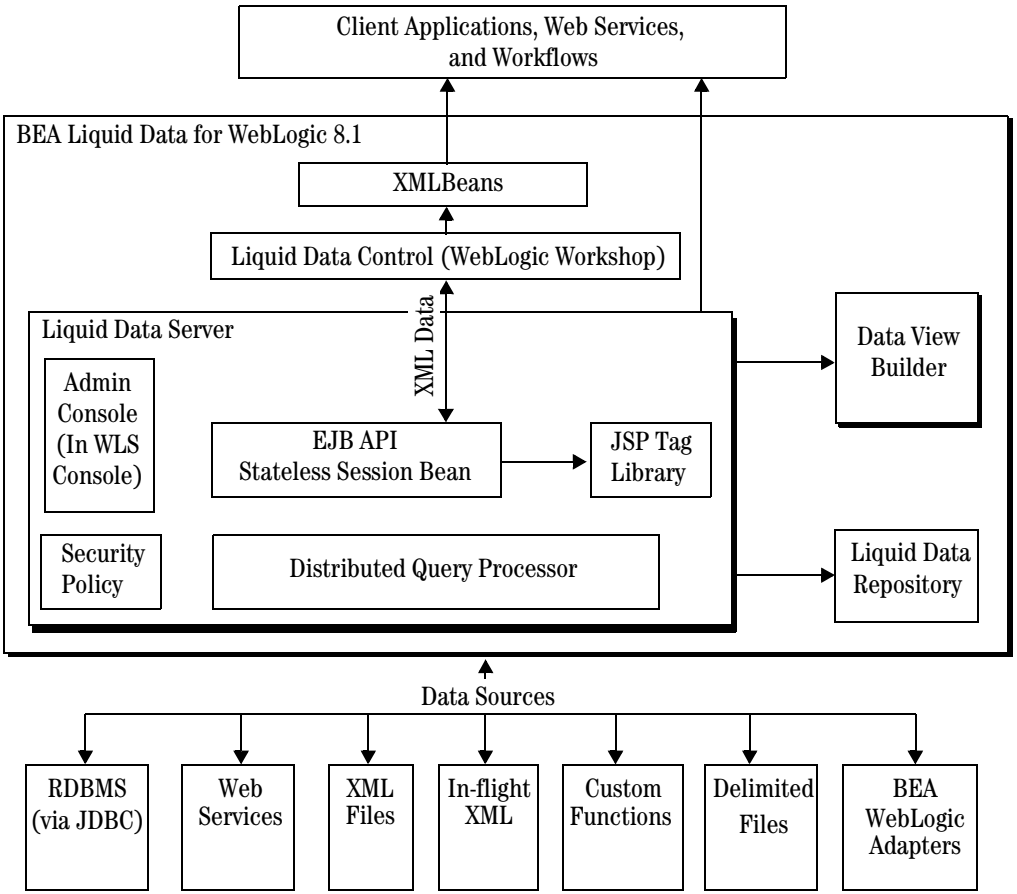
This section describes the Liquid Data architecture and contains the following subsections:

- Liquid Data Components Architecture Diagram
- Liquid Data Components

Liquid Data Components Architecture Diagram

The following diagram shows the components of Liquid Data. For a diagram showing Liquid Data and the rest of the WebLogic Platform, see Figure 2-5.

Figure 2-1 Liquid Data Components Architecture



Liquid Data Components

This section describes the main components of Liquid Data:

- Liquid Data Server and Distributed Query Processor
- Liquid Data Repository
- Data View Builder
- Administration Console
- Liquid Data Control in WebLogic Workshop
- Liquid Data Query API
- Custom Functions

Liquid Data Server and Distributed Query Processor

The Liquid Data Server runs as an application in WebLogic Server. The Liquid Data Server processes query requests from Liquid Data client applications. Liquid Data queries are written in the W3C standard XQuery language. The Liquid Data server handles incoming query requests, and then the distributed query processor translates the XQuery into an optimized query plan and executes queries against the underlying data sources. The Liquid Data server then combines the results from the underlying data sources to form a single XML result, which is returned to the client application.

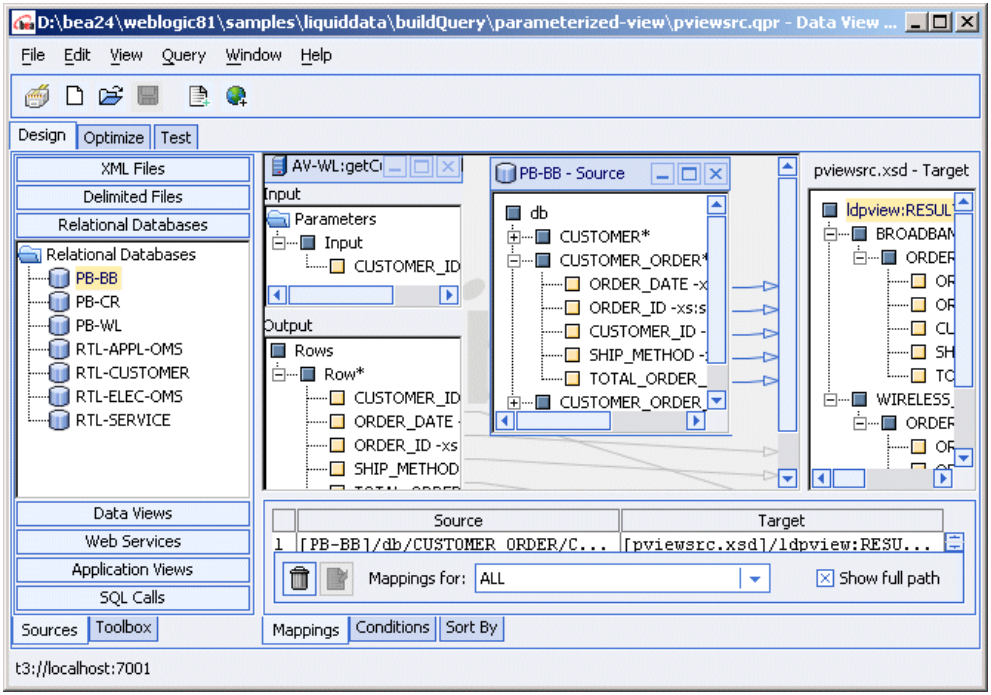
Liquid Data Repository

The Liquid Data repository is the central location for stored queries, data views, source and target schemas, XML data, web service descriptions (WSDL files), custom function libraries, stored procedures and other SQL queries, and delimited files. For more information, see "Managing the Liquid Data Repository" in the Liquid Data *Administration Guide*.

