

Integrating the HP NonStop JDBC Type 4 Driver with JBoss applications



Introduction	2
Products and components	2
Setup	2
JBoss Application Server	2
The J2EE Tutorial—Duke Bank’s Application	3
What’s different	3
Extract the Type 4 files	3
Extract Duke’s Bank files	3
Setting up the JBoss environment	3
Preparing the files	3
Compiling the Java source	4
Packaging the EJB	4
Packaging the Web Archive (WAR) file	4
Packaging the Java client	5
Assembling the EAR file	5
Setting up the database	5
Populating the NonStop SQL/MX Database	5
Starting the NonStop ODBC/MX Connectivity Service	6
Configuring JDBC Type 4 data source for the JBoss Application Server	6
Copying the t4sqlmx.jar file	7
Starting the JBoss Application Server	7
Deploying and running the sample	7
Conclusion	8
Ordering Information	8
For more information	9

Introduction

The HP NonStop JDBC Type 4 Driver (hereafter, the Type 4 Driver) is a high-performing standard compliant driver providing remote access from Java™ clients to NonStop SQL/MX Release 2.0 Database MP and MX tables. Type 4 Driver is integrated with the features of the NonStop SQL/MX Database server and provides a highly available and scalable connectivity solution to the HP NonStop platform.

The purpose of this document is to provide a step-by-step guide to integrating the Type 4 Driver in the Java 2 Platform, Enterprise Edition (J2EE) compliant JBoss Application Server. This guide uses the popular “Duke’s Bank” provided by Sun Microsystems in the J2EE tutorial. Further details about the application can be found in “The J2EE Tutorial—Duke’s Bank Application” section of this paper.

Products and components

This section describes the various products and components needed in order to successfully run the Duke’s Bank sample (see table).

Products and components needed for Duke’s Bank sample.		
Product/component	Where to obtain	Special notes
NonStop JDBC Type 4 Driver	Ordering information	
Auxiliary files for sample	Same location as this document (named NonStopFiles_JBoss.zip)	Download here
JBoss Application Server	http://www.jboss.org	Download Version 4.0 RC1 or later
Duke’s Bank Application	http://java.sun.com/j2ee/1.3/download.html#tutorial	Download the 1.3 tutorial files (and not the 1.4 ones)*
Jakarta Struts	http://jakarta.apache.org/struts	
JBoss specific files	http://www.jboss.org/docs/jbossj2ee-src.zip	
Apache Ant	http://ant.apache.org	Version 1.5.4 or later

Note: *The 1.4 version of the files have a different directory layout.

Setup

This section provides documentation to configure the JBoss Application Server environment to run the Duke’s Bank Application. Please follow the order for the installation process as described below.

JBoss Application Server

Extract the files from the JBoss Application Server archive into a directory on your system. Hereafter, this directory will be referred to as **JBOSS_HOME**.

The J2EE Tutorial—Duke Bank’s Application

As mentioned earlier, we will make use of the example applications provided by Sun in the J2EE tutorial, and, in particular, the “Duke’s Bank” Application. You can find the tutorial online at http://java.sun.com/j2ee/tutorial/1_3-fcs/index.html.

Duke’s Bank is an online banking application. It has two clients: an application client, used by administrators to manage customers and accounts, and a Web client, used by customers to access account histories and perform transactions. The clients access the customer, account, and transaction information maintained in a database through Enterprise JavaBeans (EJB). The Duke’s Bank Application demonstrates the way that many of the component technologies—EJBs, application clients, and Web components—are applied to provide a simple but functional application.

What’s different

J2EE technologies are designed so that the client code is independent of the server in which the application is deployed. The deployment descriptors for EJB and Web applications (**ejb-jar.xml** and **web.xml**, respectively) are also standard and do not change between different J2EE containers. However, there are a few things that need to be done in order to move the application to the JBoss Application Server. In particular, we have to supply JBoss specific descriptors and ensure that the database scripts will work. The **NonStopFiles_JBoss.zip** contains the necessary files to make this configuration change. The files contained in the archive are

- **dukesDB.sh**—contains the script file to create and populate the necessary database on NonStop SQL/MX Software.
- **sqlmx-ds.xml**—contains the configuration to create the default data source needed by the Duke’s Bank sample to communicate with the NonStop SQL/MX Database.

