Oracle WebLogic Server 11*g*: Advanced Administration

Activity Guide

D58686GC20

Edition 2.0

December 2010

D71660

ORACLE

Author

TJ Palazzolo

Technical Contributors and Reviewers

Prashant Agarwal, Tom Barnes, Steve Button, Dave Cabelus, Josh Dorr, Arvind Jain, Anthony Lai, Serge Moiseev, Craig Perez, Mark Prichard, Sandeep Shrivastava, John Van Pelt

This book was published using: Oracle Tutor

Table of Contents

Practices for Lesson 1

Chapter 1

Overview of Practices for Lesson 1

Practices Overview

There are no practices for this lesson.

Practices for Lesson 2

Chapter 2

Overview of Practices for Lesson 2

Practices Overview

There are no practices for this lesson.

Practices for Lesson 3

Chapter 3

Overview of Practices for Lesson 3

Practices Overview

In the practices for this lesson, you explore the capabilities of the Domain Template Builder and Configuration Wizard tools. These practices also give you the opportunity to work with some custom WLST scripts.

Naming Conventions

The exercises in this course use the following variable names to refer to commonly used locations on your file system:

Variable	Path
<MIDDLEWARE_HOME>	/u01/app/oracle/Middleware/11.1.1
<WEBLOGIC_HOME>	**<MIDDLEWARE_HOME>**/wlserver_10.3
<JAVA_HOME>	**<MIDDLEWARE_HOME>**/jdk160_18
<STUDENT>	/home/oracle/wls11g_advadm
<LAB>	**<STUDENT>**/labs/lab*XX_YY*, where *XX_YY* is the current practice number such as 03_01
<WORK>	**<STUDENT>**/work

Practice 3-1: Create a Custom Domain Template

Duration: 50 minutes

Objectives

After completing this practice, you should be able to:

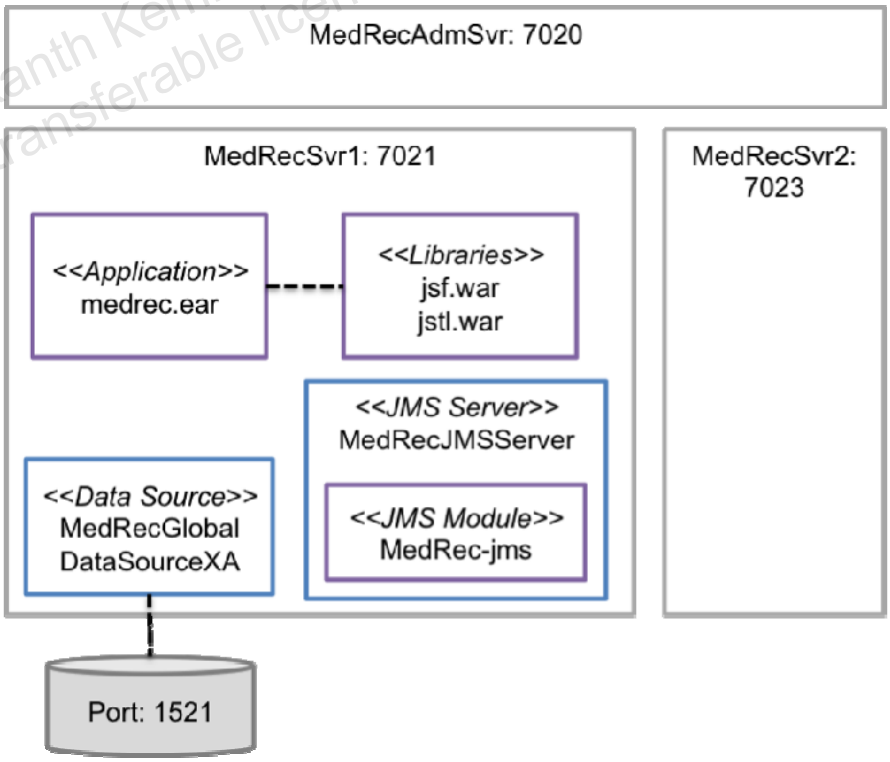
- Use the Domain Template Builder
- Add custom files and SQL scripts to a template
- Configure variable replacement in template files
- Use the patient features of the MedRec application

Overview

The Oracle Medical Records (MedRec) Java EE application provides patients with the ability to view current and previous medical examinations and prescriptions over the Web. New patients can request an account, which is later approved or denied by the MedRec administrator. The application also includes a collection of Web services to support integration with other enterprise systems.