Data View Builder

Data View Builder is a GUI tool for designing, generating, testing, and deploying Liquid Data queries. Data View Builder provides a drag-and-drop paradigm which allows query builders to easily combine data sources, functions, web services, and other query components together by connecting elements to each other and projecting the results onto a target schema. Data View Builder generates queries in the XQuery language and allows users to save queries to the Liquid Data repository. Once the queries are saved to the repository, client applications can run the queries (via a variety of interfaces), accessing the data in the underlying heterogeneous data sources.

Figure 2-2 Data View Builder



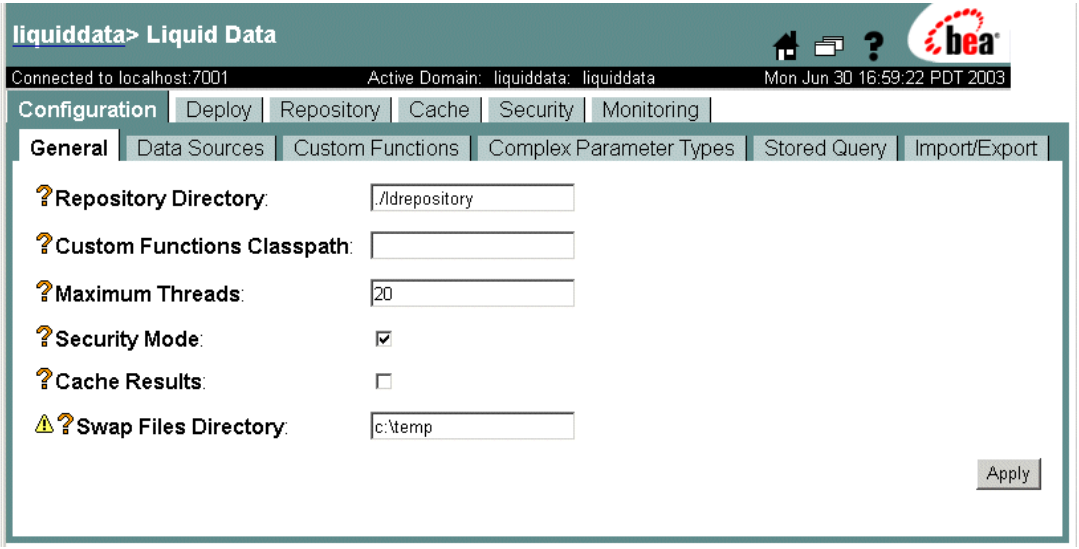
Data View Builder generates queries in the XQuery language based on the mappings, conditions, and joins defined in the design view. It also allows query builders to run, modify, and deploy the generated queries to the Liquid Data repository. Data Architects need not know the intricacies of the XQuery language to use Data View Builder and can focus instead on the data they want to integrate.

You can use the Data View Builder to create and deploy data views, which are queries that can be used as a data sources for other queries. Data views provide a logical view of the data, abstracting all of the details of the underlying data sources from the application developer. Developers can then easily create queries for their applications without requiring any knowledge about the specific underlying systems. In this way, application developers, not just data architects, can directly access distributed, heterogeneous data sources. For more information about the Data View Builder, see *Building Queries and Data Views*.

Administration Console

The WebLogic Server Administration Console is a comprehensive, Web-based administration tool for configuring and monitoring WebLogic servers. Liquid Data extends the WebLogic Server Administration Console to provide additional tabs, accessible via a Liquid Data node, that are used to configure and manage Liquid Data servers deployed in the WebLogic Server environment.

Figure 2-3 Tabs in the Liquid Data Node in the Administration Console



System Administrators use the tabs on the Liquid Data node to configure access to data sources and custom functions, manage the server repository, configure query results caching, implement security, generate Web services, and so on. For more information, see the Liquid Data *Administration Guide*.

Liquid Data Control in WebLogic Workshop

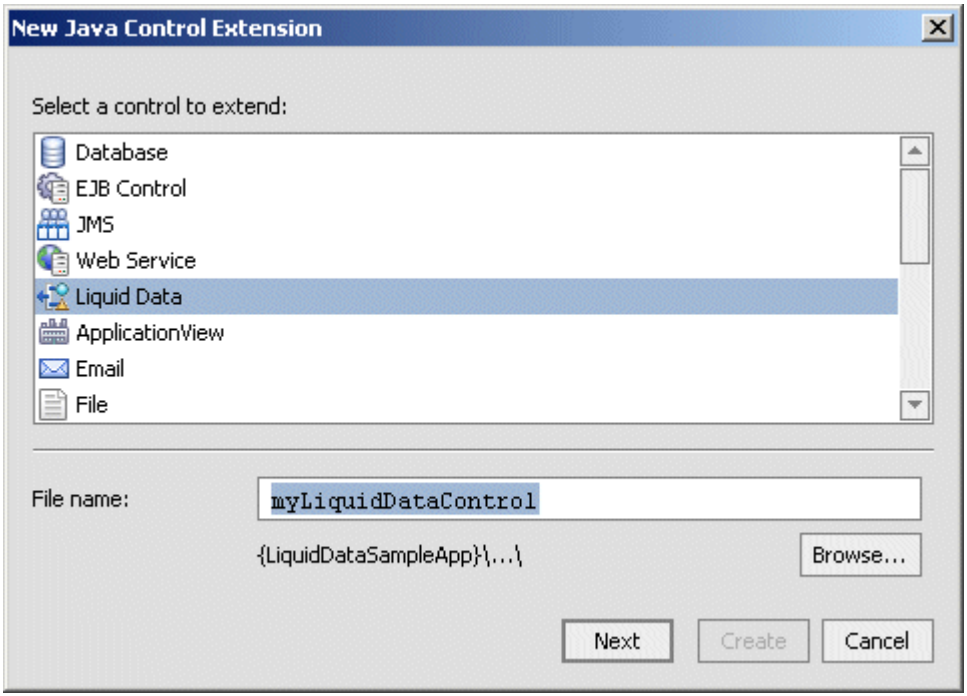
Liquid Data includes a Java Control Extension for WebLogic Workshop. The Liquid Data control provides access to Liquid Data queries directly through the WebLogic Workshop IDE. The Liquid Data control generates `XMLBean` classes which allow developers to easily access and manipulate the result set of the queries accessed from the control.

