Paradox® 9

jPdox™ Web Utilities User Guide

TABLE OF CONTENTS

jPdox Web Utilities 1

Introduction

Retrieving data and publishing it over the Internet or an intranet is becoming increasingly important for enterprise markets. As the popularity of the internet increases, and as users become accustomed to retrieving information using Web browsers, many organizations are switching from document-based Web sites to database-based Web sites.

The new advanced features of Paradox® 9 provide an Internet or intranet solution, allowing users to design and publish forms, tables, and reports to intranets or to the Internet.

jPdox Web Utilities setup

The Setup Expert, created with InstallAnywhere, is a Java application which installs the jPdox™ Web Utilities. For information on each step of the Setup Expert, see “jPdox Web Utilities setup” on page 3.

Technical Support and Services

For information about Corel Support and Services, refer to the Corel Support and Services help file located on the Paradox 9 or WordPerfect® Office 2000 CD in the following folder: Corel\Paradox\Techsupp.hlp.

jPdox Web Utilities

Paradox offers the following components, which together constitute jPdox Web Utilities:

- Paradox® Web Form Designer—lets you create forms by using embedded JavaBeans without writing code. Java applets embedded in HTML files help you create platform-independent Web forms. You can share current data from a Web site that is associated with any Paradox-compliant database or any database which has a JDBC Driver. For more information, see page 9.
- Corel® Web Server— a fully functional Web server. It offers HTTP support (version 1.0 plus many 1.1 elements), file caching, contact (access) logging using the CERN/NCSA Common Log Format plus transaction logging, custom MIME types, and support for multiple IntraBuilder sessions (multithreaded). The Corel Web Server acts as an intermediary between Web browsers and compatible applications, such as IntraBuilder and Paradox, to transfer requests and responses between them. For more information, see the Corel Web Server online Help.
- Paradox® Java Database Connectivity (JDBC) Driver—gives you standard Structured Query Language (SQL) access to databases from Java programs, including the Web Form Designer. The Paradox JDBC Driver provides you with a direct connection to the Borland Database Engine (BDE), which provides access to most desktop- and server-based databases from Java programs. For more information, see page 11.
- Paradox® Report Server—lets you create dynamic HTML documents using the HTML Table Expert or the HTML Report Expert. This feature is compliant with standard Web servers that support Java servlets, such as JRun. As the data in a database changes, the information on the Web page updates to reflect the changes. You can also publish static HTML documents that are fixed and unchanging. For more information, see page 15.
- JRun—allows your Web server to be 100% servlet API-compatible and to run Java servlets. JRun is a combination of native code to interface directly to your Web server in the fastest way possible, as well as a collection of Java classes that provides the interface layer between your server and the servlets that you run. (JRun is only included in the Paradox 9 Developer's Edition). For more information, see page 17.
- JDBC Proxy Server—redirects all incoming and outgoing information between the JDBC Client and the JDBC Server; it acts as a security barrier. For more information, see page 25.

jPdox Web Utilities setup

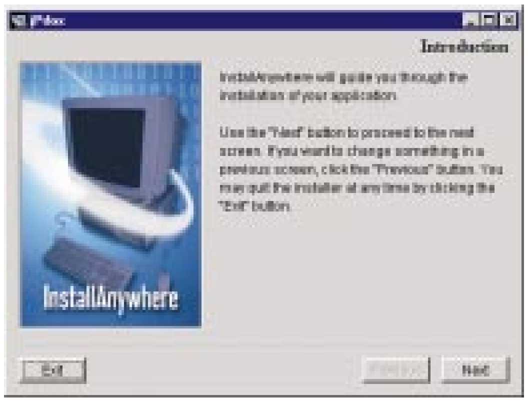
2

The Setup Expert, created with InstallAnywhere, is a Java application which installs the jPdox™ Web Utilities. The following section provides a brief description of the steps of the installation.

Step One: Introduction

The first step provides a brief introduction to the jPdox Web Utilities installation.

- Click next to continue.



InstallAnywhere will guide you through the installation of your application.

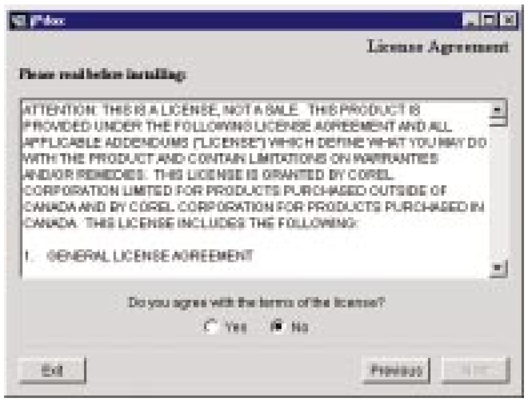
Use the "Next" button to proceed to the next screen. If you want to change something in a previous screen, click the "Previous" button. You may quit the installer at any time by clicking the "Exit" button.

Step Two: License Agreement

This is the end-user license. You must agree to this before continuing.