Extract the Type 4 files

Install the NonStop JDBC Type 4 Driver using the installation instructions provided in the product’s README.txt file. The installation directory will hereafter be referred to as **TYPE4_HOME**.

Extract Duke’s Bank files

Unpack the downloaded archive file containing the Duke’s Bank files. This will create a directory named **j2eetutorial** containing all the necessary source and configuration files.

Setting up the JBoss environment

Preparing the files

1. The zip archive named **jbossj2ee-src.zip** contains the supplementary JBoss files. Unpack the contents into the **j2eetutorial** directory (mentioned in the “Extract the Type 4 files” section), adding to the existing tutorial files.
2. The entire sample code is in the **bank** subdirectory. If you’ve unpacked the files correctly, you should find a **jboss-build.xml** file in this directory. This is the Ant build script for the JBoss Application Server version of the application.
3. Edit the **bank/jboss-build.xml** to make the following changes:

Comment out the following XML text:

```
<!--
```

```

        <fileset dir="${jboss.home}/server/all/deploy/jboss-
            net.sar">
            <include name="*.jar"/>
        </fileset>
    -->

```

Add the following XML text immediately after the above text:

```

    <fileset dir="${jboss.home}/server/all/lib">
        <include name="*.jar"/>
    </fileset>

```

4. Download the struts distribution and copy the **struts.jar**, **struts-logic.tld**, and the supporting jakarta-commons jars (all those prefixed with "commons-") to **bank/jar**.
5. In the **j2eetutorial** directory you will find the "**build.properties**" file. Edit this to set the `jboss.home` property to the full path to your JBoss installation. The build process makes use of the jar files and utilities that come with the JBoss Application Server, so it needs to know where to find them. For example, if you have unpacked JBoss Application Server to the C: drive on a Microsoft® Windows® system-based machine, you would set it to the following:

```

# Set the path to the JBoss directory containing
# the JBoss application server
# (This is the one containing directories like "bin",
# "client" etc.)

jboss.home=C:/jboss-4.0.0RC1

```

Compiling the Java source

At the command line, change to the **j2eetutorial/bank** directory. All the build commands will be run from here. Compilation should be fairly straightforward—just type the following command:

```
ant -f jboss-build.xml compile
```

This runs the "compile" target in the build script. If there aren't any errors, you should find a newly created build directory with the class files in it.

Packaging the EJB

The application has three separate EJB jars: **account-ejb.jar**, **customer-ejb.jar**, and **tx-ejb.jar**. Each jar file contains the code and description (**ejb-jar.xml** and **jboss.xml**) for the corresponding entity bean and an associated "controller" session bean, which the clients interact with (it is considered "bad" programming to have clients communicate directly to entity beans). Execute the following command:

```
ant -f jboss-build.xml package-ejb
```

This should create the controller session bean (in the **jar** directory).

Packaging the Web Archive (WAR) file

Next is the Web application, which provides the front end to allow users to interact with the business components (the EJB). The Web source (JavaServer Pages, or JSP; images; and so on) is contained in the **src/web** directory and is added unchanged to the archive.

The Ant WAR task also adds a **WEB-INF** directory, which contains the files that are not meant to be directly accessed by a Web browser but are still part of the Web application. These include the deployment descriptors (**web.xml** and **jboss-web.xml**), class files (for example, servlets and EJB

interfaces), and extra jars and JSP tag library descriptors required by the Web application (the struts files in this example). The command to build the Web client WAR file is as follows:

```
ant -f jboss-build.xml package-web
```

Packaging the Java client

In addition to the Web interface, there is a standalone Java client for administering customers and accounts. You can build it using the following command:

```
ant -f jboss-build.xml package-client
```

It contains the **application-client.xml** and **jboss-client.xml** descriptors as well as the client **indi.properties** file. The client jar will also be included as an additional module in the Enterprise Archive (EAR) file and the server.

Assembling the EAR file

The EAR file is the complete application, containing the three EJB modules and the Web module. It must also contain an additional descriptor called **application.xml**. It is also possible to deploy EJB and Web application modules individually, but the EAR file provides a convenient single unit:

```
ant -f jboss-build.xml assemble-app
```

This should produce the final file **JBossDukesBank.ear**.

Setting up the database

Before you can deploy the application, you need a viable database for it to run against. If you are writing an application that uses container-managed EJB persistence, you can configure the engine to create the tables for you at deployment; otherwise, you must have a set of scripts to do the task. This is also a convenient way of pre-populating the database with data.

