BEAAquaLogic Data Services Platform™

Installation Guide

Note: Product documentation may be revised post-release and made available from the following BEA e-docs site:

http://e-docs.bea.com/aldsp/docs21/index.html

Version: 2.1
Document Date: June 2005
Revised: March 2006

Contents

About This Document

1. Preparing to Install BEA AquaLogic Data Services Platform

2. Installing Data Services Platform Using GUI Mode

3. Installing Data Services Platform Using Console or Silent Mode

4. Post-Installation Tasks

5. Migrating Legacy DSP Applications

6. Sample Retail Application Overview

About This Document

This document explains how to install BEA AquaLogic Data Services Platform (DSP) and how to perform a quick test to verify that the software is properly installed.

This document covers the following topics:

- Chapter 1, "Preparing to Install BEA AquaLogic Data Services Platform," provides information about supported platforms databases, installation prerequisites (system software and hardware requirements), and licensing. It also summarizes the software and components that make up a full Data Services Platform installation.

- Chapter 2, "Installing Data Services Platform Using GUI Mode," explains how to install Data Services Platform using the GUI mode install process on Windows and UNIX systems.

- Chapter 3, "Installing Data Services Platform Using Console or Silent Mode," explains how to install Data Services Platform using the command-line console or *silent mode* install process.

- Chapter 4, "Post-Installation Tasks," explains how to run a quick test to verify that you have a complete and working Data Services Platform installation. The chapter also explains how to uninstall Data Services Platform, if required.

- Chapter 6, "Sample Retail Application Overview," provides a brief overview of the Retail Sample Application (RTLApp), including a description of data sources, queries, and other application components used to develop the RTLApp.

What You Need to Know

This document is intended for system administrators or others who want to install and set up DSP.

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 Data Services Platform 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 PDF files, open the Data Services Platform 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

The following documents contain information that is useful to refer to after installing DSP:

- Data Services Platform *Concepts Guide*
  (`http://e-docs.bea.com/aldsp/docs20/concepts/index.html`)

- Data Services Platform *Administration Guide*
  (`http://e-docs.bea.com/aldsp/docs20/admin/index.html`)

Contact Us!

Your feedback on the BEA Data Services Platform 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 those who create and update the DSP documentation.

In your e-mail message, please indicate that you are using the documentation for the BEA AquaLogic Data Services Platform 8.1 release.

If you have any questions about this version of DSP, or if you have problems installing and running DSP, 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 with the following:

- 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 that 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`

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

Convention	Item
{ }	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.

About This Document

Preparing to Install BEA AquaLogic Data Services Platform

Before you install BEA AquaLogic Data Services Platform (DSP), you need to verify that your system meets minimum requirements. This chapter provides information about supported platforms, system hardware and software requirements, and describes installation prerequisites. The chapter also provides an overview of what is installed.

The following sections are included:

- Supported Platforms, Databases and JDBC Drivers

- Installation Prerequisites

- What Gets Installed

Note: Data Services Platform was initially named Liquid Data. Some instances of the original name remain in the product, installation path, and components.

Installation Overview

This section provides a high level overview of the steps required to install DSP, illustrated in Figure 1-1.

Figure 1-1 Overview of Installation Process
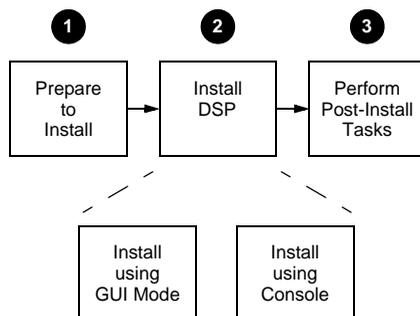
To install DSP:

1. Prepare to install the software.

   Preparing to install DSP involves checking the supported platforms and databases, and verifying that you have the installation prerequisites in place. You can also familiarize yourself with a description of what gets installed on your system. To prepare to install the software, complete the remaining sections of this chapter.

2. Uninstall any previous versions of Data Services Platform. For details see "Uninstalling Data Services Platform" on page 4-6.

   Note: The Uninstall option does not remove any Data Services Platform or other Workshop applications that you have created.

3. Install DSP 2.1.

   You can install the Data Services Platform software either in GUI mode, featuring an easy to use graphical installer application, or in interactive Console mode, suitable on UNIX systems without a graphics (windowing) workstation. Alternatively, you can choose to install DSP on Windows and UNIX systems using Silent mode, which reads an installer properties file to determine the install options.

   For more information about using GUI mode, see Chapter 2, "Installing Data Services Platform Using GUI Mode." For more information about using Console mode and Silent mode, see Chapter 3, "Installing Data Services Platform Using Console or Silent Mode."

4. Perform post-installation tasks.

   Following installation, you can verify that the installation was successful, and explore Windows shortcuts and UNIX paths to key components. For more information, see Chapter 4, "Post-Installation Tasks."

Supported Platforms, Databases and JDBC Drivers

As a WebLogic Workshop application, DSP generally supports the same platforms, operating systems, processor architecture, SDKs, and RDBMS systems as does WebLogic Platform 8.1 SP4 or SP5.

In general any thread-safe, transaction-callable DBMS accessible through JDBC should be available to DSP. For complete details see Supported Configurations for WebLogic Platforms.

Note: The minimum RAM recommendations are for one instance of WebLogic Server on which DSP is running. You may need more memory if you run two or more instances of WebLogic Server and DSP. Some systems use disk space as virtual RAM. Performance when running in virtual RAM may be markedly slower than when running in physical RAM.

Supported Operating Systems and Hardware

The following table lists operating systems and/or hardware that has been specifically tested with Data Services Platform under BEA WebLogic Platform 8.1 SP4 or SP5.

Table 1-1 Matrices of Operating Systems/Hardware Supported by DSP 2.1

Operating System	Version(s)
Microsoft Windows	• 2000 Server, Advanced Server on x86 • Server 2003 Standard, Enterprise, and Datacenter on x86 • XP on x86
Red Hat Enterprise Linux	• 3.0 AS, ES on x86 • 4.0 AS, ES on x86
Sun Microsystems	• Solaris 8 on SPARC • Solaris 9 on SPARC • Solaris 10 on SPARC
HP-UX	• 11.0 and 11i on PA-RISC

Supported SQL Databases

DSP offers two levels of database vendor support:

- **Full support.** Full support for specific vendors and versions includes reads, writes, and stored procedures. DSP 2.1 provides this support for specific offerings from from Oracle, IBM, Sybase, and Microsoft. See Table 1-2 for details on full vendor and version support.

- **Base platform support.** A base platform is support for reading and writing data, but not stored procedures. Base platforms for DSP 2.1 include:

  - Informix

  - Microsoft Access

  - MySQL

The following table lists databases which have been specifically tested with DSP under WebLogic Platform 8.1, service packs SP4 and SP5.

Table 1-2 SQL Database Versions Supported by DSP 2.1

Database	Version(s)
Oracle	9.2.0
	10g
IBM	DB2 8.1
Sybase	12.5.2
Microsoft SQL Server	2000
PointBase	4.4[1]
Other Base Platforms	n/a

1. PointBase Server is an all-Java DBMS product included in the WebLogic Platform 8.1 distribution solely for evaluation purposes, either in the form of custom trial applications or through packaged sample applications provided with WebLogic Platform. Non-evaluation development or other use of the PointBase Server requires that a separate PointBase Server license be obtained by the end user directly from DataMirror. Data Services Platform does not support PointBase for production server deployments.

Supported Database-JDBC Driver Matrices

The following table describes results of testing DSP against various databases under different JDBC drivers. Both the BEA and native drivers were tested in XA and non-XA modes. Limitation encountered are noted as CR numbers in the Comments column.

Details related to those numbers can be found in the DSP *Release Notes*.

Table 1-3 Matrices of Databases and JDBC Drivers Supported by Data Services Platform

Database	JDBC Driver Type	Data Source Type	Comments
Oracle	BEA non-XA	tables	CR199675
		stored procedures	CR202963 CR212515
	BEA XA	tables	CR199675
		stored procedures	CR202963 CR212515
	Oracle non-XA	tables	CR265965
		stored procedures	CR212515 CR202962
	Oracle XA	tables	
		stored procedures	CR212515 CR202962
DB2	BEA non-XA	tables	
		stored procedures	CR227440
	BEA XA	tables	
		stored procedures	CR227440
	DB2 non-XA	tables	
		stored procedures	CR227440
	DB2 XA	tables	CR227486
		stored procedures	CR227440

Database	JDBC Driver Type	Data Source Type	Comments
Sybase	BEA non-XA	tables	
		stored procedures	
	BEA XA	tables	
		stored procedures	
	Sybase non-XA	tables	CR223429
		stored procedures	
	Sybase XA	tables	CR223429
		stored procedures	
Informix	BEA non-XA	tables	
		stored procedures	
	BEA XA	tables	
		stored procedures	
	Native non-XA	tables	CR223486 CR226171
		stored procedures	CR230264
	Native XA	tables	
		stored procedures	CR230264
MS-SQL Server	BEA non-XA	tables	CR211701
		stored procedures	CR202041
	BEA XA	tables	
		stored procedures	
	MS-SQL Server non-XA	tables	CR214730
		stored procedures	CR202041

Database	JDBC Driver Type	Data Source Type	Comments
	MS-SQL Server XA	tables	
		stored procedures	

Configuring the DSP JDBC Driver for Reporting Applications

The following table describes the matrices for configuring the Data Services Platform JDBC driver for supported reporting applications.

Table 1-4 Matrices of Required Connectivity Software for Reporting Applications Supported by the Data Services Platform JDBC Driver

Application	Connectivity Software		
		ODBC	
	JDBC Native	EasySoft ODBC/JDBC Gateway	OpenLink ODBC/JDBC Lite Bridge
Crystal Reports 10 using ODBC	n/a	Yes	Yes
Crystal Reports 10 using JDBC plug-in	Yes	n/a	n/a
Business Objects 6.1	No	Yes	No
MS Access 2000	No	No	Yes
DBVisualizer	Yes	n/a	n/a

Supported Web Services Standards

The following table lists Web services standards supported by DSP 2.1.

Table 1-5 Supported Web Services Standards

Web Service Standard	Version
SOAP	1.1
WSDL	1.1
JAX-RPC	1.0

Installation Prerequisites

You must have the following software installed on the system where you plan to install DSP:

- **WebLogic Server or WebLogic Platform 8.1 SP4 or SP5.** The latest version of WebLogic Platform 8.1 is available as a Web download from the BEA Download Center from:

  http://commerce.bea.com/downloads/products.jsp.

  See the WebLogic Platform Installation Guide for instructions on installing WebLogic Platform.

- **WebLogic Workshop.** You need to have WebLogic Workshop available to access DSP extensions to the development environment.