1 Click Yes to accept the terms of the license.

2 Click Next to continue.



ATTENTION: THIS IS A LICENSE, NOT A SALE. THIS PRODUCT IS PROVIDED UNDER THE FOLLOWING LICENSE AGREEMENT AND ALL APPLICABLE ADDENDUMS ("LICENSE") WHICH DEFINE WHAT YOU MAY DO WITH THE PRODUCT AND CONTAIN LIMITATIONS ON WARRANTIES AND/OR REMEDIES. THIS LICENSE IS GRANTED BY COREL CORPORATION LIMITED FOR PRODUCTS PURCHASED OUTSIDE OF CANADA AND BY COREL CORPORATION FOR PRODUCTS PURCHASED IN CANADA. THIS LICENSE INCLUDES THE FOLLOWING:

1. GENERAL LICENSE AGREEMENT

Do you agree with the terms of the license?

Yes [✓ No]

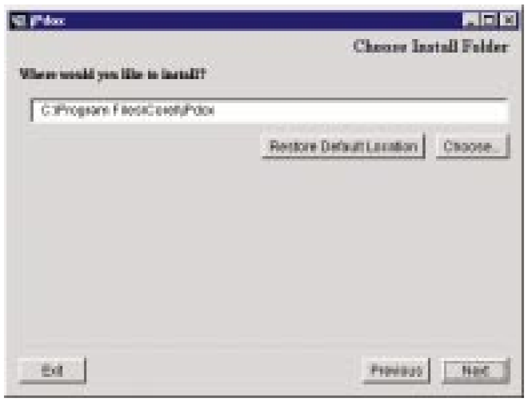
Step Three: Choose Install Folder

This screen allows you to choose where the program will be installed. The default folder is C:\Program Files\Corel\jPdox.

1 Type in the path of where you want to install jPdox Web Utilities, or click the Choose button to select a folder.

You can also click the Restore Default Location button to restore the default.

2 Click Next to continue.



Where would you like to install?

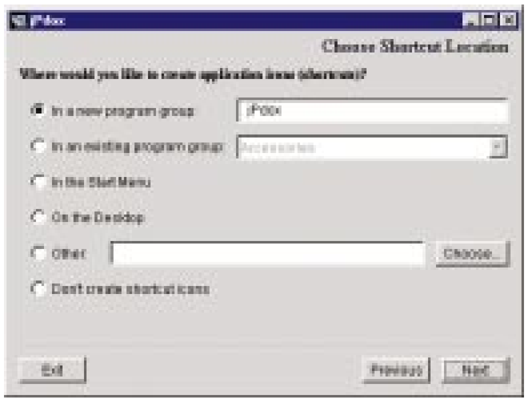
C:\Program Files\Corel\jPdox

Step Four: Choose Shortcut Location

This screen allows you to choose a program group. The default is jPdox.

1 Enable the option for where you would like the application icons created.

2 Click Next to continue.



Choose Shortcut Location

Where would you like to create application icons (shortcuts)?

- [✓ In a new program group] jPdox
- In an existing program group: Accessories
- In the Start Menu
- On the Desktop
- Other
- Don't create shortcut icons

Step Five: Choose Java Virtual Machine

This screen allows you to choose which Java Virtual Machine (JVM) will run your applications. If you already have the JRE 1.2 or JDK 1.2 (or later) installed, you can choose to select a JVM already installed on your machine.

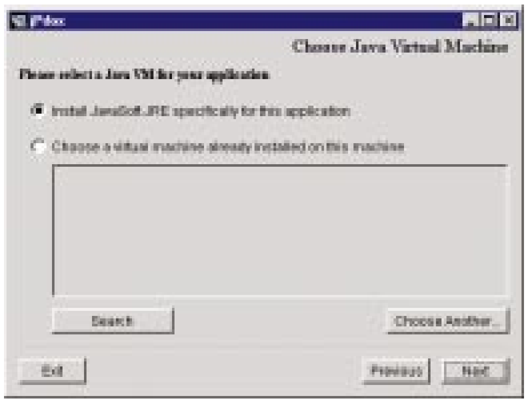
It is important that you select a JVM with a version of 1.2 or higher, if you are unsure of which version of JVM you have installed, choose to install the JavaSoft JRE specifically for this application (the default option).

1 Enable one of the following options:

- Install JavaSoft JRE specifically for this application.
- Choose a virtual machine already installed on this machine.

If you select to install a JVM already installed on your system, you can click the Search button, which will search your hard drive for all JDK or JRE virtual machines.

2 Click Next to continue.


Choose Java Virtual Machine

Please select a Java VM for your application

● Install JavaSoft JRE specifically for this application

○ Choose a virtual machine already installed on this machine

Step Six: Choose Install Set

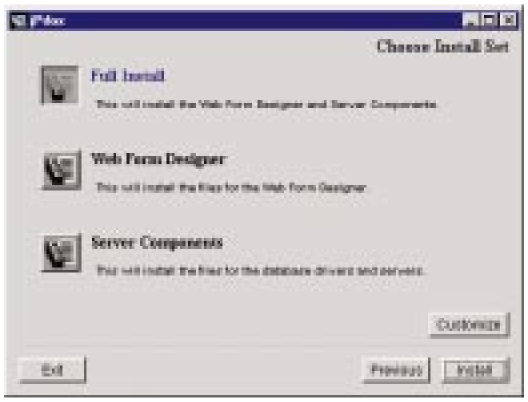
This screen lets you choose the components you wish to install.