The J2EE Software Development Kit (SDK) comes with the Cloudscape database, which is used throughout the tutorials. You will be using the NonStop SQL/MX Database that is running on the NonStop system.

You must complete a few tasks before you can successfully test your application:

- Create and populate the NonStop SQL/MX Database with the appropriate tables and values
- Start the HP NonStop ODBC/MX Service on the NonStop system
- Choose the JBoss Application Server to use the JDBC Type 4 Driver to connect to the NonStop SQL/MX Database
- Make the JDBC Type 4 Driver jar file available in the JBoss environment

Populating the NonStop SQL/MX Database

Supplementing this document is a zip file named **NonStopFiles_JBoss.zip**. One of the files in this archive, named **dukesDB.sh**, contains the script file to create and populate the necessary database on NonStop SQL/MX Software. Unpack the archive file and transfer **dukesDB.sh** (in ASCII mode) to the HP Open System Services (OSS) environment.

To set up the NonStop SQL/MX Database, change to the directory on OSS where the **dukesDB.sh** is located and run the following command:

```
OSS> ./dukesDB.sh
```

This creates the database in the schema **DUKECAT.DUKESHEMA**.

Starting the NonStop ODBC/MX Connectivity Service

The NonStop ODBC/MX Connectivity Service, also known as the MXCS Association Server, must be installed and running on the NonStop system. Check the manual documentation (T7969G10) for details on how to start the MXCS Association Server.

The following is a quick reference to start the MXCS Association Server from the TACL command prompt as a super.super user:

```
<volume>.<subvolume>.mxoas /name $<process name>,  
                               nowait, term $zhome/ -pn <port>
```

Example:

```
$data08.A2040209.mxoas /name $vn, nowait, term $zhome/ -pn 6100
```

Note: The above TACL command must be issued from an audited disk volume.

Configuring JDBC Type 4 data source for the JBoss Application Server

The **NonStopFiles_JBoss.zip** also contains a file named **sqlmx-ds.xml**. This file contains the configuration to create the default data source needed by the Duke's Bank sample to communicate with the NonStop SQL/MX Database.

The contents of the **sqlmx-ds.xml** are as follows:

```
<?xml version="1.0" encoding="UTF-8"?>  
<datasources>  
  <local-tx-datasource>  
    <jndi-name>DefaultDS</jndi-name>  
    <connection-url>jdbc:t4sqlmx://16.1.1.1:6100/:</connection-url>  
    <driver-class>com.tandem.t4jdbc.SQLMXDriver</driver-class>  
    <user-name>username</user-name>  
    <password>password</password>  
    <connection-property name="schema">DUKESHEMA  
    </connection-property>  
    <connection-property name="catalog">DUKECAT  
    </connection-property>  
    <connection-property name="T4LogLevel">ALL  
    </connection-property>  
    <connection-property name="T4LogFile">C:/T4.log  
    </connection-property>  
    <!--  
      Additional properties for the NonStop JDBC Type 4 driver  
      can be set here by adding additional tags of the type:  
      <connection-property  
name="PROP_NAME">PROP_VALUE</connection  
-->  
  </local-tx-datasource>  
</datasources>
```

Edit the value of the **<connection-url>** element for the location where the MXCS Association Server is running.

Also, edit the values of the **<user-name>** and **<password>** elements to reflect a valid user on the NonStop system that has sufficient privileges to access the Duke's Bank sample database.

You can add additional Type 4 Driver specific properties by using the **<connection-property>** tag. Refer to the Type 4 Driver manual for a list of valid properties and their effects.

Place the `sqlmx-ds.xml` file in the `JBOSS_HOME/server/default/deploy` directory.

Also, from the same directory (`JBOSS_HOME/server/default/deploy`), remove the existing `hsqldb-ds.xml` file. This will prevent the duplicate definition of the Java Naming and Directory Interface (JNDI) named "DefaultDS" during the server startup process.

Copying the `t4sqlmx.jar` file

The `sqlmx-ds.xml` file from the "Configuring JDBC Type 4 data source files for the JBoss Application Server" section defines the `SQLMXDriver` as the driver class to be used for the `DefaultDS` datasource. This driver class is part of the `t4sqlmx.jar` file and must be made available to the JBoss environment at runtime.

To do this, copy the `TYPE4_HOME/lib/t4sqlmx.jar` file to the `JBOSS_HOME/server/default/lib` directory.