Note: If you are running Windows 2000, the maximum classpath size can be exceeded. For this reason it is recommended that when you install DSP, you install it into <BEAHOME> directory with a directory name of four characters or less such as <BEA>.

What Gets Installed

The complete installation of Data Services Platform results in the installation of the following principal components on the system:

- Data Services Platform (running as a component in WebLogic Platform).
- BEA AquaLogic Data Services Platform configuration and monitoring pages added to the WebLogic Administration Console.
- Data service control
- Data Services Platform Console
- Data Services Platform Sample domain
- RTLApp sample application

The following documents contain information that is useful to refer to after installing DSP:

- Data Services Platform *Concepts Guide*
  (http://e-docs.bea.com/aldsp/docs20/concepts/index.html)

Note: Data Services Platform *Administration Guide*
     (http://e-docs.bea.com/aldsp/docs20/admin/index.html)

When you install DSP, a valid evaluation license is automatically included with the installation.

Preconfigured Samples Domain

The full Data Services Platform with Samples option provides a preconfigured samples domain as shown in the following table.

Table 1-6 DSP Samples Preconfigured Domain and Start Commands for Samples Server

Platform	Windows and UNIX Paths to Start in Each Domain	Description
Windows	Start →Programs →BEA WebLogic Platform 8.1 →BEA AquaLogic Data Services Platform 2.1 →Examples →RTL Demo	Starts sample RTLDemo on Windows

	or
	
	`<WL_HOME>/samples/domains/liquiddata/ \`
	`startWeblogic.cmd`
	`<WL_HOME>/samples/domains/ldplatform/ \`
	`startWeblogic.cmd`
		Starts the DSP samples server on Windows

UNIX	`<WL_HOME>/samples/domains/liquiddata/ \`
	`startWeblogic.sh`
	or
	`<WL_HOME>/samples/domains/ldplatform/ \`
	`startWeblogic.sh`
		Starts the DSP samples server on UNIX

Note: `<WL_HOME>` is the home location of your WebLogic installation. On Windows the default `<WL_HOME>` is:

```
c:\bea\weblogic81.
```

For a detailed explanation of domains, see "Creating WebLogic Configurations Using the Domain Control Wizard" in the WebLogic Platform documentation.

Internationalization Support

DSP is internationalized and supports multi-byte data from the underlying data sources. Specifically, DSP works with Japanese character sets, where the underlying databases are running in Japanese locales.

Installing Data Services Platform Using GUI Mode

This chapter describes how to install Data Services Platform (DSP) using GUI mode. It includes the following sections:

- Before You Install

- Installing Using GUI Mode

Following installation, you can perform specific post-installation tasks such as verifying successful installation and exploring DSP shortcuts and paths. For more information, see Chapter 4, "Post-Installation Tasks."

Figure 2-1 illustrates the current step in the overall installation process.

Figure 2-1 GUI Mode Installation Step
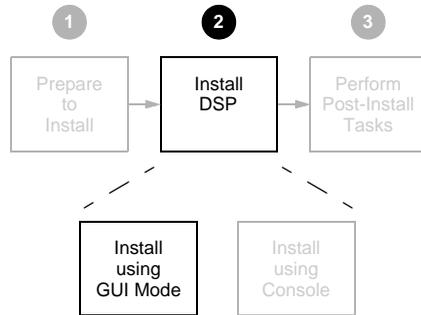
Before You Install

Before you begin installing DSP, confirm that the following conditions are met:

- A supported version of WebLogic Server or WebLogic Platform 8.1 is installed in the `<BEAHOME>` directory in which you are going to install DSP.

- Any previous installations of Data Services Platform have been uninstalled. See "Uninstalling Data Services Platform" on page 4-6 for details.

- It is recommended for Windows (and a requirement for UNIX) that an entry for the JDK142_08 `bin` directory (for example `<BEAHOME>/JDK142_08/bin`) be included in your PATH environment variable setting before any other JDK bin directories.

For additional information about installation prerequisites, see "Installation Prerequisites" on page 1-10 in Chapter 1, "Preparing to Install BEA AquaLogic Data Services Platform."

Installing Using GUI Mode

This section describes how to using the GUI installer application to install DSP on Microsoft Windows and UNIX-based platforms.

To install DSP in GUI mode:

1. Launch the GUI installer application.

    On Microsoft Windows-based systems, do the following:

    a. Navigate to the folder to which you downloaded the DSP installer using Windows Explorer.

    b. Double-click on the installer executable file. For example:

        ```
        aldsp_210_win32.exe
        ```

    On UNIX and Linux-based systems, do the following:

    a. Verify that the console attached to

    b. the machine on which you are installing the software supports a Java-based GUI.

    c. Open a command window, and change (cd) to the directory to which you downloaded the DSP installer file.

    d. Start the installer in a new shell by entering the following:

        sh *filename.bin*

        where *filename.bin* is the name of the DSP installation program specific to your platform. For example, enter the following to start the Solaris version of the DSP installation program:

        ```
        sh aldsp_210_solaris32.bin
        ```

        Alternatively, if you have the current JDK supported by WebLogic Platform 8.1 in your path you can run:

        ```
        ./aldsp_210_linux32.bin
        ```

The installer application welcomes you to install DSP, as illustrated in Figure 2-2.

Figure 2-2 DSP Installation Welcome Screen
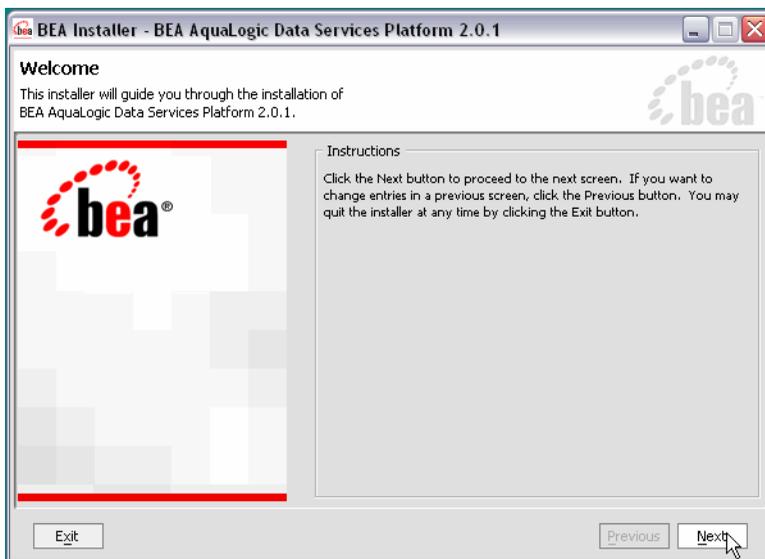
2. Click Next to begin the installation.

    The License Agreement appears.

3. Review the License Agreement, using the scroll bar to display the text. Choose Yes to accept, and click Next.

    A screen appears enabling you to specify the BEA home directory, as illustrated in Figure 2-3.

Figure 2-3 Choose BEA Home Directory
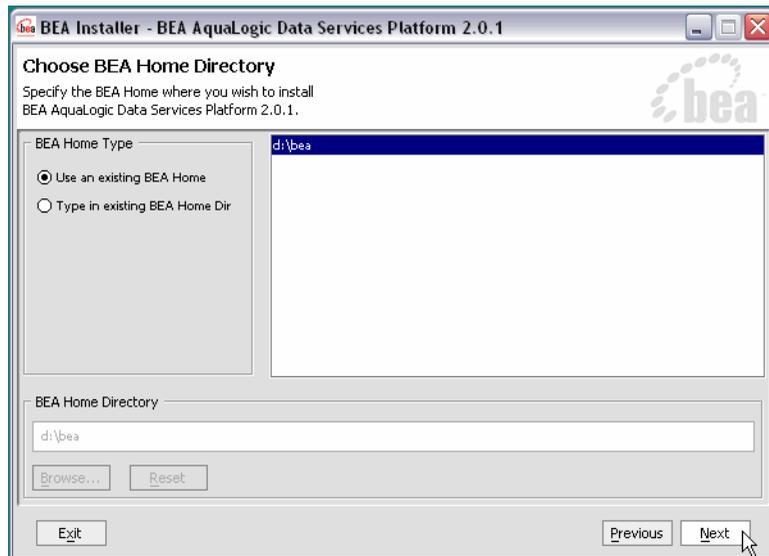
The BEA home (*<BEAHOME>*) directory serves as the central support directory for all the BEA products installed on your system. For a detailed description of how the *<BEAHOME>* directory is used, see "BEA Home Directory" in "Preparing to Install WebLogic Server" in *Installing WebLogic Server*. This document is available, in the BEA WebLogic Server document set, at the following URL:

http://e-docs.bea.com/wls/docs81/install/prepare.html

4. Specify the BEA home directory, and click Next.

   You can choose a directory from the list, or type the directory name in the BEA Home Directory field. You must install DSP in the same directory where you installed WebLogic Platform 8.1

   A warning message is displayed under two conditions:

   – When a previously installed copy of DSP is found. In this case, you must uninstall DSP before installing the product again.

   – When WebLogic Platform 8.1 is not found in the specified BEA home directory. Verify that the specified directory has a WebLogic Platform 8.1 installation. If not, exit the DSP installation and install WebLogic Platform 8.1 before proceeding.

   A screen appears enabling you to choose the type of installation, as illustrated in Figure 2-4.

Figure 2-4 Choose Install Set

5. Choose either Complete or Custom as the install type, and click Next.

    The following describes the options:

    – Complete

        This option installs all DSP components, including the DSP runtime, the WebLogic
        Workshop extensions, the WebLogic Administration Console, and sample domains.

    – Custom

        Enables you to choose which DSP components to install. Choosing Custom causes the
        screen in Figure 2-5 to appear after you click Next. Choose the components and click Next.

Figure 2-5 Choosing Components to Install
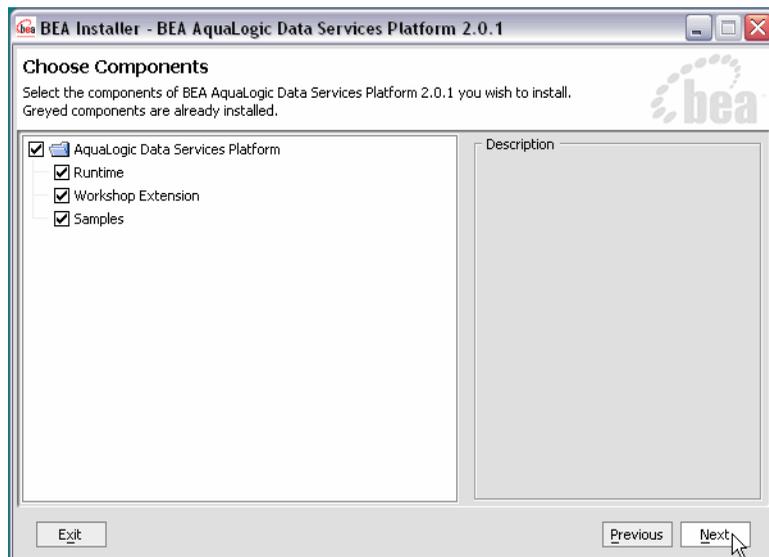
The installer displays a progress screen, and begins installing the files on your system. When
the installation is complete, the Installation Complete screen appears.