1 Select one of the following install options:

- Full Install—installs the Web Form Designer, the server programs, and the JDBC Driver.
- Web Form Designer—installs only the Web Form Designer (if you only want to create blank forms without connecting to a database).
- Server Components—only installs the server utilities and drivers which allow previously created forms to connect to a database.

2 Click Customize if you wish an alternate combination of the install components.

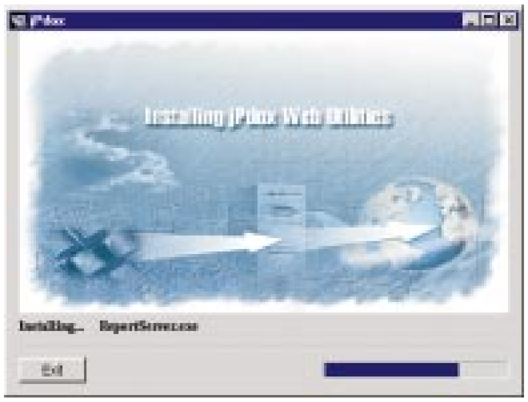
3 Click Next to continue.


jPdox
Choose Install Set
Full Install
This will install the Web Form Designer and Server Components.
Web Form Designer
This will install the files for the Web Form Designer
Server Components
This will install the files for the database drivers and servers.
Customize
Exit
Previous
Install

Step Seven: Installing

The next screen informs you that the selected jPdox Web Utilities are being installed.

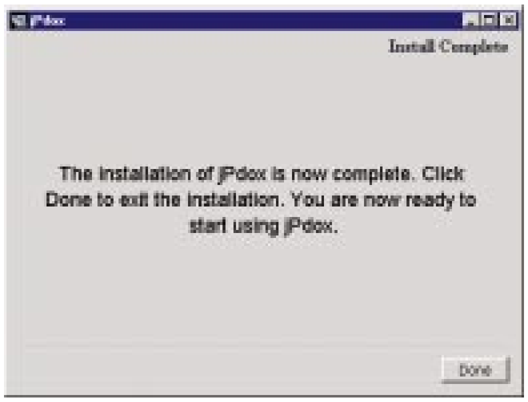
- Click Exit if you wish to end the install.


jPdox
Installing jPdox Web Utilities
Installing... ReportServer.exe
Exit

Step Eight: Install Complete

The last screen informs you that the install is complete.

- Click Done to complete the installation process.



The installation of jPdox is now complete. Click Done to exit the installation. You are now ready to start using jPdox.

Paradox Web Form Designer 3

The Paradox Web Form Designer is a new feature designed for Paradox 9. You can design Java-based Web forms to access databases on a network or the Internet. You can also design and create forms using Paradox 9, and publish them directly to the Internet or an intranet.

The Paradox Web Form Designer makes it easy to design and create Web forms from a Paradox 9 database. The design objects used to build the Web forms are based on predesigned JavaBeans. Java Applets are embedded in an HTML page to create forms. To create a form, you select and drop JavaBeans by dragging and dropping. The form is also WYSIWYG, so you do not need to preview your work in a Web browser before you publish. Once you have placed the JavaBeans in the Web form, you can Publish to HTML. The necessary files are copied to the local drive and a Web server. The new form is then displayed when the Web page is accessed.

- For more information on how to use Paradox Web Form Designer, see the Paradox Web Form Designer online Help.

Paradox JDBC Driver 4

Java Database Connectivity (JDBC) is an industry standard for database-independent connectivity between the Java platform and databases. The Paradox JDBC Driver is a developer tool used to connect databases. It provides a direct link between Java programs and the Borland Database Engine (BDE), which unifies database access across multiple platforms (Paradox 9 and the Paradox Web Form Designer come equipped with BDE 5.01).

The JDBC Driver is actually two parts: the JDBC Client and the JDBC Server. The Client consists of the Java files (PdxJDBC.jar) that need to be included in a Java application. The JDBC Server is the direct link to the BDE. The Client communicates transparently with the JDBC Server to transfer the data between the BDE and the Java application. Together, these two parts function as the JDBC Driver.

- For a list of unsupported JDBC class methods, see the Appendix on page 33.

How the Paradox JDBC Driver works

The JDBC Driver uses standard Structured Query Language (SQL) to provide Java developers with a consistent interface to desktop- and server-based databases. The Paradox JDBC Driver provides a direct link from a Java application to the BDE. A developer can use an existing alias, which is the name Paradox uses to identify different databases. This makes it possible for Paradox applications to be used concurrently with a Java application, such as the Paradox Web Form Designer.

Also, the Paradox JDBC Driver supports any database that the BDE supports, from dBASE and Microsoft Access tables to database servers such as Oracle and the Microsoft SQL Server.

- The Microsoft Access driver with the BDE uses DAO (Data Access Objects). Since DAO is not threadsafe, using the Microsoft Access driver with the JDBC server can create problems. However, Microsoft Access database can be accessed using the ODBC driver.

Configuring the Paradox JDBC Server

When connecting to a database using the Paradox JDBC Server, you may want to use different settings than those defined in the Paradox JDBC Server properties file, appsrv.properties. By editing the properties file, you can configure the setup of the Paradox JDBC Server specific to your setup requirements (editing parameters such as the Session Manager, registry ports, lease time, default logging, and log file path.)