MedRec, like all Java EE applications, is dependent on various server resources. These resources include Java Database Connectivity (JDBC) data sources, Java Message Service (JMS) queues and topics, and shared Java EE libraries. In this practice, you will create the necessary domain infrastructure to support the MedRec application and package it within a domain template by using the Domain Template Builder tool. The schema required by MedRec will also be bundled within the domain template as SQL scripts, so that administrators can quickly initialize any relational database used to support the domain.

The MedRec domain infrastructure is depicted here:



Dependencies

No prior practices need to be completed before starting this practice. However, note that this practice must be completed successfully (or the solution steps performed), because the MedRec infrastructure is a prerequisite for many subsequent exercises.

Tasks

1. Create a basic WebLogic domain.
   a. Create a folder `<WORK>/domains` if one does not already exist.

      Note: The `<STUDENT>/work` folder may also need to be created.
   b. Launch a new Linux Terminal by using the shortcuts on your desktop.
   c. Start your database by using the script `<STUDENT>/bin/startDB1.sh`.
   d. Navigate to `<WEBLOGIC_HOME>/common/bin`.

      Tip: For convenience, `WEBLOGIC_HOME` is also an OS environment variable.
   e. Execute the following to launch the Configuration Wizard:

```
./config.sh
```

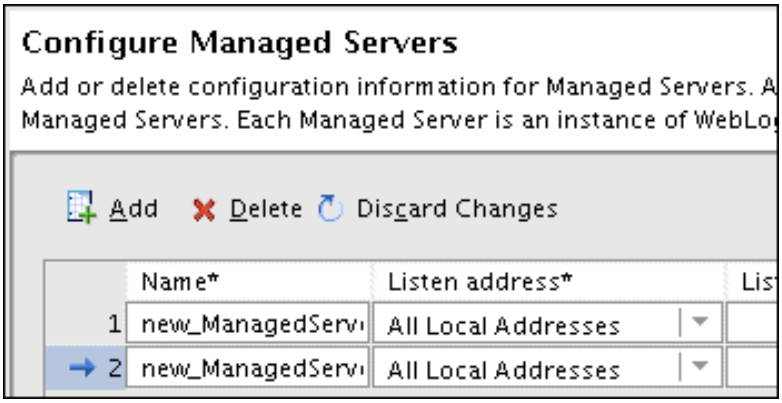
   f. Verify that the **Create a new WebLogic Domain** option is selected and click **Next**.
   g. Select the **Base this domain on an existing template** option.
   h. Click **Browse** and select the template file `<WEBLOGIC_HOME>/common/templates/domains/wls.jar`. Click **OK**. Then click **Next**.
   i. Enter the following values:

Field	Value
Domain Name	MedRecDomain
Domain Location	Browse to `<WORK>/domains`

      Click **Next**.
   j. Enter the `Welcome1` as the password and click **Next**. Be sure to enter this exactly as shown.
   k. Select the **Production Mode** option and click **Next**.
   l. Select the following check boxes and click **Next**:
      - **Administration Server**
      - **Managed Servers, Clusters and Machines**
   m. Edit the following fields:

Field	Value
Name	MedRecAdmSvr
Listen Port	7020

      Click **Next**.
   n. Click the **Add** button two times to view the following screen:



o. Enter the following values:

Name	Listen Address	Listen Port
MedRecSvr1	All Local Addresses	7021
MedRecSvr2	All Local Addresses	7023

Click **Next**.

p. Click **Next** to skip the *Configure Clusters* step.

q. Click **Next** to skip the *Configure Machines* step.

r. Click **Create**. When finished, click **Done**.

2. Start the MedRec servers.

a. Launch two Linux terminals. Within each shell, navigate to `<WORK>/domains/MedRecDomain`.

Tip: Alternatively, select **File > Open Tab** to keep all shell prompts within the same window.

b. In the first shell, start the domain's Administration Server:

```
./startWebLogic.sh
```

c. When prompted, enter the credentials `weblogic/Welcome1`.