Starting the JBoss Application Server

Start the default JBoss Application Server by executing the startup script—`run.bat`—located at `JBOSS_HOME/bin`.

Note: There may be some JMS related errors that you can safely ignore because the Duke's Bank sample does not use any JMS code.

Deploying and running the sample

Deployment using the JBoss Application Server is easy: You just have to copy the EAR file to the deploy directory. There's also a target in the build file for this purpose, so you can type

```
ant -f jboss-build.xml deploy
```

This will assemble the EAR file and deploy it. You should see something close to the following output from the server (shortened here somewhat for brevity):

```
19:30:32,966 INFO [MainDeployer] Starting deployment of package:
file:/F:/servers/jboss-3.2.3/server/default/deploy/JBossDukesBank.ear

19:30:32,997 INFO [EARDeployer] Init J2EE application: file:/F:/servers/jboss-
3.2.3/server/default/deploy/JBossDukesBank.ear

19:30:34,513 INFO [EjbModule] Deploying AccountEJB
...
19:30:45,356 INFO [Engine] StandardManager[/bank]: Seeding random number
generator class java.security.SecureRandom

19:30:45,356 INFO [Engine] StandardManager[/bank]: Seeding of random number
generator has been completed

19:30:45,356 INFO [Engine] StandardWrapper[/bank:default]: Loading container
servlet default

19:30:45,356 INFO [Engine] StandardWrapper[/bank:invoker]: Loading container
servlet invoker

19:30:45,685 INFO [EARDeployer] Started J2EE application: file:/F:/servers/jboss-
3.2.3/server/default/deploy/JBossDukesBank.ear

19:30:45,685 INFO [MainDeployer] Deployed package: file:/F:/servers/jboss-
3.2.3/server/default/deploy/JBossDukesBank.ear
```

If there are any errors or exceptions, make a note of the error message and the point at which it occurred (for example, during the deployment of a particular EJB, the Web application, and so on).

Check that the EAR file is complete and inspect the WAR file and each of the EJB jar files produced by the build to ensure they contain all the necessary components (classes, descriptors, and so on).

Note: There may be some JMS related errors that you can safely ignore because the Duke's Bank sample does not use any JMS code.

You can safely redeploy the application if it is already deployed. To undeploy it, you just have to remove the archive from the deploy directory. There's no need to restart the server in either case. If everything seems to have gone well up to this point, then point your browser at the following application URL:

<http://localhost:8080/bank/main>

You should then be forwarded to the application login page. As explained in the tutorial, you can log in with a customer ID of "200" and the password "j2ee". If you receive an error at this point, check again to ensure that you have set up the database correctly, as described in the "Configuring JDBC Type 4 data source for the JBoss Application Server" section. In particular, check that the **connection-url** is correct. Then ensure that you have populated the database with data.

You should also be able to run the standalone client application using the following command:

```
ant -f jboss-build.xml run-client
```

This is a Swing graphical user interface (GUI) client, which allows you to administer the customers and accounts.

Information about further customization to the Duke's Application can be found online from the JBoss site at the following URL:

<http://www.jboss.org/modules/html/docs/jbossj2ee.pdf>

Conclusion

The Type 4 Driver supports the portability goal of fast ROI with absolute minimum application change. This sample indicates how applications written for a third-party JDBC driver can be executed on the Type 4 Driver with just runtime configuration changes. It highlights that the Type 4 Driver's JDBC standards compliance helps you port existing applications or write new ones to use the NonStop SQL/MX or SQL/MP Database via any platform.¹

The example here is only intended to get you up and running with integrating NonStop SQL/MX Database and the Type 4 Driver with the JBoss Application Server. The applications are not intended to reflect how you should go about writing production J2EE software, as there are differences of opinion on this subject.

Ordering Information

The NonStop JDBC Type 4 Driver is an Independent Product that can be ordered using the part number SJ82V1.

¹ The platform used must have Java 1.4.1 or later running on it.

For more information

www.hp.com/go/nonstop

© 2005 Hewlett-Packard Development Company, L.P. The information contained herein is subject to change without notice. The only warranties for HP products and services are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. HP shall not be liable for technical or editorial errors or omissions contained herein.

Java is a US trademark of Sun Microsystems, Inc. Microsoft and Windows are U.S. registered trademarks of Microsoft Corporation.

04/2005