The appsrv.properties file is located in the following directory:

```
root directory\WebUtilities\appsrv\appsrv.properties
```

Session Manager

A Session Manager object is created by the Paradox JDBC Server; the JDBC client layer hooks up to the server using this object. JDBC URLs used for opening connections would refer to a Session Manager object; by default it is named as SessionMgr.

Port Information

The Paradox JDBC Server uses the RMI (Remote Method Invocation) registry service, which listens on port 1099 by default. If there is a RMI registry already running, the Paradox JDBC Server will try to use it; otherwise, it will try to create a local registry. If the RMI registry is listening on a port other than the default, the port on which the RMI registry would listen would have to be made known to Paradox JDBC Server using the configuration settings.

The Paradox JDBC Server also listens on two other TCP ports (default values are 1100 and 1101). These can also be configured.

Lease Time

The Paradox JDBC Server uses a framework of objects that are accessed from the JDBC client layer. As long as the JDBC Client holds references to these objects, the objects at the server stay alive. A lease time attribute is associated with the server objects, which is renewed as long as the client layer uses these objects. If there is an abnormal shutdown of the client, the lease period expires and the server objects become defunct. This default lease time is 10 minutes and can be configured.

Log File

The log file configuration simply tells the Paradox JDBC Server where the log file is located. When you configure the log file path, you must enter all backslashes as "\\".

The following is an example of the properties file (appsrv.properties) used by the Paradox JDBC Server for configuration:

```
## Name of the SessionMgr Object
com.corel.pdx.appserver.SessionMgrName=SessionMgr

## Registry port for the Local Registry
com.corel.pdx.appserver.RegPort=1099

## ports used by AppServer objects
## if set to 0, will take next available port
## default value for AppSrvPort1 = 1100
## default value for AppSrvPort2 = 1101
com.corel.pdx.appserver.AppSrvPort1=1100
com.corel.pdx.appserver.AppSrvPort2=1101

## Lease time used by AppServer Objects(in minutes)
## default value = 10 minutes
com.corel.pdx.appserver.Lease=10

## Enable/disable Logging.
## Com.corel.pdx.appserver.LogCalls=[true|false]
com.corel.pdx.appserver.LogCalls=true

## Log file path.
## Note: all backslashes should be put as '\\'
com.corel.pdx.appserver.LogFile=.\\event.log
```

Using the Paradox JDBC Driver in Java applications

If you want to use the Paradox JDBC Driver as your connection between the Borland Database Engine and your Java application, you must do two things. First, you must include the path to the driver package (PdxJDBC.jar) in your Classpath. And second, you must include the following in your application source code:

- an import line which imports the JDBC classes into your application
- a Class.forName statement which loads the JDBC Driver Java class
- the JDBC URL string for the database to which you are connecting

Import line

- import com.corel.pdx.driver.*

Class.forName statement

- Class.forName("com.corel.pdx.driver.PdxJDBCDriver")

JDBC URL string

- jdbc:bdea://*MachineName*/*ParadoxDatabaseAlias*;SM=*SessionManagerName*;RemoteDbUser=*ParadoxRemoteUser*;PwdList=*ParadoxTableCommaSeparatedPasswordList*

- All parameters except Machine Name, Paradox Database Alias, and Session Manager are optional, depending on your circumstances, and can be specified in any order.
- If you chose a different RMI registry port than the default (1099), the syntax for MachineName would be *MachineName[:port]*.
- Everything in italics is user specific.

For an example of a class and how the above compenents are implemented, see Appendix A, Class Example of Paradox JDBC Driver, on page 37.

Paradox Report Server 5

Paradox 9 lets you take database information from tables and reports and publish them to the Internet or an intranet. In Paradox 8, you could create static and dynamic HTML documents for a Web page. Publishing a static HTML document creates a snapshot of a Paradox table or report for use on the Internet. Publishing a dynamic HTML document uses the Corel Web Server, and creates an HTML document where the data is dynamically updated each time the HTML document is loaded on the Web browser. However, you can only publish dynamically to a Windows platform.

You can still publishing static and dynamic HTML documents in Paradox 9. The new feature of Paradox 9 is the capability to publish a dynamic HTML document to any platform using Paradox Report Server. To publish dynamically using the Paradox Report Server, your Web server must support the Java Servlet API (such as JRun), and you must configure it to run the Paradox Report Server.

The Report Server uses the properties file *reportsrv.properties* to read the RMI registry port. The default registry port is 1099. If you are already using this port, or wish to change the default port, you will have to edit the properties file and your servlet runner administration. You can find reportsrv.properties in the default path:

- C:\Program Files\Corel\jPdox\dynpub\reportsrv.properties

- For more information on using a Java Servlet (such as JRun) and configuring it to support the Paradox Report Server, see Paradox Report Server and JRun, on page 17.

Dynamic HTML documents

A dynamic Web document is updated each time an HTML table or report created from a Paradox database is accessed. Dynamic HTML documents always contain the most up-to-date information available from the database.

Dynamic HTML documents are created in Paradox with the help of the HTML Table Expert or HTML Report Expert, which create table and report templates. The templates are published to a Web page. Each time the dynamic Web page is accessed, the current data is retrieved from the Paradox database and published to the page.

The Expert converts Paradox reports and tables to an HTML text file so that you can publish the information on the Internet or an intranet. The Expert automatically inserts appropriate HTML tags and parameters. You can modify the published table in the dynamic HTML document the same way that would in any other HTML document.