Tip: Recall that, to avoid supplying credentials each time a server is started, you can optionally create a `boot.properties` file for each of your servers.

d. Confirm that the server started successfully:

```
<Notice> <WebLogicServer> <BEA-000360> <Server started in
RUNNING mode>
```

e. In the second shell, navigate to the `bin` subfolder.

f. Start `MedRecSvr1`:

```
./startManagedWebLogic.sh MedRecSvr1 localhost:7020
```

g. Enter the same administration credentials.

3. Use WLST to initialize domain resources.

a. Launch a new Lab Framework command shell by executing the `<STUDENT>/bin/prompt.sh` file.

Tip: This shell includes the necessary environment variables to run WebLogic command line tools such as WLST.

b. Navigate to `<LAB>/resources/jdbc`.

c. Execute the `createDataSource.py` WLST script:

```
java weblogic.WLST createDataSource.py
```

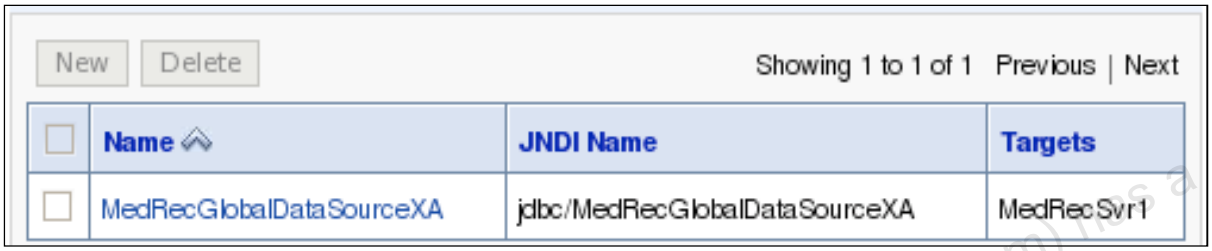
d. Confirm that the script executed successfully:

```
Data Source created successfully.
```

e. Launch a Web browser and log in to the administration console: `http://localhost:7020/console`.

f. In the **Domain Structure** panel, navigate to **Services > JDBC > Data Sources**.

g. Verify that the `MedRecGlobalDataSourceXA` data source was added to the domain:

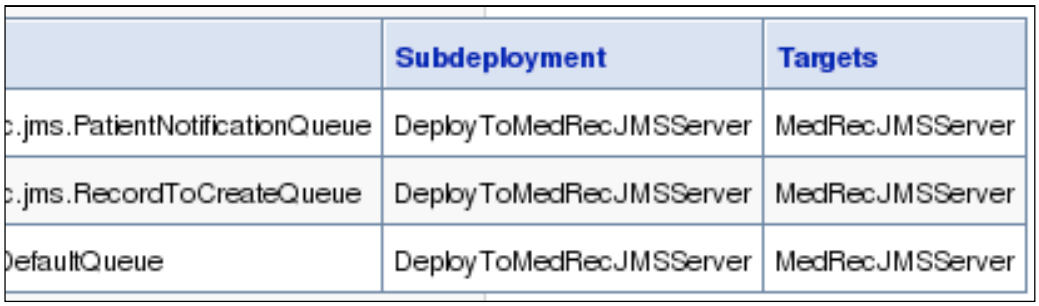


h. Return to your Lab Framework prompt. Execute the following additional WLST scripts:

- `<LAB>/resources/jms/createJMSServerAndModule.py`
- `<LAB>/resources/apps/deployLibraries.py`

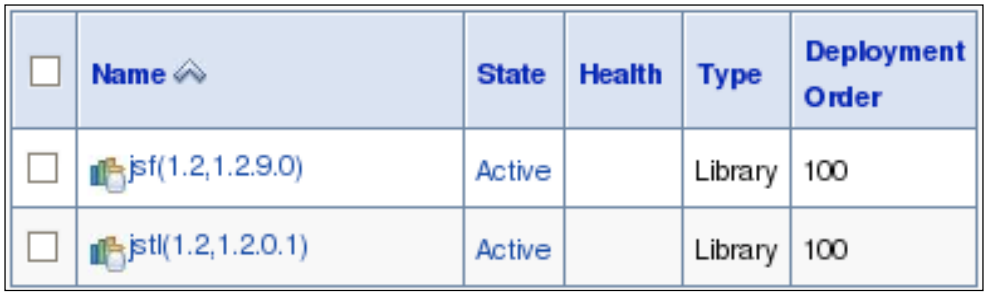
i. Within the console, navigate to **Services > Messaging > JMS Modules**. Click the **MedRec-jms** module:

j. Scroll down to the **Summary of Resources** table. Confirm that all resources in this module are targeted to `MedRecJMSServer`:



k. Within the **Domain Structure** panel, click **Deployments**.