Developers can then use the Liquid Data Control in conjunction with other controls and features of WebLogic Workshop to develop complex applications. For example, developers can use the Liquid Data Control and NetUI to create web applications that access distributed data. Developers can also create service oriented applications where Liquid Data provides data integration services.

WebLogic Workshop, with its controls architecture, together with the Liquid Data Control provide a seamless developer experience for building portals, web services, and business processes.

The Liquid Data control appears with all of the other built-in and custom controls in WebLogic Workshop.

Figure 2-4 Liquid Data Control in WebLogic Workshop



For details about using WebLogic Workshop to develop Liquid Data applications, see the Liquid Data *Application Developer's Guide*.

Liquid Data Query API

Liquid Data provides an API (application programming interface) that allows client Java applications to submit queries to the Liquid Data for processing. Liquid Data supports three types of clients:

- Applications developed using WebLogic Workshop can use the Liquid Data control to access queries (for more details, see Liquid Data Control in WebLogic Workshop).

- EJB clients can invoke queries on Liquid Data query EJBs, as described in "Invoking Queries in EJB Clients" in the *Application Developer's Guide*.
- JSP (Java Server Pages) clients can invoke queries using the Liquid Data JSP tag library, as described in "Invoking Queries in JSP Clients" in the *Application Developer's Guide*.

In addition to custom client applications, the Liquid Data Query API allows other BEA products—WebLogic Portal, WebLogic Workshop, WebLogic Web services, and the Business Process Management component of WebLogic Integration—to invoke Liquid Data queries transparently.

Custom Functions

Liquid Data provides a set of standard functions to use when creating data views and queries. Application Developers can also define *custom function*s to extend the power and functionality of Liquid Data. Queries can invoke custom functions during query execution just as they can standard functions. For more information, see "Creating Custom Functions" in the *Application Developer's Guide*.

Liquid Data and the WebLogic Platform

Liquid Data is part of the WebLogic Platform 8.1. It takes advantage of the advanced, enterprise-class capabilities of WebLogic Server to deliver scalable, reliable, and highly available distributed data access services. Liquid Data provides complementary functionality to WebLogic Integration, and allows you to use WebLogic Portal to add applications that use Liquid Data to display disparate data into your existing portal infrastructure. Liquid Data also includes a Java Control Extension for WebLogic Workshop, providing a strong integration with the WebLogic Workshop integrated development environment (IDE).

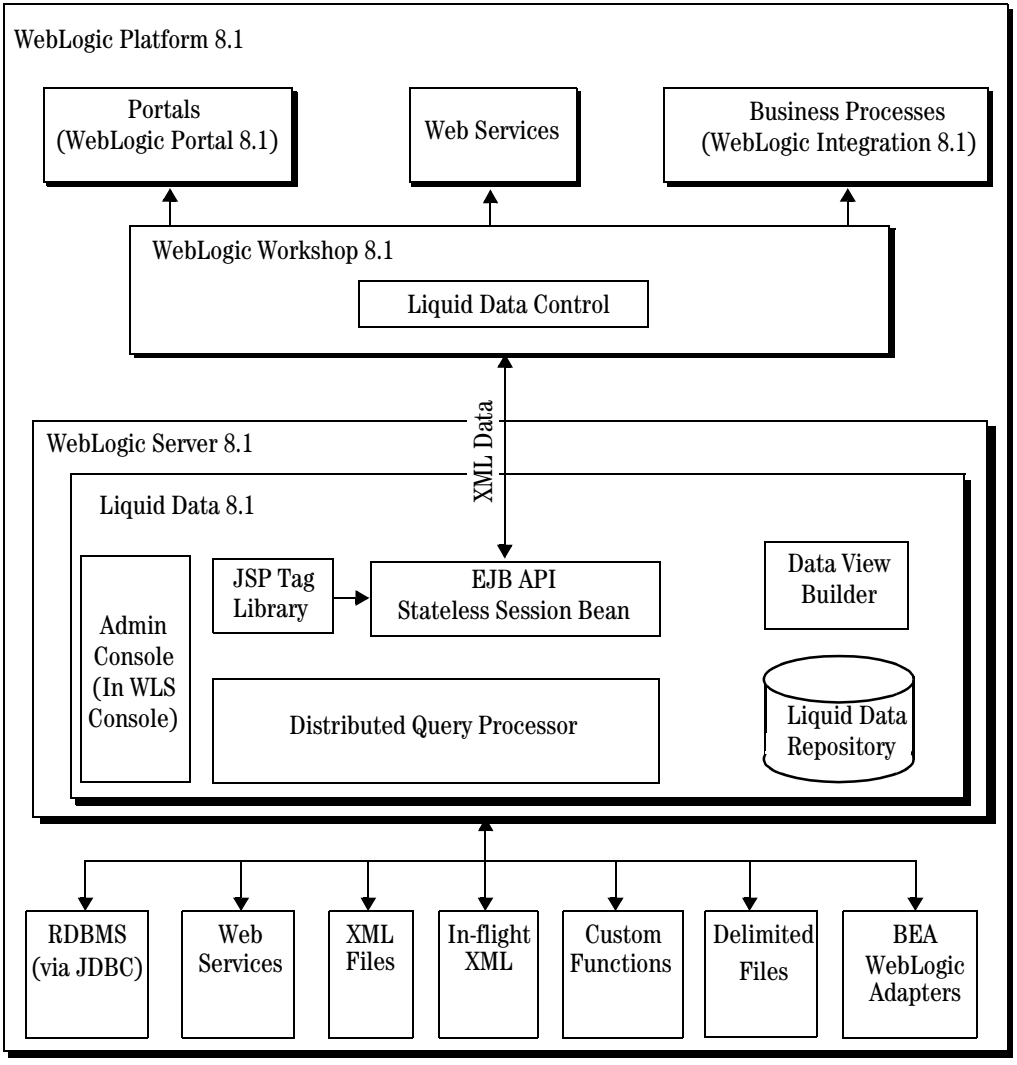
This section describes the Liquid Data architecture and how it fits in with the WebLogic Platform 8.1. The following sections are included.