Paradox Report Server and JRun 6

Paradox 9 includes a Java application known as the Paradox Report Server. When this servlet is integrated with JRun, or with any servlet runner, it responds to requests for documents dynamically created in Paradox (using the HTML Table Expert or the HTML Report Expert and specifying the Paradox Report Server publishing option).

JRun is a Java based program that allows your Web server to be 100% servlet API-compatible and to run Java servlets. JRun extends your current Web server to include server-side Java functionality, or you can use JRun's built-in Web server.

- Only Paradox 9 Developer's Edition includes JRun. For more information about JRun, visit **http://www.livesoftware.com**.

JRun itself does not work directly with Paradox. It manages and uses the servlet to communicate with Paradox Report Server, which in turn communicates with the database. JRun should be set up to function in conjunction with your current Web server, or with JRun's own built in Web server. If you have not installed JRun and are using another servlet runner, you need to set it up to function with Paradox Report Server.

- You cannot use the Corel Web Server or the Corel Web Server .OCX control (an Active-X control shipped with Paradox) with JRun. You must use a Web server that supports the Java Servlet API and is configured to run the Paradox Report Server.

The Paradox Report server uses the properties file *reportsrv.properties* to read the RMI registry port. If you are already using this port, or wish to change the default port, you will have to edit the properties file. As well, when you are configuring your servlet runner (JRun), you will also have to add a parameter that tells the Report Servlet which port to use.

Setting up the Paradox Report Server

Depending on how you want to use the Paradox Report Server and your servlet runner (JRun or other), you must configure the servlet runner so that the two applications work together. Following are two examples of when you would have to customize the servlet runner administration and how you would do it. These examples discuss how to modify JRun administration—if you are using another servlet runner, you will have to modify it similarly.

Configuring your default port

If you are going to accept the default port (1099), you do not need to add this parameter to your Servlet Administration. This procedure explains how to configure the default port if you are using JRun as your Report Servlet.

To configure the Report Servlet

1. Click Start, Programs, JRun, Administration.
2. On the JRun Service Manager tab, select jsm.default and click Configure.
3. Click the Services tab.
4. In the Service ID column, select jse and click Service Config.
5. Click the Alias tab.
6. In the Name column, select ReportServlet and double-click the Init Arguments field.
7. In the Init Args Editor dialog box, click Add.
8. In the Name column, enter the following:

   com.corel.pdx.reportserver.RegistryPort

9 In the value column, enter the value set in the Report Server's property file (the port).

10 Click OK.

11 Click Save and click OK to close the Information dialog box.

Two-tier Web Solution

In a two-tier Web solution, you have two systems: a client machine with a Web browser installed, and a server machine with Paradox (and the Borland Database Engine), JRun, and a Web server installed.

First, you will need to locate the dynpub.jar file (usually in the default directory under \Corel\Paradox\dynpub\dynpub.jar). The file dynpub.jar is a Java archive file containing the class files which make up the Paradox Report Servlet. You must tell JRun where to find the file using the Java Classpath.

To edit the JRun Administration

1 On the Server machine, click Start, Programs, JRun, Administration.

2 On the JRun Service Manager tab, select jsm.default and click Configure.

To set the JRun Service Manager default configurations

1 Click the General Tab and then click the Java Tab.

2 In the Arguments box, add the following path to the end of the arguments list (the default directory may change depending on where the dynpub.jar file is installed):

   - ;c:\Progra~1\Corel\WebUti~1\run\dynpub.jar

   You must enter the eight character DOS equivalent for the directory names.

3 Click Save and click OK to close the Information dialog box.

To set the JRun Service Manager service configurations

1 Click the Services tab.

2 In the Service ID column, select jse and click Service Config.

3 Click the Aliases tab.

4 Click Add to add a new alias.

5 Double-click in the Name column and type in ReportServlet.

6 Double-click in the Class column and type the following class (case-sensitive):

- com.corel.pdx.reportservlet.ReportServlet

7 Click Save and click OK to close the Information dialog box.

8 Click Close to close the Service Config dialog box, click Close to close the jsm-default dialog box, and click Close again to close the JRun Administrator dialog box.

- Once JRun is configured to load the Paradox Report Servlet (for example, an alias and base path are set), you will need to modify some of the Windows registry settings on the machine where the Paradox documents are to be published (the server machine).

Testing the two-tier Web solution

On the server machine, ensure that your Web server and JRun (or your own servlet runner) are correctly set up and running. You can run a test that will tell you if the Web server, the Paradox Report Server, and the servlet runner are functioning. The first test provides an example from one of JRun's supplied examples (which are installed when you install JRun). If you do not have JRun, go to example two.

The second test shows you how to launch a dynamic report or table created in Paradox stored in your Paradox Web Repository (you need to first create the report or table in Paradox and publish it to HTML using the HTML Report Expert or HTML Table Expert).

Before you attempt the test, make sure the Web server, Paradox Report Server, and servlet runner are all running (for example, in Windows you need to launch these from the Start menu, or in Windows NT the servlet runner may be launched as a service).

- If you are using JRun as your servlet runner, you can run the first example. If you are using another servlet runner, you cannot run the first example, but you can still run the second example.