6. Click Done.

    You have successfully installed an instance of Data Services Platform.

Installing Data Services Platform Using Console or Silent Mode

This section describes how to install BEA AquaLogic Data Services Platform (DSP) using either Console mode or Silent mode. Console mode is an interactive installation that you can use on UNIX systems without a graphics (windowing) workstation.

Silent mode is a non-interactive installation on a Windows or UNIX system. In Silent mode, the installer application uses a properties file to obtain installation parameters.

The chapter includes the following sections:

- Before You Install

- Using Console Mode to Install DSP

- Using Silent Console Mode to Install DSP

Following installation, you can perform specific post-installation tasks such as verifying successful installation and exploring DSP shortcuts and paths. For more information, see Chapter 4, "Post-Installation Tasks."

Figure 3-1 illustrates the current step in the overall installation process.

Figure 3-1 Console Mode Installation Step
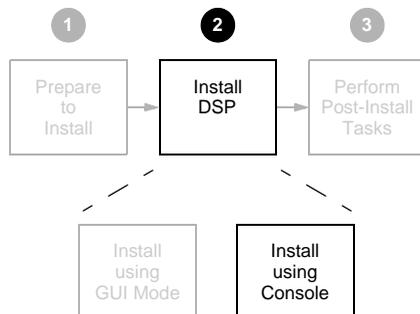
Before You Install

Before you begin installing DSP, confirm that the following conditions are met:

- WebLogic Server or WebLogic Platform 8.1 SP5 is installed in the *<BEAHOME>* directory in which you are installing DSP.

- DSP is not running on the system.

  You must remove DSP in cases when it is located in the same *<BEAHOME>* directory to which you plan to install. See "Uninstalling Data Services Platform" on page 4-6 for details.

- An entry for the JDK142_08 bin directory (for example *<BEAHOME>*/JDK142_08/bin) must be included in your PATH environment variable setting before any other JDK bin directories.

For additional information about installation prerequisites, see "Installation Prerequisites" on page 1-10 in Chapter 1, "Preparing to Install BEA AquaLogic Data Services Platform."

Using Console Mode to Install DSP

This section describes the console-mode installation procedure.

Note: You cannot install DSP in console mode on Solaris systems running in a Japanese locale. In cases when you are installing on a Japanese Solaris system, change your locale in the shell in which you are installing to English, and then install the product. After DSP is installed, you can run it in a Japanese locale.

To install DSP on a UNIX system in console mode:

1. Change to the directory (cd) that contains the DSP installer.

2. Start the installation program.

    The installer file name has the following form:

        filename.bin

    where *filename* is the name of the DSP installation program specific to your platform. For example, enter the following to start the Solaris version of the DSP installation program:

    ```
    sh aldsp_210_solaris32.bin -mode=console
    ```

    Similarly, enter the following to start the Linux version of the DSP installation program:

    ```
    ./aldsp_210_linux32.bin -mode=console
    ```

    A message appears indicating that the software is being extracted, followed by a welcome message, as shown in the following text:

    ```
    <--------- BEA Installer - BEA AquaLogic Data Services Platform 2.1
    --------->
    Welcome:
    --------
    This installer will guide you through the installation of BEA AquaLogic
    Data Services Platform 2.1. Type "Next" or enter to proceed to the next
    prompt. If you want to change data entered previously, type "Previous".
    You may quit the installer at any time by typing "Exit".

    Enter [Exit][Next]>
    ```

3. Press Enter to continue.

    The license agreement is displayed. Review the entire agreement. When you reach the end of the agreement, you are prompted to accept or reject the terms.

    ```
    Select Option:
    1 - Yes, I agree with the terms of the license
    2 - No, I do not agree with the terms of the license

    Enter option number to select OR [Down][Exit][Previous]>
    ```

4. Enter 1 if you accept the license agreement.

    You are prompted to choose the BEA home directory, as shown in the following text:

    ```
    <--------- BEA Installer - BEA AquaLogic Data Services Platform 2.1
    --------->
    Choose BEA Home Directory:
    -------------------------

    "BEA Home" = [/usr/local/bea]

    Input existing BEA Home OR [Exit][Previous][Next]>
    ```

    The *<BEAHOME>* directory serves as the central support directory for all the BEA products installed on your system. For a detailed description of how this directory is used, see "BEA Home Directory" in "Preparing to Install WebLogic Server" in *Installing BEA WebLogic Server*. This document is available, in the WebLogic Server document set, at the following URL:

    http://e-docs.bea.com/wls/docs81/install/prepare.html

    A warning message is displayed in cases when a previous installation of DSP is found, or when WebLogic Server 8.1 is not found in the specified BEA home directory.

5. Press Enter to accept the default BEA home directory, or enter an alternative directory.

    When prompted, confirm the directory you selected. You are prompted to choose whether you want to perform a complete or custom installation, as shown in the following text:

    ```
    <--------- BEA Installer - BEA AquaLogic Data Services Platform 2.1
    --------->
    Choose Install Type:
    -------------------

    -> 1|Complete
        |Install the complete BEA AquaLogic Data Services Platform 2.1.
       2|Custom Installation
        |Choose software components to install and perform optional
    configuration.

    Enter index number to select OR [Exit][Previous][Next]>
    ```

6. Enter 1 or 2 to choose a complete or custom installation respectively.

    When you specify a custom installation, the installer enables you to choose the components to install, as shown in the following text:

    ```
    <--------- BEA Installer - BEA AquaLogic Data Services Platform 2.1
    --------->
    Choose Components to install:
    ----------------------------
    Release 2.0.1.0
    |_____BEA AquaLogic Platform [1] x
    |_____BEA AquaLogic Runtime [1.1] x
    |_____Workshop Extension [1.2] x
    |_____Samples [1.3] x

    Enter number exactly as it appears in brackets to toggle selection OR
    [Exit][Previous][Next]>
    ```

    Enter the value for the component, for example, 1.1 for the DSP (Liquid Data) Runtime. The installer prompts you for additional component values. Continue selecting components, and press Enter without a value when you are done.

    The installation begins and the installation status is displayed as it progresses. When the installation is complete, "Congratulations!" is displayed.

7. Press Enter to exit the installer.

    This completes the Console Mode installation process.

Using Silent Console Mode to Install DSP

The Windows, UNIX, and Linux versions of the installer provide a noninteractive, or *silent-mode* installation, that you can use in cases when you want to install DSP without needing to supply information from the keyboard during the installation process. Instead, the installer gets the required information from a properties file that you provide.

Before launching a silent-mode installation, make sure that all installation prerequisites are met and that all the information in the properties file is correct. After the silent installer is started, it proceeds in the background and does not report exceptions. Some exceptions are ignored.

However, if a previously installed copy of DSP is detected, a dialog box will appear asking if you want to override the old version. Other exceptions cause the installer to fail. For example, if the specified BEA Home directory `<BEAHOME>` or the specified DSP install directory (`USER_INSTALL_DIR`) do not exist or are incorrect, the installer fails.

Note: On UNIX systems, the installer displays the message `Installation Complete` when it finishes. This message does not necessarily indicate that the installer was successful; it means only that the process has finished running.

If a fatal exception occurs during installation, the installer displays a message and no changes are made to the system.

Using Silent Mode on Windows and UNIX Systems

This section describes how to install DSP on Windows and UNIX systems using Silent Mode.

To install using silent mode:

1. Create the required installer properties file.

   The content of the file is described in "Exploring the Silent Mode Installer Properties File" on page 3-9. You can use any legal file name for the installer properties file. Verify that the `<BEAHOME>`, `<USER_INSTALL_DIR>`, and other values specified in the properties file are correct and that all requirements have been met.

2. Open a command window.

3. Navigate to the directory containing the DSP installer.

   – On Windows, the installer application is:

   ```
   aldsp_210_win32.exe
   ```

   – On UNIX and Linux, the installer application is:

   filename.bin

   where *filename.bin* is the name of the DSP installation program specific to your platform; for example, `aldsp_210_solaris32.bin` for the Solaris version of DSP.

   You will find similarly-named installer files for all UNIX and Linux versions of DSP.

4. Run the installer application, specifying the properties file and the log file name as options.

   – On Windows, run the following command:

   ```
   aldsp_210_win32.exe -mode=silent
         -silent_XML=<drive:\properties_file_path>
         -log=<drive:\log_file_path>
   ```

   where `drive` is the letter that identifies the hard disk drive, `properties_file_path` is the complete pathname of the DSP silent installation properties file, and `log_file_path` is the complete pathname of the log file.

   For example, you could enter the following:

   ```
   aldsp_210_win32.exe -mode=silent
         -silent_XML=c:\temp\ld_properties.txt
         -log=c:\temp\logfile.txt
   ```

   You are returned to the command prompt and the DSP installation preparation dialog box is briefly displayed. The installation proceeds in the background with the information specified in the installer properties file.

   To verify that the installer is running, open the Windows Task Manager. The installer is listed as the `javaw.exe` process.

– On UNIX, run the following command:

```
sh aldsp_210_solaris32.bin -mode=silent
      -silent_XML=complete_properties_file_path
      -log=complete_log_file_path
```

where *complete_properties_file_path* is the complete pathname of the properties file and *complete_log_file_path* is the complete pathname of the log file. A complete path is required, even when the file resides in the same directory as the `aldsp_210_solaris32.bin` file.

The message Preparing to Install is displayed. After the installer decompresses the required files, the installation proceeds with the information specified in the installer properties file. When the process is successful, Installation Complete is displayed.

– On Linux, run the following command:

```
./aldsp_210_linux32.bin -mode=silent
      -silent_XML=complete_properties_file_path
      -log=complete_log_file_path
```

where `complete_properties_file_path` is the complete pathname of the properties file and `complete_log_file_path` is the complete pathname of the log file. A complete path is required, even when the file resides in the same directory as the `aldsp_210_linux32.bin` file.

The message `Preparing to Install` is displayed. After the installer decompresses the required files, the installation proceeds with the information specified in the installer properties file. When the process is successful, Installation Complete is displayed.

Exploring the Silent Mode Installer Properties File

Table 3-1 describes the required installer properties used with a Silent Mode installation.