- Liquid Data and WebLogic Platform Architecture Diagram
- Runs Within WebLogic Server
- Integrates With WebLogic Integration
- Integrates With WebLogic Portal
- Application Development With WebLogic Workshop

Liquid Data and WebLogic Platform Architecture Diagram

The following diagram shows the components of the Liquid Data architecture and how Liquid Data fits in the WebLogic Platform architecture.

Figure 2-5 Liquid Data and WebLogic Platform 8.1 Architecture



Runs Within WebLogic Server

Liquid Data is deployed as an enterprise application within WebLogic Server. It therefore takes advantage of all the services available to WebLogic Server, such as JDBC Connection Pools and Data Sources, scalability and clustering, and the full array of J2EE features and services. It operates seamlessly with other applications running in WebLogic Server.

The core of the Liquid Data server is a sophisticated distributed query processor that divides user requests for information into optimized sub-queries, which in turn are processed against many data sources. Liquid Data supports data aggregation from most enterprise data sources, including relational databases, web services, enterprise application and mainframe resources (through WebLogic Adapters) files such as XML files and delimited files, and to any other custom data source through custom functions.

For comprehensive information about BEA WebLogic Server, see the WebLogic Server documentation at the following URL:

http://edocs.bea.com/wls/docs81/index.html

Integrates With WebLogic Integration

WebLogic Integration is a solution delivering application integration, business process management, and B2B integration functionality for the enterprise.

Complex business processes and application integration projects usually involve complex data lookups. Examples of these include enrichment of a incoming purchase order with related information from multiple enterprise data sources, delivering supporting information about a customer to a supervisor to help them make better decisions while approving a purchase order in a worklist, and so on.

Liquid Data simplifies complex data aggregation in integration projects. Through the Liquid Data control, developers can now perform an efficient, single-step lookup of distributed data instead of having to write complex workflows that embody the essence of a distributed query processor. Liquid Data provides essential data integration capabilities that, when used in conjunction with WebLogic Integration, reduce the cost and complexity of enterprise integration.

For comprehensive information about BEA WebLogic Integration, see the WebLogic Integration documentation at the following URL:

http://edocs.bea.com/wli/docs81/index.html

For more information about Liquid Data and WebLogic Integration, see "Invoking Queries in Business Process Manager Applications" in the *Application Developer's Guide* and "Deploying Liquid Data in a WebLogic Integration Domain" in "Deployment Tasks" in the Liquid Data *Deployment Guide*.

Integrates With WebLogic Portal

Portals provide a single point of access to enterprise information, applications and processes, personalized to fit the role and needs of the end-user. Portals are also considered to be the "business face" to enterprise information. One of the most challenging aspects of providing a business face to enterprise information is the ability to access related data from distributed, heterogeneous data sources and delivering them to numerous portals and audiences.

Liquid Data provides a cost-effective data abstraction layer that reduces the cost, complexity and time-to-value of portal development. Liquid Data hides the complexity and specificity of underlying data sources and provides a simple API through which developers can access distributed data. Consequently, developers can focus on *what* information they need and leave the *how* to access and aggregate that information to Liquid Data. Through its repository and reusable views, Liquid Data also promotes reuse and consistency in how information is dealt with across various portal initiatives.

Developers can use the Liquid Data control to access distributed information and convert the results into personalized portlets in a single, seamless development environment—WebLogic Workshop. For more information, see "Invoking Queries in BEA WebLogic Portal Applications" in the *Application Developer's Guide* and "Deploying Liquid Data in a WebLogic Portal Domain" in "Deployment Tasks" in the *Deployment Guide*.

Application Development With WebLogic Workshop

WebLogic Workshop is an IDE and development framework designed to simplify J2EE application development. WebLogic Workshop provides an abstraction model for enterprise resources through controls. Controls provide a simple, convenient mechanism by which to develop enterprise class applications. Liquid Data provides a control for use in the WebLogic Workshop development environment.

Enterprises looking to develop complex, service-oriented applications will find Workshop to be a convenient and cost effective development environment. Liquid Data is an integral part of any service-oriented architecture providing a virtual data access layer that hides the complexity and specificity of underlying data sources. Developers can now program to a simple API that enables the access and aggregation of data from multiple, heterogeneous data sources. The Liquid Data control generates an `XMLBean` interface to distributed data. This can be used in conjunction with other powerful features of Workshop including NetUI to easily deliver complex applications.

Liquid Data Roles and Responsibilities

This section describes the roles, responsibilities, and resources for the following types of Liquid Data users:

- Data Architects
- Application Developers
- System Administrators

Each type of user has different tasks and tools for using the Liquid Data technology.

Data Architects

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. For more information, see *Building Queries and Data Views*.

Application Developers

Application Developers create applications that use the data views and stored queries, created by the Data Architects, to access real-time information. Application Developers can access data views and stored queries using the following mechanisms:

- Integration with WebLogic Workshop with the Liquid Data control.
- Invoking queries in EJB client applications using the Liquid Data Query API.
- Invoking queries in JSP (Java Server Pages) clients using the Liquid Data JSP Tag Library
- Invoking queries that have been published as Web services

Application Developers can also write custom functions in Java code to extend Liquid Data functionality.

For more information, see *Application Developer's Guide*.

System Administrators

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 the following kinds of tasks:

- Deploy Liquid Data in WebLogic domains.
- Configure access to data sources and queries.
- Set up and configure the Liquid Data server repository.
- Implement security using the extensive WebLogic Server security features, such as configuring users, groups, and defining security roles.
- Configure query caching to optimize performance.
- Monitor Liquid Data server operation, tune performance, and provide support.

For more information, see the Liquid Data *Administration Guide* and *Deployment Guide*.

CHAPTER 3

Liquid Data Standards and Features

This section provides an overview of core BEA Liquid Data for WebLogic features and also some usage information. It contains the following sections:

- Liquid Data Implements the XQuery Standard
- Definitions of Key Terms

For an overview of Liquid Data components, see Chapter 2, "Liquid Data Architecture."

Liquid Data Implements the XQuery Standard

This topic describes XQuery and how Liquid Data implements the XQuery standard. It contains the following sections:

- About XQuery
- XQuery Links
- XQuery Example

Liquid Data provides a scalable, highly available platform for building and processing queries written using XQuery. Data Architects can use the Data View Builder tool to create XQuery queries graphically (without needing to hand write queries), as well as create ad hoc XQuery queries by hand. Client applications can use the Liquid Data Query API or generated Liquid Data Web services to invoke stored XQuery queries programmatically.