Example One: JRun example

1 On the client machine, launch your Web Browser.

2 In the Location box, type in the following URL (italics indicate user specific information):

   - http://*machine name*/servlet/SnoopServlet

3 Press Enter.

Example Two: Database

1 On the client machine, launch your Web Browser.

2 In the Location box, type the following path (italics indicate user specific information):

   - http://*machine name*/servlet/ReportServlet?*machine name*?*HTML page name*

   The HTML page name indicates the one that is saved in the Paradox Web Repository.

In each example, if you have the Paradox Report Server window open on the server machine, you will notice the requests processing.

Three-tier Web Solution

In a three-tier Web solution, you have three systems: a client computer with a Web browser installed, a Web server machine with a Web server and JRun installed, and a database machine with Paradox (and the Borland Database Engine) installed.

As in the two-tier Web solution, you must locate the dynpub.jar file. A copy of this file will need to be placed on the Web server machine. The file can be placed in any directory; however, you must tell JRun where to find the file. The only difference between the two-tier and three-tier Web solutions is the placement of the dynpub.jar file. In the two-tier Web solution the dynpub.jar file is only on the server machine. In a three-tier Web solution, the dynpub.jar file must be on both the Web server machine and the database machine.

To edit the JRun Administration

1 On the server machine, click Start, Programs, JRun, Administration.

2 On the JRun Service Manager tab, select jsm.default and click Configure.

To set the JRun Service Manager default configurations

1 Click the General Tab and then click the Java Tab.

2 In the Arguments box, add the path of where the dynpub.jar file is located on the Web server machine.

   You must enter the eight character DOS equivalent for the directory names.

3 Click Save and click OK to close the Information dialog box.

To set the JRun Service Manager service configurations

1 Click the Services tab.

2 In the Service ID column, select jse and click Service Config.

3 Click the Aliases tab.

4 Click Add to add a new alias.

5 Double-click in the Name column and type in ReportServlet.

6 Double-click in the Class column and type the following class (case-sensitive):

- com.corel.pdx.reportservlet.ReportServlet

7 **Click Save and click OK to close the Information dialog box.**

8 Click Close to close the Service Config dialog box, click Close to close the jsm-default dialog box, and click Close again to close the JRun Administrator dialog box.

Testing the three-tier Web solution

Before you attempt the test, make sure the Web server, Paradox Report Server, and servlet runner are all running on their respective machines (for example, in Windows you need to launch these from the Start menu, or in Windows NT they may be launched as a service).

For this test, you will specify a URL in the Web Browser on the client machine which corresponds to a dynamic HTML page you had previously published on the database machine (make sure you have a report or table available in your Paradox Web Repository).

Example

1 On the client machine, launch your Web Browser.

2 In the Location box, type the following URL (italics indicate user specific information):

- http://*Web server machine name* /servlet/ReportServlet?*database machine name?html page name*

The previous path is dependent on the settings that you defined in the JRun Administration. The actual format that you type into the Location box is as follows:

- http://*Web server machine name*/servlet base path/servlet alias?*database machine name*?*html page name*

- For more information on setting up JRun to function with a Web server, see the JRun documentation (at **http://www.livesoftware.com**).

JDBC Proxy Server 7

The JDBC Proxy Server is a Java application that redirects all incoming and outgoing information between the JDBC Client and the JDBC Server. It can be used as a security barrier, which is often an integral part of security enforcement in corporate firewalls within intranet setups. The JDBC Server can be configured to use specific ports. It could be exposed to the clients directly if the network server was configured to allow traffic on those fixed ports. For added security, the JDBC Proxy Server acts as a monitor between the network server and the JDBC Server, therefore not exposing the JDBC Server, and consequently the database information, to the outside world.

When a client opens a form or report that was dynamically published to HTML (using the HTML Report Expert or the HTML Table Expert in Paradox), the request for database information transparently passes through the JDBC Proxy Server on the Web server machine and is redirected to the JDBC Driver on the database machine (working in a three-tier configuration).

Configuring the JDBC Proxy Server

If you use the JDBC Proxy Server, you need to configure its properties file so that it knows the SessionMgr Name and the ports used by the Paradox JDBC server for which it is proxying.

You must set the following entries:

- SessionMgrName: the same as that set for the Paradox JDBC Server

  com.corel.pdx.appserver.SessionMgrName

- ServerHost: the hostname where the Paradox JDBC Server is running. If the RMI registry on the JDBC Server host is run on a port other than the default port (which is 1099), this entry should be set to hostname:RMIRegistryPort.

  com.corel.pdx.appserver.ServerHost

- AppServPort2: the same as that set for AppSrvPort2 for the Paradox JDBC server.

  com.corel.pdx.appserver.AppSrvPort2

The JDBC Proxy Server presents itself as the JDBC Server to the client. You can configure the JDBC Proxy Server to use a different SessionMgr object name and ports from what the JDBC Server is using (for example, ports and name of the SessionMgr Object to be used by the Proxy Server which will determine how clients contact the Proxy Server would need to be configured). You may change the following entries:

- SessionMgrName: the JDBC URL at the client would refer to this SessionMgr Object.