Table 3-1 Installer Properties

Sample Property Setting	Description
`BEAHOME=d:\\bea` `BEAHOME=/bea`	BEA Home directory. The specified directory must exist on the system. For full installation options that include DSP, WebLogic Platform version 8.1 must be installed in the specified `<BEAHOME>` directory. For Windows, specify the absolute path, including the drive. Backslashes must be escaped. For UNIX, specify the absolute path.
`USER_INSTALL_DIR=D:\\bea\\weblogic81\\liquiddata` `USER_INSTALL_DIR=/bea/weblogic81/liquiddata`	Product installation directory. For Windows, specify the absolute path, including the drive. Backslashes must be escaped. For UNIX, specify the absolute path.
`COMPONENT_PATHS`	The components to install on the system. You can specify the following: • DSP **Runtime**. Installs DSP. • **Workshop Extension**. Installs DSP extensions to WebLogic Workshop. • **Samples**. Installs the DSP sample domains.

The following shows a sample Silent Mode installer properties file:

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Silent installer option: -mode=silent-silent_xml=/home/me/silent.xml
-->
<domain-template-descriptor>
      <input-fields>
              <data-value name="BEAHOME" value="C:\bea" />
              <data-value name="USER_INSTALL_DIR"
value="C:\bea\weblogic81" />
              <data-value name="COMPONENT_PATHS" value="Liquid Data
Runtime|Workshop Extension|Samples" />
</input-fields>
</domain-template-descriptor>
```

Post-Installation Tasks

This chapter describes tasks you can after installation of an instance of Data Services Platform (DSP), including verifying the installation, starting development of applications, and exploring Windows shortcuts and UNIX paths. The chapter also describes how to uninstall DSP, if required.

The chapter includes the following sections:

- Verifying the Installation
- Starting the Development of a Data Integration Solution
- Exploring DSP Windows Shortcuts and UNIX Paths
- Uninstalling Data Services Platform

Figure 4-1 illustrates the current step in the overall installation process.

Figure 4-1 Post-Installation Step
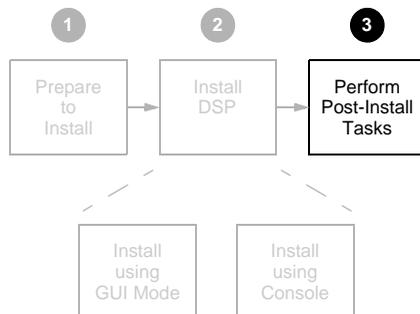
Verifying the Installation

After you have installed DSP, you can verify that the installation was successful by building the RTLApp sample application and testing a page flow using WebLogic Workshop.

To verify the installation:

1. Start the DSP sample domain using the Windows Start menu:

   ```
   Start →All Programs →BEA WebLogic Platform 8.1 →BEA AquaLogic Data
   Services Platform 2.1 →Examples →RTL Demo →Launch RTL Demo Sample
   Server
   ```

   Note: You can go on to Step 2 while the sample domain server is starting.

2. Open WebLogic Workshop and the RTLApp by executing the `RTLApp.work` file located in the following directory:

   ```
   <WL_HOME>/samples/liquiddata/RTLApp/RTLApp.work
   ```

3. Build the RTLApp.

   Right-click RTLApp in the navigation tree, and choose Build Application from the context-sensitive menu. WebLogic Workshop displays the status of the build. Confirm that the build was successful, as indicated by the build status messages.

4. Start a page flow.

5. Double-click on the Java page flow file `demoPageFlowController.jpf`. It is located in the following folder:

   `RTLSelfService\Pages`

6. With the page flow diagram in your work area choose the Start option from the Workshop Debug menu.

   `Debug →Start`

   The appearance of a login page in the Workshop Test Browser, as illustrated in Figure 4-2, indicates that DSP is correctly installed and running.

Figure 4-2 Login Page in the Workshop Test Browser
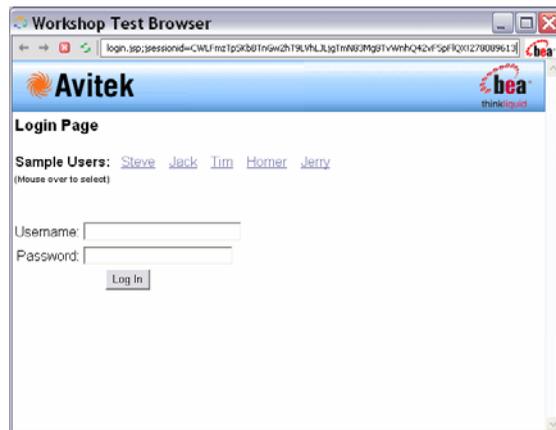
For an overview of the Retail Sample Application (RTLApp), including a description of data sources, queries, and other application components, see Chapter 6, "Sample Retail Application Overview."

Exploring DSP Windows Shortcuts and UNIX Paths

When installing DSP on a Windows system, the installer program uses the BEA WebLogic Platform home directory (*<WL_HOME>*) as the parent folder for WebLogic Server and BEA DSP. It then appears in the Start menu as follows:

```
Start →Programs →BEA WebLogic Platform 8.1 →
    BEA AquaLogic Data Services Platform 8.1
```

The DSP folder contains the following shortcuts:

Table 4-1 DSP Shortcuts and Paths

Item	Description
Examples →RTL Demo →Launch RTL Demo Sample Server	Starts the RTL Demo server by running the startWebLogic.cmd command in the following domain: *<BEA_HOME>*/weblogic81/samples/domains/ldplatform/
Examples →Other Sample →Launch Other Demo Sample Server	Starts the DSP Other Demo server by running the startWebLogic.cmd command in the following domain: *<BEA_HOME>*/weblogic81/samples/domains/ldplatform/
Examples →RTL Demo →PointBase Console	Launches the PointBase database console. In order to access the console, you need to enter the Samples URL, such as: jdbc:pointbase:server://localhost:9093/workshop You can determine the URL in use through the WebLogic Administration Console. In the Console select Service →JDBC →Connection Pools. All the pools use the same URL. The default user and password are both PBPUBLIC. For additional information see "How Do I: Administer the Default PointBase Database" in WebLogic Workshop documentation.
Examples →RTL Demo →WebLogic Server Console	Starts the WebLogic Administration Console for the DSP server that is currently running.
Examples →RTL Demo →DSP Console	Starts the DSP Console.

Table 4-1 DSP Shortcuts and Paths (Continued)

Online Documentation	Launches the DSP online documentation in your default Web browser. The path to Data Services Platform documentation is: http://e-docs.bea.com/aldsp/docs21/index.html
Uninstall Data Services Platform 2.01	Uninstalls Data Services Platform. After a full installation, you may need to manually remove some of the sample files that have been placed in: *<BEA_HOME>*/weblogic81/samples/domains/liquiddata/

Starting the Development of a Data Integration Solution

Developing your own data integration solution consists of a design phase and an implementation phase.

- When you are ready to start setting up and configuring your own data sources, refer to the Data Services Platform *Administration Guide*.

- When you are ready to start mapping source and target XML schemas and constructing queries using WebLogic Workshop, refer to *Data Services Developer's Guide*.

- When you are ready to create an application such as a Data Services Platform-enabled Workshop application, see the *Client Application Developer's Guide*.

Uninstalling Data Services Platform

You can automatically remove DSP from your system. Once DSP has been removed, you can manually remove any extra files, such as modified samples or other user-created files, which were not removed automatically by the uninstaller.

You should also undeploy any DSP applications before uninstalling DSP from your system. For information about undeploying, see the topic "Undeploying Deployed Applications" in the WebLogic Server documentation.

Note: If you are planning to reinstall a version of Data Services Platform make sure that the `<WL_HOME>` directory into which you plan to install that new version does not contain any files from a previously installed version of DSP.

To uninstall Data Services Platform:

1. Run the DSP uninstaller.

   – On Windows, choose the following:

   ```
   Start →Programs →BEA WebLogic Platform 8.1 →BEA AquaLogic Data
   Services Platform 2.0 →Uninstall BEA AquaLogic Data Services
   Platform 2.0.
   ```

   – On UNIX, run the following command in cases when you installed using GUI mode (as described in Chapter 2, "Installing Data Services Platform Using GUI Mode"):

   ```
   <WL_HOME>/<LD_HOME>/uninstall/uninstall
   ```

   Alternatively, run the following command with the "-i CONSOLE" option in cases when you installed using Console mode (as described in Chapter 3, "Installing Data Services Platform Using Console or Silent Mode")

   ```
   <WL_HOME>/<LD_HOME>/uninstall/uninstall -i CONSOLE
   ```

When the uninstall is complete, you may be notified that some files could not be removed by the automated uninstall. This could be a result of new files that were either generated or user-created after DSP was installed.

For example if you added new target schemas, stored queries, Web services, and so on to the DSP repository, the uninstaller will not remove these files. If you need these files, be sure to save them elsewhere before proceeding to the next step.

Note: The Uninstall option does not remove any Data Services Platform or other Workshop applications that you have created.

2. Remove remaining files, as appropriate.

You can manually delete remaining files by removing the directories specified in Table 4-2.

Table 4-2 Directories to Remove Manually Following an Uninstall

Directory	Platform	Location
DSP home directory	Windows	*<WL_HOME>**<LD_HOME>*
	UNIX	*<WL_HOME>*/*<LD_HOME>*
DSP samples directory	Windows	*<WL_HOME>*\samples\LiquidData
	UNIX	*<WL_HOME>*/*<LD_HOME>*/samples/liquiddata
DSP samples domain directory	Windows	*<WL_HOME>**<LD_HOME>*\samples\domains\ liquiddata
	UNIX	*<WL_HOME>*/*<LD_HOME>*/samples/domains/ liquiddata

Post-Installation Tasks

Migrating Legacy DSP Applications

The basic migration path for BEA AquaLogic Data Services Platform 2.0.*n* application and projects is straightforward. This section provides instructions for such upgrades including:

- Platform Installation and Configuration
- Data Services Platform 2.1 Installation and Deployment
- Data Services Platform Upgrade Options
- Migrating Earlier (2.0) Projects to DSP 2.1

Platform Installation and Configuration

Prior to installing Data Services Platform, WebLogic Platform 8.1 SP4 or SP5 must be installed and configured properly.

Data Services Platform 2.1 Installation and Deployment

In order to migrate an application DSP 2.0*x* you must first install DSP 2.1. In a production environment, that installation must be completed for each WebLogic Platform 8.1 domain that you are using with DSP.

Data Services Platform 2.1 Installation

Follow the appropriate preparation, installation, and post-installation steps described in other sections of this document starting with "Installation Prerequisites" on page 1-10.

Data Services Platform Upgrade Options

There are a several of steps to take when upgrading to the latest version of Data Services Platform.

- Upgrading Earlier (2.0) Applications to DSP 2.1
- Migrating Earlier (2.0) Projects to DSP 2.1

Upgrading Earlier (2.0) Applications to DSP 2.1

Applications can be migrated through WebLogic Workshop or using the DSP 2.1 Convert utility.

Using Workshop to Upgrade 2.0 DSP Applications

After you have installed DSP 2.1 you need to upgrade several JAR files in your existing DSP applications in order to deploy those applications under DSP 2.1. In WebLogic Workshop the update process is semi-automatic. When you open your application in WebLogic Workshop you will get a message indicated that some DSP JAR files are out-of-date and asking if you want to update your application.

Choose Yes to update the files. This will automatically cause your application to be built or rebuilt.

Note: If you choose No, your 2.0 application will not be updated and you will not be able to deploy it under DSP 2.1.

Using the Convert Utility to Upgrade Legacy DSP Applications

If your Data Services Platform 2.0 application is already packaged in an EAR file you can use a command-line utility to upgrade the EAR to DSP 2.1. The utility inspects the EAR file for out-of-date copies of the following:

- `ld-server.app.jar`
- `LiquidDataControl.jar`

If pre-DSP 2.1 versions of these files are found, they are replaced with 2.1 versions.

Upgrading EAR Files on Windows

To upgrade a Windows-based DSP 2.0 EAR file to DSP 2.1 using the Convert migration utility following these steps:

1. Back up your EAR file.

2. Open a command window using Window's Run command.

3. Set the WebLogic Server environment, if it is not already set on your system:

   `<WL_HOME>\server\bin\setWLSEnv.cmd`

4. Run the Convert command-line utility on your EAR file. For example:

   `<WL_HOME>\liquiddata\bin\convert.cmd myEAR.ear`

Upgrading EAR Files on UNIX Systems

To upgrade a UNIX-based DSP 2.0 EAR file to DSP 2.1 using the Convert migration utility following these steps:

1. Back up your EAR file.

2. Set the WebLogic Server environment, if it is not already set on your system:

   `<WL_HOME>/server/bin/setWLSEnv.sh`

3. Run the Convert command-line utility on your EAR file. For example:

   `<WL_HOME>/liquiddata/bin/convert.sh myEAR.ear`

Migrating Earlier (2.0) Projects to DSP 2.1

Earlier versions of Data Services Platform managed functions returning sequences by generating additional "arrayOf" types (`.xsd` schema files) in schema projects. For example, when a data service was created around an XMLtype of customer (`customer.xsd`) an XML type arrayOfcustomer (`arrayOfCustomer.xsd`) was created automatically. Applications calling functions returning sequences could then utilize the arrayOf type.

This approach to handling functions returning sequences has been deprecated. Instead, functions returning sequences of records are handled by converting each item in the sequence into a standard data object (SDO). With DSP 2.1 Java arrarys of SDO are supported. (For details on SDO and handling of sequence SDOs see the *Application Developer's Guide*.)

If your Data Services Platform project was created using an earlier version of DSP you have a choice:

- By default your legacy project will utilize the original arrayOf technique for returning sequences through functions.

- Or you can migrate your project to the improved means of handling sequences on a project-by-project basis. This is done through the Migrate Data Services Project menu option.

You can access the Migrate Data Services Project option by right-clicking on your legacy project (see Figure 5-1).

After migrating projects existing Java mediator client code that relied upon the ArrayOf types will need to be updated.

Figure 5-1   Upgrade Data Services Project Option
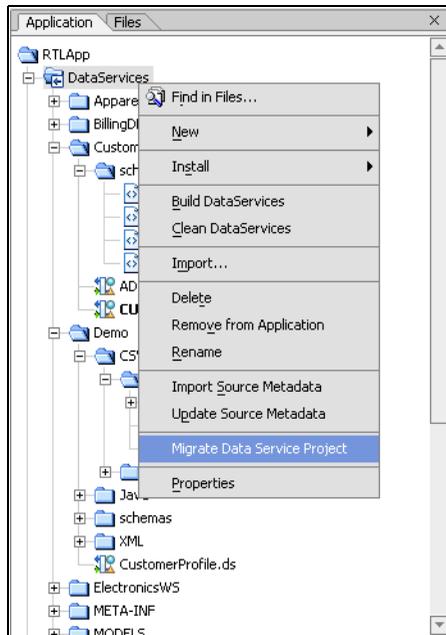
Note: Migration of projects to DSP 2.1 are not reversible.

Sample Retail Application Overview

This is a brief overview of the Avitek Sample Retail Application (RTLApp) that is included with a complete BEA AquaLogic Data Services Platform (DSP) installation. This chapter describes the data sources and other application components used to develop the RTLApp.

- Quick Start Instructions for the Avitek Sample Application (RTLApp)

- The Challenge of Disparate Data

- Business Case for the Avitek Self-Service Web Site

- Purpose of the Avitek RTLApp

- Running Retail Sample Application in a Browser

- Viewing RTLApp Source

- Where To Go From Here

About Avitek Ltd.

Avitek is an imaginary retailer that has grown through acquisitions. The company started out selling apparel but recently merged with an electronic retailer to expand business and consolidate cost centers such as accounting and web development. Because of this acquisition the company now has two very different order management systems (OMS), one each for electronics and apparel orders. Avitek also has a customer relationship management (CRM) system to manage customer profile information. Finally, Avitek has a Customer Service system to manage the support cases for Electronic products.

General IT Goals

General immediate and mid-range goals for the newly consolidated IT department include:

- An easy solution to providing integration of different back-end resources.

- An easy solution to providing read/update to these back-end resources.

- An abstraction layer to hide the complexities of interacting with different systems.

Specific Projects

An early "low-hanging fruit" project is to build a customer self-service web site which can also be used by support representatives. This web site needs to provide an easy solution to integrating different back-end resources and to providing two-way communication (read/update). It will also need to provide an abstraction layer to hide the complexities in the different systems.

The front-end architecture of the web site will provide a Java-based Customer Self-Service portal where customers can update their profile information and review or modify their orders. The back-end architecture will access the following five different sources of data (as shown in Figure 6-1):

- Customer related information stored in a CRM database (Oracle)

- Credit card information from a billing system (Sybase)

- Case information from a support database (DB2)

- Apparel order information from the company's Order Management system (SQL Server)

- Electronics Order System accessed through a web service. (The original source of information is mainframe inherited with the merger.)

The project also needs a code name. RTLApp is selected.

Figure 6-1RTLApp Front and Back End Architecture
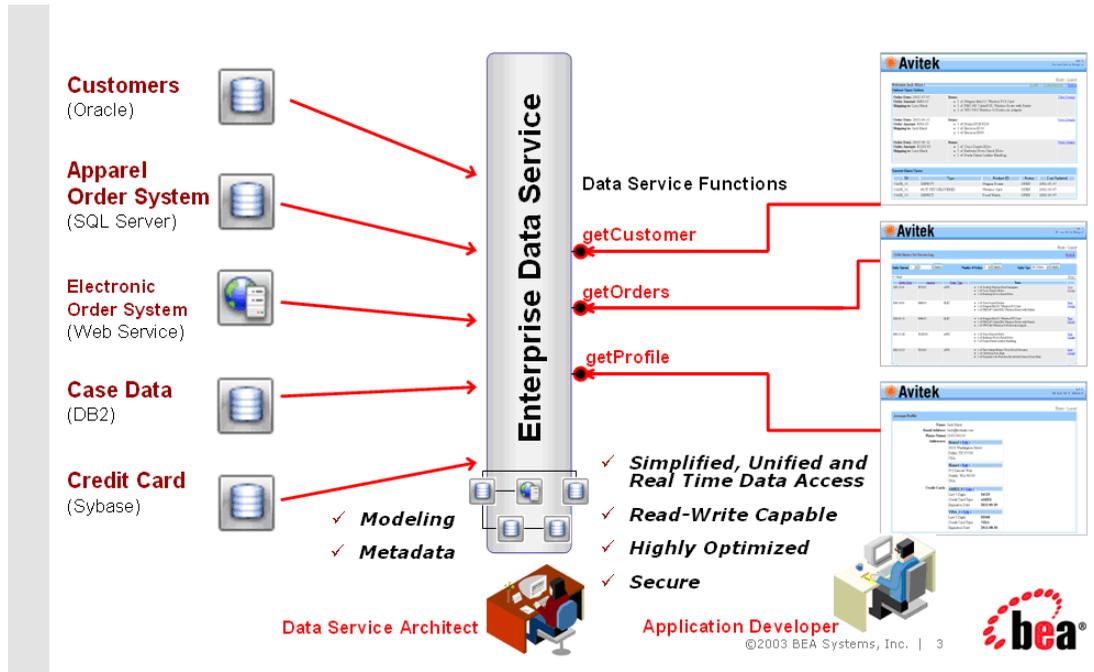
Quick Start Instructions for the Avitek Sample Application (RTLApp)

Make sure that both the RTLApp development environment and Web application are available. To start the RTLApp in the IDE, follow the instructions outline in , including starting your BEA WebLogic Server.

The steps involved in running RTLApp from inside WebLogic Workshop are similar to starting the IDE. (If you have already build and deployed RTLApp in Workshop you can skip to step 4.)

1. Start the DSP sample domain using the Windows Start menu:

   ```
   Start →All Programs →BEA WebLogic Platform 8.1 →BEA AquaLogic Data
   Services Platform 2.1 →Examples →RTL Demo →Launch RTL Demo Sample
   Server
   ```

   Note: You can go on to Step 2 while starting the sample domain server.

2. Open WebLogic Workshop and the RTLApp by executing the `RTLApp.work` file located in the following directory:

   `<WL_HOME>/samples/liquiddata/RTLApp/RTLApp.work`

3. Build RTLApp.

   Right-click on RTLApp (top node in the Application pane), and choose Build Application from the context-sensitive menu. WebLogic Workshop displays the status of the build. Confirm that the build was successful, as indicated by the build status messages.

4. Double-click on the Java page flow file `demoPageFlowController.jpf`. It is located in the following folder:

   `RTLSelfService\Pages`

5. With the page flow diagram in your work area choose the Start option from the Workshop Debug menu.

   `Debug →Start`

   Log in using one of the names shown in the dialog box. Simply move your mouse over the login name of your choice to automatically fill-in the name and password (Figure 6-2).

Figure 6-2 RTLApp Login Page
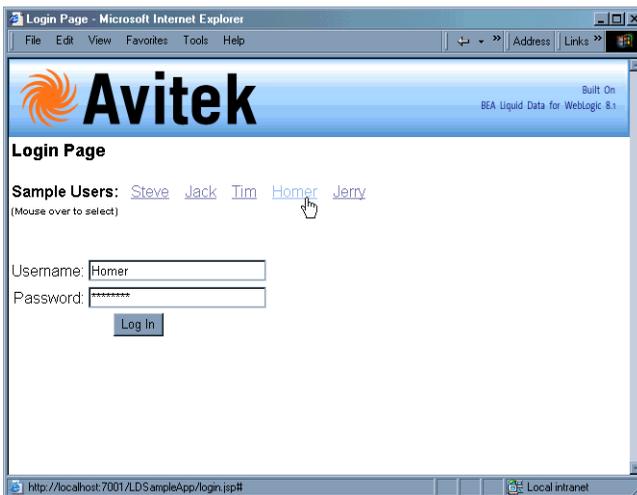
The Challenge of Disparate Data

The sample retail application illustrates in simplified form the kinds of data integration challenges often encountered by Information Technology (IT) managers and staff. Issues include:

- What is the best way to normalize data drawn from widely divergent sources?

- Having normalized the data, can you access it, ideally through a single point of access.

- Having such a single point of access to your data, can you develop reusable queries that are easily tested, stored, and retrieved?

- Once you can query data, can you as easily update it in a secure manner?

- Can you cache data to improve performance while protecting users from stale information?

- How difficult is it to consume query results in client applications?

Other questions may occur. Is the data-rich solution scalable? Is it reusable throughout the enterprise? Are the original data sources transparent to the application — or do they become an issue each time you want to make a minor adjustment? When changes to underlying data inevitably occur, how difficult will it be to propagate those changes to the application layer?

So many questions...

Business Case for the Avitek Self-Service Web Site

A survey commissioned by Avitek Marketing found customers to be dissatisfied with the call-in wait time required to track orders or update customer information. In a focus group the idea of a self-service web site resonated. Customer Service agreed; they have been requesting such a site for years, but the internal costs were always above budget. But now that Marketing is on board …

Web Site Design Requirements

Site requirements seem simple. Fulfillment identifies that customers will need to be able to:

- **Securely log in and out.**

- **Review order status.** A Home page will provide information on open orders, including product names and quantities of items ordered, order amount, shipping instructions, and a summary of any open customer service cases, with details.

- **Review order history.**

- **Review and change personal profile information.**

- **Search through orders.** A sample page provides the ability to retrieve orders based on a range of prices, range of dates, etc.

Bottom line: If customers can perform this level of self-service the company will save a lot of money.

Maintenance Requirements

A cross-sectional team of marketing and customer service develop maintenance requirements for the web application:

- 7x24

- Less than 10 pages

- Low-maintenance

- Easily modified

Design Requirements

An application/UI designer begins "specing out" the required JSPs. Pages are the easy part. Data is needed so he shoots off an email to the consolidated IT department.

Information Technology (IT) Weighs In: The Moment of Truth

When an IT data architect analyzes the requirements she turns up a problem. In surveying the information needed by the application -- customer data for one data source, order data from two very separate divisions of the company (two more data sources), and customer support data (a fourth data source) — the architect realizes that integrating data from these diverse data sources will be complicated and time consuming with additional maintenance problems down the road. Challenges included:

- **Cost of development.** Writing the code to access and integrate data from multiple diverse data sources would take more time than originally expected and require more expensive and scarcer resources.

- **Time to market**. Developing and testing an application against multiple data sources would extend beyond the date when the application is needed.

- **Cost of testing and maintenance.** High.

Perhaps most frustrating: little of the specialized code needed by the application can be reused.

So much for low-hanging fruit.

Search for an Alternative

Developing a unified view into distributed data is one of the most persistent challenges faced by IT departments. Just when you get all the available data sources normalized, new sources appear that must be dealt with, but which also make yesterday's data integration solution obsolete.

This problem is so pervasive that each year thousands of arguably critically-needed applications go unwritten, are delayed, or are delivered in highly-compromised form because of the data integration challenges faced by even the most sophisticated enterprises.

Compared to the above, the RTLApp team preferred a solution that:

- Provides a layer of data abstraction so that queries can treat highly-divergent data sources as a single, virtual data source.

- Allows development of human-readable, reusable queries.

- Can be easily accessed by consuming applications through a simple API.

- Protects the integrity and security of the underlying data.

- Allows read-write testing and deployment.

A Possible Solution

When Avitek looked at Data Services Platform, they found a product that addressed the underlying challenges posed by the *apparently* simple RTLApp:

- Addressing the problem of data access, DSP provides data integration through a highly-accessible graphical interface.

- In DSP, queries are developed graphically and are self-describing.

- Once data integration is achieved, persistent queries are generated.

- Read-write to live or staged data is available at every step of the development process.

Specifically, the features that the team found most appealing included:

Data Access. DSP allows the application to access information from anywhere in the company — or beyond — through an easily-created *virtual data access layer*. Once accessed, data can easily be aggregated through a combination of reusable queries and views that are maintained in the DSP server.

Query Development. Then, once the data is collected under a single point of access, it is not difficult to create query functions that consolidate data from these disparate sources and present a common, reusable view ready for more specialized queries.

The declarative form of DSP artifacts (query functions in data services) makes them very readable.

Query Deployment. Once developed, queries are easily integrated into a client application thorough a variety of access methods such as the DSP Mediator API, a data service control, JDBC, or SQL.

Integration. Business logic for the RTLApp is provided by the NetUI and page flow features of Workshop. However, DSP query functions could have as easily been integrated and maintained within the business logic of any J2EE application.

Purpose of the Avitek RTLApp

The RTLApp is designed to illustrate how DSP-generated queries can aggregate data from potentially highly disparate data sources, allowing access to that data through a single point of access that itself is easily integrated with the application.

Note: To simplify the running of the RTLApp, the multiple data sources described in this document are simulated using the PointBase RDBMS which is shipped with DSP. In the original implementation, these databases were represented by major vendor RDBMS systems.

Finding the Components

The RTLApp is located in the following directory:

    <WL_HOME>/samples/domains/liquiddata/RTLApp

RTLApp controls, pages, processes, and resources used to create the SampleApp can be found at:

    <WL_HOME>/samples/domains/liquiddata/RTLSelfService

Some of the schemas used in the RTLApp are in the following directory:

    <WL_HOME>/samples/domains/liquiddata/schemas

Others are located in their respective data service.

The Workshop `.work` file for the SampleApp is located at:

    <WL_HOME>/samples/domains/liquiddata/RTLApp/RTLApp.work

When you install DSP with the WebLogic Platform, the source code for the RTLApp can be accessed from Workshop. For instructions, see "Viewing RTLApp Source" on page 6-21.

For additional information and references, see:

- "Running Retail Sample Application in a Browser" on page 6-14.

- "Using Workshop Controls to Develop Data Service Platform Applications" in the *Application Developer's Guide* for the specific steps required to create the data service control.

Analyzing the RTLApp Architecture

Several BEA technologies are exercised by the RTLApp.

- **Query Development.** Data services and queries that draw data from multiple data sources are created in WebLogic Workshop.

- **Query Access.** Query functions are accessible through the DSP mediator API, JDBC, and the Workshop data service control and can be tested within Workshop.

- **Client-side Development**. Workshop also provides an environment for building application logic through NetUI, page flow, and other technologies.

Available Data Sources

Although the RTLApp is very simple, the underlying data acquisition is challenging because date comes from four heterogeneous data sources. These are:

- **Customer Relationship Management (CRM) system.** CRM data (customer and credit card information) is stored in a database called RTLCUSTOMER.

- **Order Management System (OMS).** Avitek has two order management systems:
  - **Apparel products.** Information is maintained on site in a second database. This is represented in DSP as RTLAPPLOMS.
  - **Electronic products.** OMS information is available from a legacy system (RTLELECOMS) via a web service. The web service has several methods such as getCustomerOrderByCustomerID(), that takes a customer ID as input and returns a list of customer open order information through a web service.

- **Customer Service.** Service data is stored in a third database. The schema for this data is RTLSERVICE.

- **Billing information.** Credit card information is maintained in a separate, highly secure database, RTLBILLING.

Retail Sample Application Queries

The following section describes work done by some of the RTLApp data services.

Physical Data Services

There physical data services correspond with the physical data sources needed by the application.

- ApparelDB includes the following data services: CUSTOMER_ORDER, CUSTOMER_ORDER_LINE_ITEM, and PRODUCT.

- BillingDB includes a CREDIT_CARD data service.

- CustomerDB includes ADDRESS and CUSTOMER data services.

- ElectronicsWS includes data services based on the web services. The data services include: getCustomerOrderByCustomerID, getCustomerOrderByOrderID, and getProductList.

- ServiceDB is the source for the SERVICE_CASE data service.

In RTLApp physical data services typically have a read function and, often, one or several navigation functions that correlate to the primary key/foreign key relationship between relational sources.

Logical Data Services

Logical data services (services based on physical data services or other logical data services) are located in a folder called RTLServices. It is in these logical data services that read and navigation functions drawing on multiple data sources are developed and maintained.

Avitek Customer Self-Service Sample Application

Table 6-1 shows the relationship between the data sources, major data elements (for RDBMS systems: tables and columns), sample application JSPs, and the RTLApp web pages.

Table 6-1 RTLApp Primary Pages and Their Data Sources, Primary Data Elements, and Associated JSPs

Application Page	Data Sources	Primary Data Objects	JSP
Profile page	• Customer Relationship Management (CRM) RDBMS (Oracle) • Credit card data (Sybase)	from RTLServices/Customer • Customer profile • Bill-to, ship-to address from RTLServices/CreditCard • Credit card information	`ProfileView.jsp`
Open Orders page	• Customer Relationship Management (CRM) RDBMS • Order Management System (OMS) RDBMS • OMS via a web service	from RTLServices/Customer • CustID, name from RTLServices/ApplOrder • Order summary information from RTLServices/ApplOrderDetail View • Order detail information from RTLServices/ElecOrder • Order summary information from RTLServices/ElecOrderDetail View • Order detail information	`DefaultView.jsp`

Application Page	Data Sources	Primary Data Objects	JSP
Support	**For apparel order detail:** • Order Management System (OMS) RDBMS • Customer Relationship Management (CRM) RDBMS **For electronics order detail:** • OMS via a web service • Customer Relationship Management (CRM) RDBMS	**from RTLServices/Customer:** • CustID, name **For customer support cases:** • case status	caseView.jsp
Order History page	Same as Open Orders page	Same as Open Orders page	OrderHistory.jsp
Search page	Same as Open Orders page	from RTLServices/ApplOrder • Description • Order date • Order amount from RTLServices/ElecOrder • Description • Order date • Order amount	OrderSearch.jsp

Running Retail Sample Application in a Browser

If you have not already done so, use "Quick Start Instructions for the Avitek Sample Application (RTLApp)" on page 6-3 to start Retail Sample Application. You can choose from users Steve, Jack, Tim, Homer, or Jerry. Each uses the password `weblogic`.

Once a customer logs in, she or he sees their MyProfile screen. From there he or she can navigate to information on open orders, order history, support, search, and logout. Customers can edit open orders (see Figure 6-4) and get details on completed orders. Search allows the user to supply product description information, start or end date, or order amount brackets.

The application also demonstrates some DSP facilities including the ability to:

- Refresh data

- Restrict or unrestrict access

- Make a data source unavailable

- Enable Cache

- Show SQL reports created from Crystal Reports

- Update data

My Profile Page in RTLApp

The My Profile page illustrates DSP's ability to perform automatic read/write on distributed data.

Figure 6-3 My Profile Page

My Profile page consolidates Customer and Credit Card information. Users can change their personal profile information, address information, and credit card information. Changes are initially reflected on their MyProfile page. If satisfied, the user can click Submit All Changes to persist changes to the respective data sources.

Page Design

The page is based on `ProfileView.jsp` and the ProfileView data service (`ProfileView.ds`).

Open Order Page in RTLApp

The RTLApp Open Order page consolidates a customer's electronics and apparel open orders.

Figure 6-4 RTLApp Open Order Page

Data Sources

From the user's perspective, changing data is simply a matter of select and type. However, the underlying update mechanisms for electronic orders (which are maintained as a web service) and apparel orders are quite different.

Update Mechanisms

Electronic orders are derived from a web service. In this case, updating a electronic order demonstrates web service custom update capabilities.

Apparel orders are derived from a relational database and updates are automatic.

Caching Options

The Enable Cache option turns on caching for the function underlying the Open Orders page. You can use the Refresh button to verify that the execution time when cache is enabled is significantly faster than when it is not.

Handling Unavailable Sources

Click the Make Electronics Source Unavailable button to disable the web service. This action also refreshes your Open Orders page. Notice that you can still retrieve partial results (apparel orders) when a data source becomes unavailable.

Access LD via JDBC

The Show SQL Report button illustrates accessing data through JDBC. This demonstrates the integration of DSP query functions with reporting and business intelligence tools such as Crystal Reports.

Page Design

The Open Order page is controlled by a JSP named `defaultView.jsp`. Call-outs in Figure 6-4 show underlying data sources. The page derives its font and other look-and-feel characteristics from a cascading stylesheet.

Order History in RTLApp

The RTLApp Order History page displays historical order information for electronic and apparel orders.

Figure 6-5 RTLApp Order History Page

Users have several options associated with viewing their previous orders:

- Items can be ordered by date, amount, or order type (electronic or apparel).

- A operational filter can be applied based on order amount.

- The user can restrict the number of orders retrieved to a pre-set amount (5, 10, 15, 20, or all).

- The user can restrict orders to apparel or electronic.

An Apply button is provided to control page refresh.

Security

DSP security can be demonstrated when the number of orders is set to All. Then when the Restrict Access option is selected, data is automatically *redacted* to show only orders where order amounts are less than $500.00.
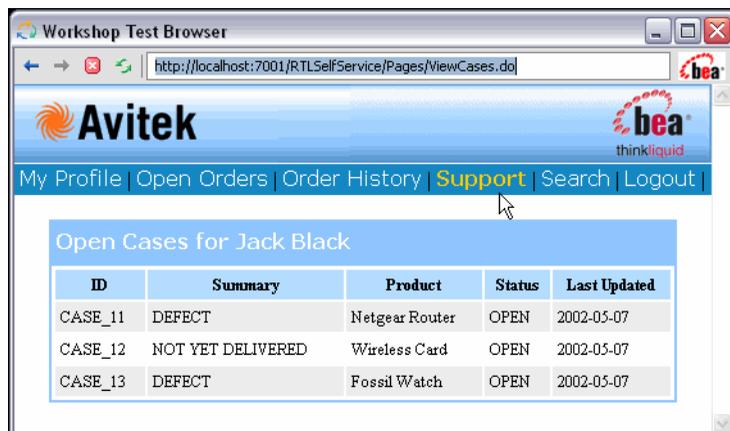
Page Design

The Order History page is controlled by a `orderHistory.jsp`.

Support Page in RTLApp

The Support page provides information on any open support cases for the currently logged-in customer.

Figure 6-6 RTLApp Customer Support Page

The underlying JSP is `CaseView.jsp`.

Search Page in RTLApp

Users can search for specific orders based on order dates, items, or amounts. The underlying code executes ad hoc DSP query functions.

Figure 6-7 RTLApp Search Page
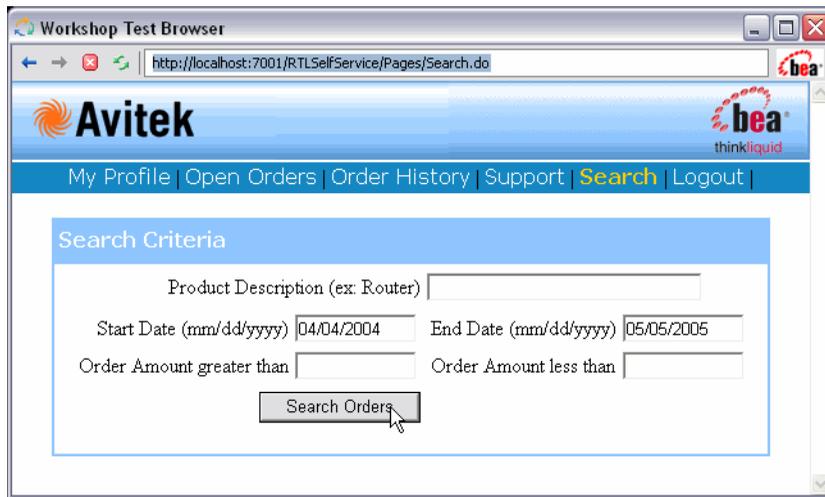
Users can search on any combination of search field criteria including product description, start or end data, and a range of order amounts.

The JSP for this page initiates a small Java program that incorporates customer input and generates an XQuery based on the selected parameters.

Viewing RTLApp Source

The data services, functions, JSP pages, and connection logic for the RTLApp were created in Workshop. You can view the source for the RTLApp in Workshop as well.

To open the RTLApp application, open the `RTLApp.work` file in Workshop. The full path is:

```
<WL_HOME>/samples/domains/liquiddata/RTLApp/RTLApp.work
```

Note: If you are already running WebLogic Workshop or have previously run it with a different server setting, you may need to change your application server properties settings. If necessary, select the following from the WebLogic Workshop menu:

Tools →Application Properties →WebLogic Server

When opening the RTLApp in WebLogic Workshop, you initially see the application components and the work area.

Figure 6-8 Initial Retail Sample Application Initial Project View
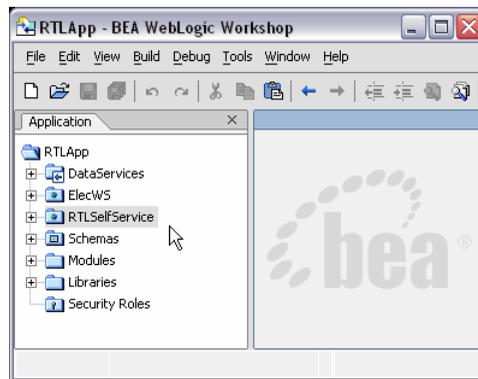
WebLogic Workshop allows you to work with the application you have under development as components or files. Web pages can be displayed and run, page flows and application logic can be developed and tested.

Note: Getting Started with WebLogic Workshop fully describes the IDE and how it can be used to build enterprise applications. The on-line document includes numerous examples and tutorials.

The WebLogic Workshop Samples application contains a number of samples relevant to data access and XML. See:

```
<WL_HOME>/samples/workshop/SamplesApp/SamplesApp.work
```

WebLogic Workshop Components of the RTLApp

Like all WebLogic Workshop applications, components of the RTLApp are arranged in folders. This section briefly describes some of these folders and their contents.

Data Services Folder

The Data Services folder contains both physical and logical data services.

Figure 6-9 Major Components of the RTLApp DataServices Folder
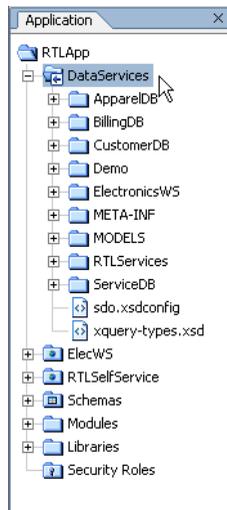
Physical data services, which are based on physical data sources, include apparel orders (ApparelDB), electronic orders (ElectronicsWS), credit card information (BillingDB), and customer service information (ServiceDB). Logical data services are all grouped in the RTLServices folder.

Model diagrams based on both physical and logical data services are contained in the MODELS folder. Of particular interest is the EnterpriseDataModel, which models all the physical data services in the application. Logical model diagrams include the RTLApp model, which summarizes the data service relationships that make up the RTLApp.

Note: The DataServices/Demo directory contains simple examples of delimited file data, XML file data, and Java functions being used as the basis for data services.
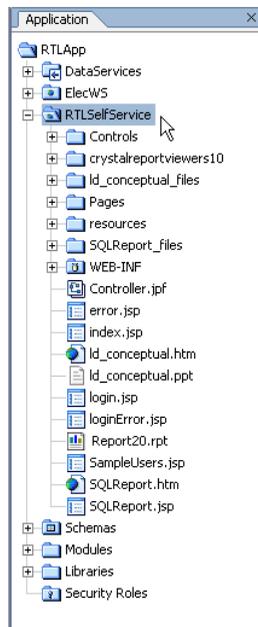
ElecWS Folder

The ElecWS folder contains the web service that serves as the basis for the electronics orders data service used by RTLApp.

RTLSelfService Folder

Many of the building blocks of the Retail Sample Application are available from the RTLSelfService folder.

Figure 6-10 Major Components of the RTLApp RTLSelfService Folder

The following components are of special interest:

- **Controls folder.** The Controls folder the RTLApp/RTLSelfService folder contains the `RTLControl.jcx` file. This is a file generated by the data service control that contains methods for each query function used in the DSP application. In Design View each function is listed. Clicking on a name switches you to Source View where you can see details about the function including:

  – Qualified name (qname) of the function's data service

  – Function name

  – Schema URI

  – Schema root element

  – Function call and parameters

  For additional information on the RTLControl see "Controls" on page 6-25.

- **cryistalreportviewers10 folder.** Contains files related to Crystal Reports 10.

- **ld_conceptual_files folder.** Contains conceptual information about **DSP** that appears at the bottom of some of the RTLApp pages.

- **Pages folder.** Contains the JSPs and the page flow controller, `demoPageFlowController.jpf`, that make up the RTLApp web application.

  For additional information on the pages see "Application Pages" on page 6-27.

- **resources folder.** Contains common resources such as the HTML style sheet, graphics files and common JSPs such as headers and footers.

- **SQLReport_files folder.** Contains files related to SQL reports that can be generated through applications such as Crystal Reports.

- **WEB-INF folder.** Contains application source components, in particular struts and NetUI components that come with WebLogic Workshop.
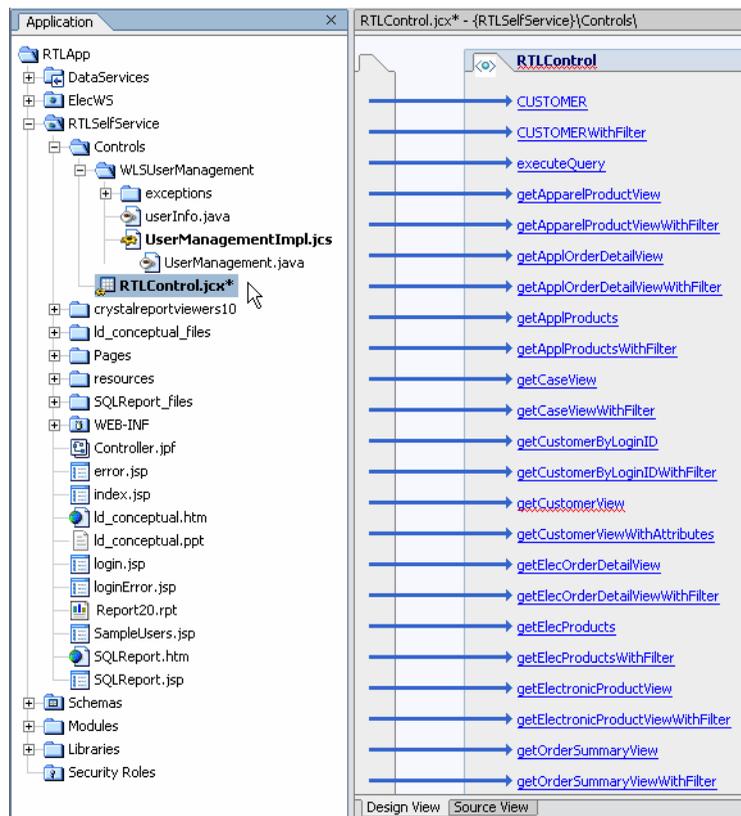
Schemas

The RTLApp/Schemas folder contains schemas developed for logical data services and associated XML Bean classes.

Controls

The `RTLControl.jcx` contains automatically-generated methods based on a set of stored queries selected by the developer of the data service controls. For information on accessing DSP queries in WebLogic Workshop see the section "Select Data Service Functions to Add to a Control" in Accessing Data Services from Workshop Applications in the *Application Developer's Guide*.

Figure 6-11 Design View of RTLControl Design View

A portion of the Source View of the RTLOrderSummary method of the RTLControl JCX file is shown in Figure 6-12. Note that the name of the target schema for the query appears in comments.

Figure 6-12 Data Service Control Source View