About XQuery

XQuery is a standard query language, published by the W3C (World Wide Web Consortium), that uses XML (Extensible Markup Language) notation to define query requests and handle query results. XQuery is designed to be an efficient language in which queries are concise and easily understood. XQuery is flexible enough to query a broad spectrum of data sources, including relational databases, XML documents, Web services, packaged applications, and legacy systems.

Applications of XQuery include filtering a document to produce a table of contents, providing joins across multiple data sources, grouping and aggregates, and queries based on sequential relationships in documents. Joins, in particular, represent a powerful and likely principal use of XQuery in the enterprise. The XQuery language is derived from various sources, including SQL. Developers familiar with SQL will find XQuery very easy to learn.

XQuery Links

For more information about XQuery, see the following topics on the W3C's Web site:

- XQuery 1.0: An XML Query Language. Liquid Data supports the December 20, 2001 working draft at the following URL:

  http://www.w3.org/TR/2001/WD-xquery-20011220/

- XQuery 1.0 and XPath 2.0 Functions and Operators. Liquid Data supports the April 30, 2002 working draft at the following URL:

  `http://www.w3.org/TR/2002/WD-xquery-operators-20020430/`

- XML Query Use Cases at the following URL:

  `http://www.w3.org/TR/xmlquery-use-cases`

- XML Path Language (XPath) 2.0 at the following URL:

  `http://www.w3.org/TR/xpath20/`

- XQuery 1.0 and XPath 2.0 Data Model at the following URL:

  `http://www.w3.org/TR/query-datamodel/`

For details of the XQuery functions (both standard and extensions to the standard) implemented in Liquid Data, see "Functions Reference" in *XQuery Reference Guide*.

XQuery Example

This section shows an example of XQuery code and sample query results from the Retail Sample Application. For more information about the Retail Sample Application and other samples included with Liquid Data, see *Liquid Data by Example*.

Sample XQuery Query

The following code example shows the use XQuery for the Customer Order query in the sample application. This query retrieves customer information out of several data sources and combines the results to make them available for the application. For details on the sample application shipped with Liquid Data, see "Overview of the Avitek Self-Service Sample Application" in *Liquid Data by Example*.

Listing 3-1 XQuery for the Order Query from the Avitek Sample

```
{--      Generated by Data View Builder 8.1--}

namespace ElectOrderService = "http://www.bea.com//examples/ldi/webservice/customerOrder"
namespace retailerType = "urn:retailerType"
namespace elecOrd = "java:examples.ldi.webservice"
namespace retailer = "urn:retailer"

<retailer:CustomerView>
  {
  for $RTL_CUSTOMER.CUSTOMER_1 in document("RTL-CUSTOMER")/db/CUSTOMER
  where ($#custid of type xs:string  eq  $RTL_CUSTOMER.CUSTOMER_1/CUSTOMER_ID)
  return
  <CUSTOMER_VIEW>
    <CUSTOMER_ID>{ xf:data($RTL_CUSTOMER.CUSTOMER_1/CUSTOMER_ID)}
    </CUSTOMER_ID>
    <FIRST_NAME>{ xf:data($RTL_CUSTOMER.CUSTOMER_1/FIRST_NAME) }</FIRST_NAME>
    <LAST_NAME>{ xf:data($RTL_CUSTOMER.CUSTOMER_1/LAST_NAME) }</LAST_NAME>
    <ORDERS>
      {
      for $RTL_APPL_OMS.CUSTOMER_ORDER_2 in document
        ("RTL-APPL-OMS")/db/CUSTOMER_ORDER
      where ($RTL_CUSTOMER.CUSTOMER_1/CUSTOMER_ID  eq
        $RTL_APPL_OMS.CUSTOMER_ORDER_2/CUSTOMER_ID)
          and ($RTL_APPL_OMS.CUSTOMER_ORDER_2/STATUS  eq  "OPEN")
      return
      <retailerType:ORDER_SUMMARY TYPE={"APPL"}>
        <ORDER_ID>{ xf:data($RTL_APPL_OMS.CUSTOMER_ORDER_2/ORDER_ID) }</ORDER_ID>
        <ORDER_DATE>{ xf:data($RTL_APPL_OMS.CUSTOMER_ORDER_2/ORDER_DATE) }</ORDER_DATE>
        <TOTAL_ORDER_AMOUNT>{ xf:data($RTL_APPL_OMS.CUSTOMER_ORDER_2/TOTAL_ORDER_AMOUNT) }
        </TOTAL_ORDER_AMOUNT>
        <SHIP_TO_NAME>{ xf:data($RTL_APPL_OMS.CUSTOMER_ORDER_2/SHIP_TO_NAME) }