  com.corel.pdx.appserverprx.SessionMgrName

- Registry Port: used by the Proxy server

  com.corel.pdx.appserverprx.RegPort

- Ports used by the JDBC Proxy server (values may be different than those used by the JDBC server)

  com.corel.pdx.appserverprx.AppSrvPort1

  com.corel.pdx.appserverprx.AppSrvPort2

The following is an example of the appsrvprx.properties file used by the Proxy Server for configuration. (App Server = Paradox JDBC Server)

```
## ----------------------------------------
## Details about the App Server
## ----------------------------------------

## Name of the SessionMgr Object residing in
## the  AppServer.
Com.corel.pdx.appserver.SessionMgrName=SessionMgr

## HostName for the AppServer.
## Format= hostname[:RMIRegistryPort]
com.corel.pdx.appserver.ServerHost=vinayw_x2

## Ports Used
## port used by AppServer proxy objects
## default value = 1101
com.corel.pdx.appserver.AppSrvPort2=1101

## ----------------------------------------
## Proxy Server Configuration
## ----------------------------------------

## Enable/disable Logging for the AppServer Proxy.
## Com.corel.pdx.appserverprx.LogCalls=[true|false]
com.corel.pdx.appserverprx.LogCalls=true

## Name of the SessionMgr Object residing
## in the AppServer.
## This is the object that would be exposed
## to the client
com.corel.pdx.appserverprx.SessionMgrName=SessionMgr

## Registry port for the Local Registry
## default value = 1099
com.corel.pdx.appserverprx.RegPort=1099

## Ports Used
## port used by AppServer proxy objects
## default value = 1100
com.corel.pdx.appserverprx.AppSrvPort1=1100

## port used by AppServer proxy objects
## default value = 1101
com.corel.pdx.appserverprx.AppSrvPort2=1101
```

How the Utilities Work Together 8

There are a number of scenarios describing how the jPdox Web Utilities can work together to provide users with an Internet or intranet solution. The following examples provide a brief overview.

Case I: Enterprise

In the first situation, a developer has created a form using Paradox Web Form Designer on the developer machine, and has published the form to HTML. The developer wants the form to be available on the company intranet, where employees can view and edit current data from a Paradox database. This situation requires the following components:

System (Platform)	Installed Components
Developer Machine (any)	Paradox Web Form Designer
Client Machine (any)	Java enabled Web Browser
Web Server Machine (any)	Web Server (which supports JDK 1.2), JDBC Proxy Server
Database Server Machine (Windows)	Paradox 9 (with Borland Database Engine), and Paradox JDBC Server

The dynamic form created by the developer with Paradox Web Form Designer is saved as an applet and is stored on the Web server machine. When the client runs the applet, the components interact with the Paradox JDBC Server through the JDBC Proxy Server (on the Web server machine). The JDBC Proxy Server transparently coordinates incoming requests from clients and responses from the Paradox JDBC Server. The information from the database server machine passes through the JDBC Proxy Server and the form is displayed on the client machine. When the user enters new data, the JDBC Proxy Server directs the information back to the database server machine and updates the database.

Case 2: Small Business

In the second situation, a company wants to gather information from people who are visiting their external Web site. This situation requires the following components:

System (Platform)	Installed Components
Developer Machine (Windows)	Paradox Web Form Designer
Web Server machine (Windows)	Paradox 9 (with Borland Database Engine), Web Server (which supports JDK 1.2), and Paradox JDBC Server
Client Machine (any)	Java enabled Web Browser

The developer creates a simple data entry form and publishes it to HTML. The form is stored on the Web server machine, which is accessed by visitors who are viewing the company's external Web site. When the visitor enter their personal information into the Web form, the JDBC Server receives the information and stores it in the Paradox database.

Case 3: Customer Service

In the third situation, the customer service department has a database of all current products and pricing. They would like to provide this information to their clients via their company Web site. However, this information is constantly changing and provides a dynamic publishing solution. This situation requires the following components:

System (Platform)	Installed Components
Developer Machine (Windows)	Paradox 9 (with Borland Database Engine) and Report Server
Client Machine (any)	Java enabled Web Browser
Web Server machine (any)	Web Server (which supports JDK 1.2), Report Servlet, and JRun

The designer from the customer service department publishes a report dynamically using the HTML Report Expert in Paradox, which is saved in the Web repository on the developer machine. The client accesses the report from the company Web site on the Web server which dynamically retrieves the current information from the database.

Case 4: Advanced Java Developer

In the fourth situation, a developer is writing an inventory control application for the Java platform (using the Java Development Kit). The inventory control information is stored in a Paradox database. This situation requires the following components:

System (Platform)	Installed Components
Developer Machine (Windows)	Borland Database Engine, Java Development Kit, and JDBC Server
Client Machine (any)	Java Runtime Environment and the Inventory Control Application (with JDBC Client embedded)

When the Client runs the Inventory Control Java Application, the JDBC Client talks to the JDBC Server on the developer machine, which processes requests for data from the database (BDE).