```
RTLControl.jcx - {RTLSelfService}\Controls\

    package Controls;

    import weblogic.jws.control.*;
    import com.bea.ld.control.LDControl;
    import com.bea.ld.filter.FilterXQuery;
    import com.bea.ld.QueryAttributes;

    /**
     *  @jc:LiquidData application="RTLApp" urlKey="RTLApp.RTLSelfService.Controls.RTLControl"
     */
    public interface RTLControl extends LDControl, com.bea.control.ControlExtension
    {

        /* Generated methods corresponding to stored queries.
         */

        /**
         *
         * @jc:XDS locator="ld:DataServices/RTLServices/ApplOrderDetailView.ds" functionName="submitApplOrderDetailView"
         */
        java.util.Properties[] submitApplOrderDetailView(retailer.ORDERDETAILDocument rootDataObject) throws Exception;

        /**
         *
         * @jc:XDS locator="ld:DataServices/RTLServices/ProfileView.ds" functionName="submitArrayOfProfileView"
         */
        java.util.Properties[] submitArrayOfProfileView(retailer.ArrayOfPROFILEDocument rootDataObject) throws Exception;

        /**
         *
         * @jc:XDS locator="ld:DataServices/RTLServices/ElecOrderDetailView.ds" functionName="submitElecOrderDetailView"
         */
        java.util.Properties[] submitElecOrderDetailView(retailer.ORDERDETAILDocument rootDataObject) throws Exception;
```

Each method in the data service control corresponds to a query function. Each method returns an XMLBean type. The XMLBeans are generated when the control is created, and are stored in the Libraries directory of the WebLogic Workshop application.

Application Pages

In the RTLSelfService/Pages folder you can find the Java Server Pages (JSPs) that make up the
RTLApp. These were constructed in WebLogic Workshop using queries RTLControl and NetUI
graphical elements.

Figure 6-13 JSP Pages in the RTLApp
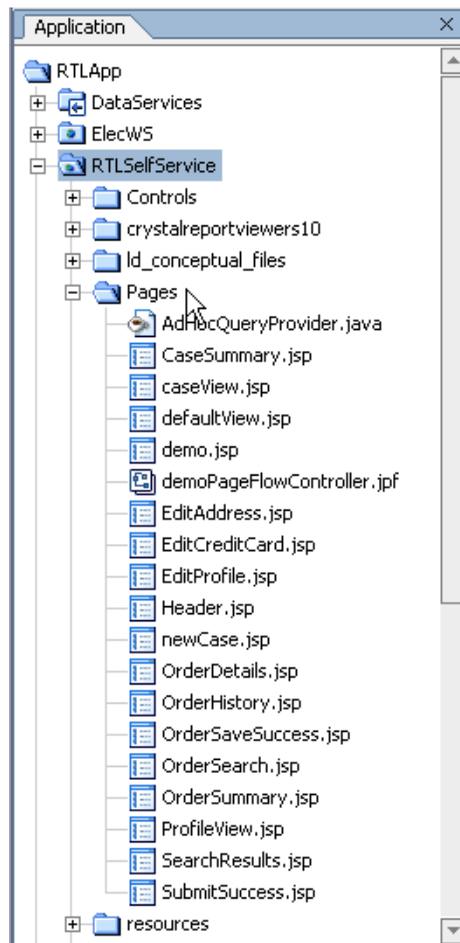
Application Logic: Page Flow

The RTLApp is composed of java server pages (JSPs) that are managed by a WebLogic Workshop PageFlowController. The PageFlowController is named `demoPageFlowController.jpf`.

From an application logic perspective, whenever a user releases control of a page by selecting an option such as Next, Previous, Ok, Cancel, and so forth, application logic returns to the PageFlowController. Once that logic is processed, the user sees the appropriate web page.

Using WebLogic Workshop you can inspect, change and extend page flow programmatically or graphically. There are three views of page flow: Flow View, Action View, and Source View.

Page Flow: Flow View

The Workshop page flow schematic (Figure 6-14) illustrates graphically the relationship between JSPs, including has modeless page flow is determined by user actions.

When you click on a particular page name, the page opens in the WebLogic Workshop development browser.

Figure 6-14 Retail Sample Application Page Flow

The three primary elements found in Page Flow View are:

- **Pages.** Clicking on a particular page will open the page for editing.

- **Actions.** The transition between pages is controlled by actions. Specific actions are user-driven such as in the case of a user accepting an option ("OK") or deciding against it ("Cancel") via a particular form or dialog box.

- **Source.** When source is selected, the code for the page appears in the work area.

Page Flow: Action View

The Action View tab helps in finding the location of page flow actions associated with the RTLApp.

Figure 6-15 RTLApp Action View
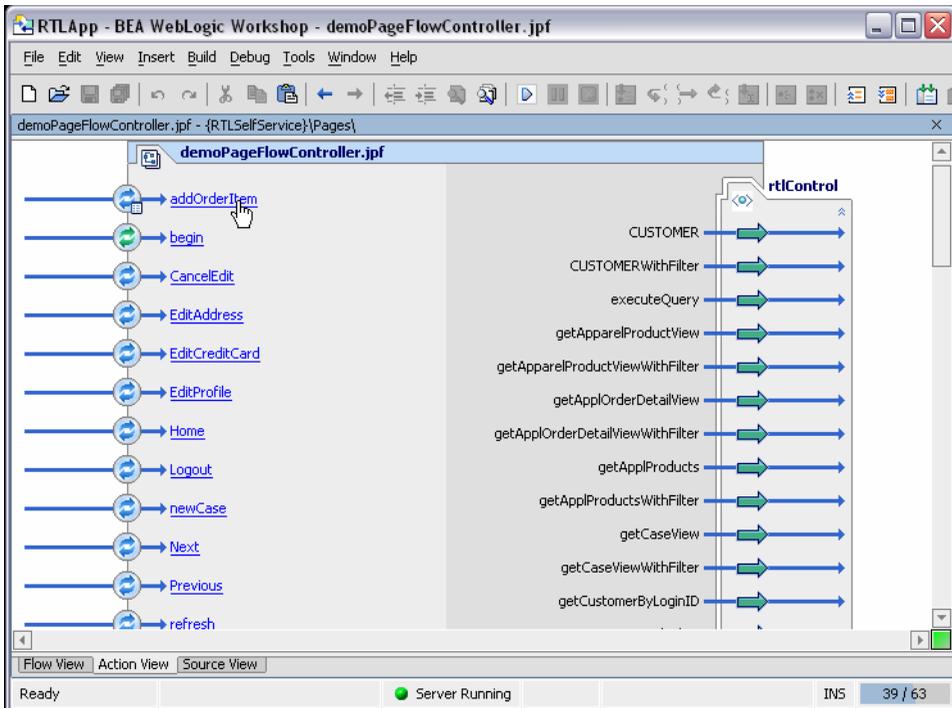
Clicking on an action items opens demoPageFlowController to the relevant section of code.

Page Flow: Source View

The PageFlowController file contains several parts:

- Declarations of graphical elements in the application.

- Public transient simple and array variables that facilitate and persist application logic between pages.

- @jpf:forward calls within comments that associate user actions with appropriate target JSPs.

- Queries and associated logic.

Summary

In summary, the RTLApp provides:

- A virtual data access layer that allows you to treat heterogeneous data as from a single source.

- Ability to access the data through declarative queries that can be created in the XQuery Editor or developed externally.

- Availability of DSP queries and server for easy integration into applications or processes.

Where To Go From Here

Here are some additional resources for learning more about Data Services Platform and WebLogic Workshop:

- The Data Services Platform *Concepts Guide* provides a comprehensive overview of DSP technology.

- The *Data Services Developer's Guide* explains how to create the data services, model diagrams, and query functions that are called by client applications.

- The *Client Application Developer's Guide* describes the several ways that client applications can access DSP read-write query functions.

- To learn more about XML Beans see *Getting Started with XML Beans*.

- To learn more about WebLogic Workshop see *Getting Started with WebLogic Workshop*.

- The WebLogic Workshop Samples application installed with WebLogic Platform contains a number of samples relevant to data binding and XML. See:

    <WL_HOME>/samples/workshop/SamplesApp/Samples.work