        </SHIP_TO_NAME>
```

```
        <ESTIMATED_SHIP_DATE>{ xf:data($RTL_APPL_OMS.CUSTOMER_ORDER_2/ESTIMATED_SHIP_DATE) }
        </ESTIMATED_SHIP_DATE>
        <TRACKING_NUMBER>{ xf:data($RTL_APPL_OMS.CUSTOMER_ORDER_2/TRACKING_NUMBER) }
        </TRACKING_NUMBER>
        {
        for $RTL_APPL_OMS.CUSTOMER_ORDER_LINE_ITEM_3 in document("RTL-APPL-OMS")
          /db/CUSTOMER_ORDER_LINE_ITEM
        where ($RTL_APPL_OMS.CUSTOMER_ORDER_2/ORDER_ID  eq
          $RTL_APPL_OMS.CUSTOMER_ORDER_LINE_ITEM_3/ORDER_ID)
        return
        <LINE_ITEM>
          <PRODUCT_DESC>{ xf:data($RTL_APPL_OMS.CUSTOMER_ORDER_LINE_ITEM_3/PRODUCT_DESC) }
          </PRODUCT_DESC>
      <QUANTITY>{ cast as xs:int(xf:data($RTL_APPL_OMS.CUSTOMER_ORDER_LINE_ITEM_3/QUANTITY)}
    </QUANTITY>
        </LINE_ITEM>
        }
      </retailerType:ORDER_SUMMARY>
      }
      {
      for $ElectOrderService:getOrderSummaryByStatus.__5 in
      ElectOrderService:getOrderSummaryByStatus($#custid of type xs:string, "OPEN")/result/*
      let $xfext:date_from_dateTime2_7 :=
xfext:date-from-dateTime($ElectOrderService:getOrderSummaryByStatus.__5/elecOrd:orderDate)
      let $xfext:date_to_string_with_format2_8 :=
        xfext:date-to-string-with-format("yyyy-MM-dd",  treat as
          xs:date($xfext:date_from_dateTime2_7))
      let $xfext:date_from_dateTime_11 :=
        xfext:date-from-dateTime
          ($ElectOrderService:getOrderSummaryByStatus.__5/elecOrd:estimatedShipDate)
      let $xfext:date_to_string_with_format_12 :=
        xfext:date-to-string-with-format("yyyy-MM-dd",  treat as
          xs:date($xfext:date_from_dateTime_11))
      return
      <retailerType:ORDER_SUMMARY TYPE={"ELEC"}>
        <ORDER_ID>{ xf:data($ElectOrderService:getOrderSummaryByStatus.__5/elecOrd:orderID)
                  }
        </ORDER_ID>
        <ORDER_DATE>{  treat as xs:date(xfext:date-from-string-with-format("yyyy-MM-dd",
            $xfext:date_to_string_with_format2_8)) }
        </ORDER_DATE>
        <TOTAL_ORDER_AMOUNT>{ xf:data
            ($ElectOrderService:getOrderSummaryByStatus.__5/elecOrd:totalOrderAmount) }
        </TOTAL_ORDER_AMOUNT>
        <SHIP_TO_NAME>{ xf:data
          ($ElectOrderService:getOrderSummaryByStatus.__5/elecOrd:shipToName) }
        </SHIP_TO_NAME>
        <ESTIMATED_SHIP_DATE>{  treat as xs:date(xfext:date-from-string-with-format
          ("yyyy-MM-dd", $xfext:date_to_string_with_format_12)) }
        </ESTIMATED_SHIP_DATE>
        <TRACKING_NUMBER>{ xf:data
          ($ElectOrderService:getOrderSummaryByStatus.__5/elecOrd:trackingNumber) }
        </TRACKING_NUMBER>
        {
```

```
        for $ElectOrderService:getOrderSummaryByStatus.__15 in
            $ElectOrderService:getOrderSummaryByStatus.__5/elecOrd:items/*
        return
        <LINE_ITEM>
          <PRODUCT_DESC>{
            xf:data($ElectOrderService:getOrderSummaryByStatus.__15/elecOrd:productDesc) }
          </PRODUCT_DESC>
          <QUANTITY>{
            xf:data($ElectOrderService:getOrderSummaryByStatus.__15/elecOrd:quantity) }
          </QUANTITY>
        </LINE_ITEM>
        }
      </retailerType:ORDER_SUMMARY>
      }
    </ORDERS>
                <CASES>
      {
      for $RTL_SERVICE.SERVICE_CASE_16 in document("RTL-SERVICE")/db/SERVICE_CASE
      where ($RTL_SERVICE.SERVICE_CASE_16/CUSTOMER_ID  eq
              $RTL_CUSTOMER.CUSTOMER_1/CUSTOMER_ID)
        and ($RTL_SERVICE.SERVICE_CASE_16/STATUS  eq  "OPEN")
      return
      <CASE>
        <CASE_ID>{ xf:data($RTL_SERVICE.SERVICE_CASE_16/CASE_ID) }</CASE_ID>
        <CASE_TYPE>{ xf:data($RTL_SERVICE.SERVICE_CASE_16/CASE_TYPE) }</CASE_TYPE>
        <PRODUCT_ID>{ xf:data($RTL_SERVICE.SERVICE_CASE_16/PRODUCT_ID) }</PRODUCT_ID>
        <STATUS>{ xf:data($RTL_SERVICE.SERVICE_CASE_16/STATUS) }</STATUS>
        <STATUS_DATE>{ xf:data($RTL_SERVICE.SERVICE_CASE_16/STATUS_DATE) }</STATUS_DATE>
      </CASE>
      }
    </CASES>
  </CUSTOMER_VIEW>
  }
</retailer:CustomerView>
```