Appendix A

Unsupported JDBC Class methods

PreparedStatement() methods

AddBatch()
getMetaData()
setArray()
setBlob(int i, Blob x)
setCharacterStream(int parameterIndex, Reader reader, int length)
setClob(int i, Clob x)
setDate(int parameterIndex, Date x, Calendar cal)
setNull(int paramIndex, int sqlType, String typeName)
setRef(int i, Ref x)
setTime(int parameterIndex, Time x, Calendar cal)
setTimeStamp(int parameterIndex, Timestamp x, Calendar cal)

Statement() methods

addBatch(String sql)
clearBatch()
executeBatch()
getFetchDirection()
getFetchSize()
getMaxFieldSize()
getMoreResults()
getQueryTimeout()
getResultSetConcurrency()
getResultSetType()
setCursorName(String name)
setEscapeProcessing(boolean enable)
setFetchDirection(int Direction)
setFetchSize(int rows)
setMaxFieldSize(int max)
setMaxRows(int max)
setQueryTimeout(int seconds)

ResultSet() methods

DeleteRow()
getArray(*)
getBlob(*)
getCharacterStream(*)
getClob(*)
getFetchDirection()
getFetchSize()
insertRow()
isLast()
moveToCurrentRow()
moveToInsertRow()
setFetchDirection()
setFetchSize()
updateXXX(*)
getConcurrency()
getCursorName()
getRef(*)
getStatement()
refreshRow()
cancelRowUpdates()
getBigDecimal(int columnIndex)
getBigDecimal(String columnName)
getDate(int columnIndex, Calendar cal)
getDate(String columnName, Calendar cal)
getObject(int i, Map map)
getObject(String colName, Map map)
getTime(int columnIndex, Calendar cal)
getTime(String columnName, Calendar cal)
getTimestamp(int columnIndex, Calendar cal)
getTimestamp(String columnName, Calendar cal)

ResultSetMetaData() methods

getColumnClassName(int column)
getColumnDisplaySize(int column)
getColumnTypeName(int column)
getSchemaName(int column)
getTableName(int column)
isAutoIncrement(int column)
isCaseSensitive(int column)
isDefinitelyWritable(int column)
isNullable(int column)
isReadOnly(int column)

isSearchable(int column)
isSignedint column)
isWritable(int column)

- The Borland Database Engine (BDE) offers only limited support for PreparedStatements, including the following:
    - Sub-queries in comparison expressions are not supported by PreparedStatement().
    - Sub-queries in ‘EXISTS’ expressions are not supported by PreparedStatement().
    - Sub-queries in ‘IN’ expressions are not supported by PreparedStatement().
    - Sub-queries in quantified expressions are not supported by PreparedStatement().
- Correlated subqueries are not supported by PreparedStatement().
- The Paradox JDBC Driver does not support CallableStatements.
- Concatenations between null and non-null values are not null.
- The Paradox JDBC Driver is not fully Sun JDBC compliant.

Appendix B

Example of Paradox JDBC Client Class

The following is an example of a class that uses a java.sql.Statement object to query a Paradox database

```
import java.sql.*;

//YOU MUST INCLUDE THE FOLLOWING LINE--IT INCLUDES THE
//DRIVER.

import com.corel.pdx.driver.*;

class Example
{
    public static void main (String args[])
   {

// THE FOLLOWING LINE IS THE URL
String url = "jdbc:bdea://georgec_95/Sample;SM=SessionMgr;
RemoteDbUser=tester;PwdList=pdx";

    //Sample Queries

    String query = "SELECT * FROM lineitem";

//NOTE: Column names with spaces that appear in a query String must be
//enclosed in backslashes. For example: \"My Column\"

 //String query = "SELECT t.\"Order No\", t.\"Stock No\" FROM lineitem t";

    try
    {
//YOU MUST INCLUDE THE FOLLOWING LINE--IT LOADS THE JDBC
 //DRIVER
    Class.forName ("com.corel.pdx.driver.PdxJDBCDriver");

//establish a connection to the server
Connection con = DriverManager.getConnection (url);

// Create a Statement object so we can submit SQL statements to the driver
Statement stmt = con.createStatement();
```

```
// Submit our query, creating a ResultSet object
ResultSet rs = stmt.executeQuery(query);

// DISPLAY ALL COLUMNS AND ROWS FROM THE RESULTSET
//To get info from the resultset, you must use a ResultSet class getxxx()
//method.
//To query other tables from Sample, you must setup a
//getxxx() method for each column of the other table.
//The following getxxx() methods will only display
 //resultsets from the lineitem table
while (rs.next())
{
//load resultset data into java primitive types
double numb = rs.getDouble("Order No");
double stock = rs.getDouble("StockNo");
double price = rs.getDouble("Selling Price");
double qty = rs.getDouble("Qty");
double total = rs.getDouble("Total");

//Print the results to the screen

System.out.println(numb + " " + stock + " " + price + " " + qty + " " +
total);
          }
// Close the result set

rs.close();
// Close the statement
          stmt.close();

          // Close the connection
          con.close();
       }
   //CATCH ERRORS
       catch (SQLException ex)
       {

// An SQLException was generated.  Catch it and
// display the error information.  Note that there
// could be multiple error objects chained together
```

```
System.out.println ("\n*** SQLException caught ***\n");

         while (ex != null)
         {
            System.out.println ("SQLState: " + ex.getSQLState ());
            System.out.println ("Message:  " + ex.getMessage ());
            System.out.println ("Vendor:   " + ex.getErrorCode ());
            System.out.println ("");
            ex.printStackTrace ();
            ex = ex.getNextException ();
            System.out.println ("");
         }
      }
      catch (java.lang.Exception ex)
      {
         // Got some other type of exception.  Dump it.
         System.out.println("Lang Exception");
         ex.printStackTrace ();
      }
   }
}
```

Index