Sample Query Results

The following listing shows the results returned in XML format from the Customer View query where the customer ID parameter is Homer.

Listing 3-2 Results for the Order Query from the Avitek Sample

```
<prefix1:CustomerView xmlns:prefix1="urn:retailer">
  <CUSTOMER_VIEW>
    <CUSTOMER_ID>Homer</CUSTOMER_ID>
    <FIRST_NAME>Homer</FIRST_NAME>
    <LAST_NAME>Simpson</LAST_NAME>
```

```
<ORDERS>
  <prefix2:ORDER_SUMMARY TYPE="APPL" xmlns:prefix2="urn:retailerType">
    <ORDER_ID>ORDER_9_2</ORDER_ID>
    <ORDER_DATE>2001-12-17</ORDER_DATE>
    <TOTAL_ORDER_AMOUNT>134.55</TOTAL_ORDER_AMOUNT>
    <SHIP_TO_NAME>Marge Simpson</SHIP_TO_NAME>
    <ESTIMATED_SHIP_DATE>2001-12-20</ESTIMATED_SHIP_DATE>
    <TRACKING_NUMBER>ORDER_9_28925737231</TRACKING_NUMBER>
    <LINE_ITEM>
     <PRODUCT_DESC>Osh Kosh Lt Lilac Poplin Jumper Dress</PRODUCT_DESC>
      <QUANTITY>3</QUANTITY>
    </LINE_ITEM>
    <LINE_ITEM>
      <PRODUCT_DESC>Guess Garden Denim Skirt</PRODUCT_DESC>
      <QUANTITY>1</QUANTITY>
    </LINE_ITEM>
    <LINE_ITEM>
      <PRODUCT_DESC>Gap denim front-slit skirt</PRODUCT_DESC>
      <QUANTITY>1</QUANTITY>
    </LINE_ITEM>
  </prefix2:ORDER_SUMMARY>
  <prefix2:ORDER_SUMMARY TYPE="APPL" xmlns:prefix2="urn:retailerType">
    <ORDER_ID>ORDER_9_7</ORDER_ID>
    <ORDER_DATE>2002-06-29</ORDER_DATE>
    <TOTAL_ORDER_AMOUNT>221.55</TOTAL_ORDER_AMOUNT>
    <SHIP_TO_NAME>Homer Simpson</SHIP_TO_NAME>
    <ESTIMATED_SHIP_DATE>2002-07-06</ESTIMATED_SHIP_DATE>
    <TRACKING_NUMBER>ORDER_9_77132565448</TRACKING_NUMBER>
    <LINE_ITEM>
      <PRODUCT_DESC>Kenneth Cole Reaction Broadcloth Fancy Dress Shirt
      </PRODUCT_DESC>
      <QUANTITY>3</QUANTITY>
    </LINE_ITEM>
    <LINE_ITEM>
      <PRODUCT_DESC>Gap personal jean</PRODUCT_DESC>
      <QUANTITY>1</QUANTITY>
    </LINE_ITEM>
    <LINE_ITEM>
```

```
    <PRODUCT_DESC>Hooded Pullover Fleece Sweatshirt</PRODUCT_DESC>
    <QUANTITY>1</QUANTITY>
  </LINE_ITEM>
</prefix2:ORDER_SUMMARY>
<prefix2:ORDER_SUMMARY TYPE="APPL" xmlns:prefix2="urn:retailerType">
  <ORDER_ID>ORDER_9_12</ORDER_ID>
  <ORDER_DATE>2003-01-09</ORDER_DATE>
  <TOTAL_ORDER_AMOUNT>476.55</TOTAL_ORDER_AMOUNT>
  <SHIP_TO_NAME>Marge Simpson</SHIP_TO_NAME>
  <ESTIMATED_SHIP_DATE>2003-01-12</ESTIMATED_SHIP_DATE>
  <TRACKING_NUMBER>ORDER_9_127771190162</TRACKING_NUMBER>
  <LINE_ITEM>
    <PRODUCT_DESC>Lands End Athelic Slides</PRODUCT_DESC>
    <QUANTITY>3</QUANTITY>
  </LINE_ITEM>
  <LINE_ITEM>
    <PRODUCT_DESC>Hush Poppies Angella II</PRODUCT_DESC>
    <QUANTITY>1</QUANTITY>
  </LINE_ITEM>
  <LINE_ITEM>
    <PRODUCT_DESC>Debra Sandal at Nodstrom</PRODUCT_DESC>
    <QUANTITY>1</QUANTITY>
  </LINE_ITEM>
</prefix2:ORDER_SUMMARY>
<prefix2:ORDER_SUMMARY TYPE="APPL" xmlns:prefix2="urn:retailerType">
  <ORDER_ID>ORDER_9_17</ORDER_ID>
  <ORDER_DATE>2003-07-22</ORDER_DATE>
  <TOTAL_ORDER_AMOUNT>2331.55</TOTAL_ORDER_AMOUNT>
  <SHIP_TO_NAME>Homer Simpson</SHIP_TO_NAME>
  <ESTIMATED_SHIP_DATE>2003-07-29</ESTIMATED_SHIP_DATE>
  <TRACKING_NUMBER>ORDER_9_175524055754</TRACKING_NUMBER>
  <LINE_ITEM>
    <PRODUCT_DESC>Burberry Nova Check Hobo</PRODUCT_DESC>
    <QUANTITY>3</QUANTITY>
  </LINE_ITEM>
  <LINE_ITEM>
    <PRODUCT_DESC>Prada Patent Leather Handbag</PRODUCT_DESC>
    <QUANTITY>1</QUANTITY>
```

```
        </LINE_ITEM>
        <LINE_ITEM>
          <PRODUCT_DESC>Prada tote</PRODUCT_DESC>
          <QUANTITY>1</QUANTITY>
        </LINE_ITEM>
      </prefix2:ORDER_SUMMARY>
      <prefix2:ORDER_SUMMARY TYPE="ELEC" xmlns:prefix2="urn:retailerType">
        <ORDER_ID>ORDER_9_4</ORDER_ID>
        <ORDER_DATE>2003-01-09</ORDER_DATE>
        <TOTAL_ORDER_AMOUNT>429.55</TOTAL_ORDER_AMOUNT>
        <SHIP_TO_NAME>Marge Simpson</SHIP_TO_NAME>
        <ESTIMATED_SHIP_DATE>2003-01-11</ESTIMATED_SHIP_DATE>
        <TRACKING_NUMBER>ORDER_9_42871564731</TRACKING_NUMBER>
        <LINE_ITEM>
          <PRODUCT_DESC>Samsung SS8</PRODUCT_DESC>
          <QUANTITY>3</QUANTITY>
        </LINE_ITEM>
        <LINE_ITEM>
          <PRODUCT_DESC>D-Link Wireless 22 Mbps Broadband Router
          </PRODUCT_DESC>
          <QUANTITY>1</QUANTITY>
        </LINE_ITEM>
        <LINE_ITEM>
          <PRODUCT_DESC>Samsung DVD-V2000 DVD/VCR Combo</PRODUCT_DESC>
          <QUANTITY>1</QUANTITY>
        </LINE_ITEM>
      </prefix2:ORDER_SUMMARY>
    </ORDERS>
    <CASES/>
  </CUSTOMER_VIEW>
</prefix1:CustomerView>
```

Definitions of Key Terms

This topic introduces the following key Liquid Data concepts:

- Data Sources
- Queries, Results, and Schemas
- Data Views
- Caching
- Custom Functions

For a description of Liquid Data components, see "Liquid Data Architecture Components" on page 2-2.

Data Sources

In Liquid Data, a *data source* is a source of information that can be queried. Liquid Data supports querying the following types of data sources:

- Relational databases (RDBMSs), including stored procedures, via JDBC
- Web services
- Application views, which are business-level interfaces to data in packaged application such as Siebel, PeopleSoft, or SAP. Application views are used in conjunction with the WebLogic Adapters.
- XML files, delimited files
- Data views, which are the dynamic results of queries stored along with the queries that produce them

You use the WebLogic Administration Console to configure access to data sources. For more information, see the following topics in the Liquid Data *Administration Guide*:

- "Configuring Access to Relational Databases"
- "Configuring Access to XML Files"
- "Configuring Access to Delimited Files"
- "Configuring Access to Web Services"
- "Configuring Access to Application Views"

- "Configuring Access to Data Views"

You can use the Data View Builder to design and build queries that query the data sources you have configured. Alternatively, you can use a text editor to create the queries. Queries are built based on source and target schemas that describe the structure of the data queried (source) and returned (target). For more information about creating queries, see *Building Queries and Data Views*.

Queries, Results, and Schemas

A *query* is a request for information based on explicit criteria. In Liquid Data, queries are created and used in compliance with the XQuery language specification. With the Data View Builder, you can create queries using a drag-and-drop paradigm between source and target schemas.

Users can create and access two types of queries:

- *Stored queries* are XQuery queries that have been saved in the Liquid Data repository.
- *Ad hoc queries* are queries that have not been stored in the Liquid Data repository but rather are passed to the Liquid Data server on the fly. An ad-hoc query is any hand-coded query or Data View Builder-generated query that is not a stored query.

A *result* is the information that a query yields. In Liquid Data, the query result is generated in XML format. The *target schema* is an 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 in the form in which you define for *target data*.

For more information about queries, results, and target schemas, see "Overview and Key Concepts" in *Building Queries and Data Views*.

Data Views

A *data view* is a query that can be used as a data source for other queries. Data views provide a logical view of the data, abstracting all of the details of the underlying data sources from the application developer. Developers can then easily create queries for their applications without requiring any knowledge about the specific underlying systems. In this way, application developers, not just data architects, can directly access distributed, heterogeneous data sources. For more information about data views, see "Overview and Key Concepts" and "Using Data Views as Data Sources" in *Building Queries and Data Views*.

Caching

Liquid Data caches information about commonly executed queries for subsequent, efficient retrieval, thereby enhancing overall system performance. Liquid Data supports two types of caching:

- The *query plan cache* is a memory-based cache that stores the query plans of commonly executed queries.
- The *result set cache* is a database-based cache that stores the results of commonly executed queries. For more information, see "Configuring the Query Results Cache" in the Liquid Data *Administration Guide*.

Custom Functions

Custom functions are user-defined functions that perform specialized tasks. The Liquid Data provides a set of standard 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. Custom functions can be used for a wide range of requirements including implementing customized data manipulation, accessing external or custom legacy systems, and so on. Once configured, custom functions display as functions available for use in the Data View Builder. For more information about custom functions, see "Using Custom Functions" in the *Application Developer's Guide* and "Configuring Access to Custom Functions" in the Liquid Data *Administration Guide*.

CHAPTER 4

Liquid Data Quick Start

The following sections guide you through all the tasks required to design and implement a real-time data access and aggregation solution using BEA Liquid Data for WebLogic. This chapter provides a high-level view of a typical process. To read more detail about any particular task, follow the links provided to the document where that topic is discussed in full detail.

The following sections are included:

- Step 1. Install the Software
- Step 2: Explore the Liquid Data Samples
- Step 3. Define the Data Access and Aggregation Requirements and Scope
- Step 4. Configure Access to Data Sources
- Step 5. Create and Test Queries in the XQuery Language
- Step 6: Develop Applications to Display Liquid Data Queries
- Step 7. Deploy the Data Access and Aggregation Solution

For a walkthrough of steps 3, 4, and 5, see Getting Started With Liquid Data.

Step 1. Install the Software

To use Liquid Data, you begin by installing the WebLogic Platform and the Liquid Data software according to the instructions in *Installing Liquid Data*.

Step 2: Explore the Liquid Data Samples

After you install the software, launch the samples server, and then explore Liquid Data using the Liquid Data samples. For more information, see "Overview of the Avitek Self-Service Sample Application" in *Liquid Data by Example*.

Step 3. Define the Data Access and Aggregation Requirements and Scope

Before you can create queries, you need to clarify the requirements and scope of your data access and aggregation solution, answering such questions as:

- Who is the audience for data access and aggregation?
- What kinds of problem statements/queries, in plain English, are you trying to solve?
- What are the data sources and their types?
- What is the required information *output*, also known as the *query result*? What specific content is required and how should it be presented (layout, order, and so on)? How should the output or query result be structured in the *target schema*? What source-to-target mappings do you need to create in order to shape the result as needed?
- What is the required information *input*? Which data sources can provide such information? What *types* of data sources are involved—relational databases, XML files, Web services, application views, or data views? What is required to obtain access to these data sources? What is the structure of the data sources (*source schemas*)? What *conditions* do you need to define in a query in order to obtain the appropriate view on the information?

Step 4. Configure Access to Data Sources

Before you can create queries, you must configure access to the data sources that provide these queries with the required content. You use the Liquid Data node in the Administration Console to configure access to data sources, via *data source descriptions*, as described in the following topics in the Liquid Data *Administration Guide*:

- "Configuring Access to Relational Databases"
- "Configuring Access to XML Files"
- "Configuring Access to Web Services"
- "Configuring Access to Application Views"
- "Configuring Access to Data Views"

Certain data source types, such as relational databases and application views, require additional configuration tasks that are also described in their respective sections. You might also need to perform other configuration tasks described elsewhere in the Liquid Data *Administration Guide*.

Step 5. Create and Test Queries in the XQuery Language

Once you have configured access to data sources, you can create XQuery queries to retrieve the information needed for your data access and aggregation solution. You can either use the Data View Builder to generate the XQuery for you (based on a design you construct using its graphical tools), or you can hand code the XQuery. Either way, you use the Data View Builder to save queries as stored queries in the Liquid Data server repository. For more information, see *Building Queries and Data Views*.

Step 6: Develop Applications to Display Liquid Data Queries

Client applications can invoke Liquid Data queries in the following ways:

- The WebLogic Workshop Liquid Data control provides easy access to Liquid Data queries in the WebLogic Workshop environment. The Liquid Data control provides access to both stored and ad-hoc queries, and generates XMLBeans for the stored queries (based on the schema associated with the stored queries). The XMLBean interface makes it easy to use NetUI and other workshop features to create robust applications that display and process data from Liquid Data.

- The Liquid Data Query API (Application Programming Interface) allows EJB and JSP client applications to invoke queries programmatically. Developers can create custom Java applications or build JSP pages. In addition, other BEA software—WebLogic Portal, the Business Process Management component of WebLogic Integration, WebLogic Web Services, and WebLogic Workshop—can invoke queries on the Liquid Data Server, providing real-time integration with Liquid Data.
- Web service clients can invoke Liquid Data queries using Web services that have been generated in the Administration Console from stored queries.

For more information, see the "*Application Developer's Guide.*"

Step 7. Deploy the Data Access and Aggregation Solution

After you have created all the components of your data access and aggregation solution, you need to deploy it in a production environment. In addition to defining deployment requirements and designing the deployment, you need to deploy the Liquid Data Server, the server repository, and other resources in a WebLogic domain in a production environment. For more information, see "*Deploying Liquid Data*."

Index

Q

R

S

T

W

X