l. Confirm that two shared libraries are deployed:



Name	State	Health	Type	Deployment Order
jsf(1.2,1.2.9.0)	Active		Library	100
jstl(1.2,1.2.0.1)	Active		Library	100

4. Start a custom template using your existing domain.
   a. Shut down your Administration Server as well as `MedRecSvr1`. Unless otherwise indicated, **Ctrl + c** should be sufficient to shut down your servers in these exercises.
   b. Create the following folder if it does not already exist: `<WORK>/templates`.
   c. From a terminal, navigate to the location `<WEBLOGIC_HOME>/common/bin`. Execute the following:

```
./config_builder.sh
```

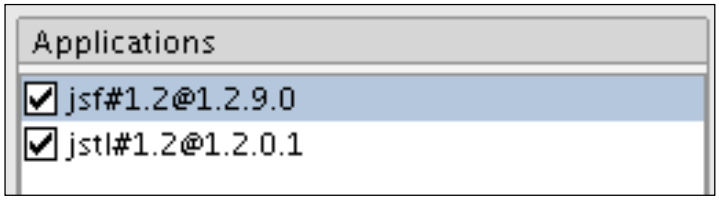
   d. Verify that the **Create a Domain Template** option is selected and click **Next**.
   e. Browse to and select your domain folder, `<WORK>/domains/MedRecDomain`. Click **Next**.
   f. Enter the following values:

Field	Value
Name	MedRecDomain
Version	10.3.3.0
Author	Healthcare IT Services
Category	MedRec
Description	Production domain to support MedRec

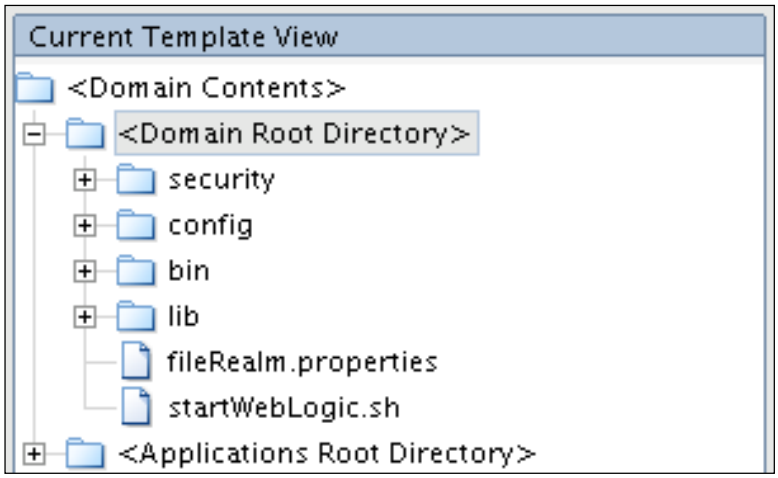
   Click **Next**.
   g. Enter the following values:

Field	Value
Template JAR Name	MedRecDomain
Template Location	Browse to `<WORK>/templates`

   Click **Next**.
   h. Confirm that your two shared libraries are selected and click **Next**:



   i. In the right panel labeled **Current Template View**, expand and select **<Domain Root Directory>**:



j. In the left panel labeled **File System View**, locate and select the file `<LAB>/resources/files/startServer1.sh`.

k. Click the **Add** button.

l. Repeat the previous steps to add the following additional files to the template:
   - `<LAB>/resources/files/startServer2.sh`
   - `<LAB>/resources/files/wlstPrompt.sh`

m. Click **Next**.

n. Select the following options:

Field	Value
Type	Oracle
Version	Any

o. Click the **Add SQL File** button.

p. Browse to and select the following files at `<LAB>/resources/sql/`:
   - `medrec_data.sql`
   - `medrec_dropall.sql`
   - `medrec_tables.sql`

q. Click the **Add SQL Files** button.

r. Use the up and down buttons to reorder the SQL scripts as shown:
   - medrec_dropall.sql
   - medrec_tables.sql
   - medrec_data.sql

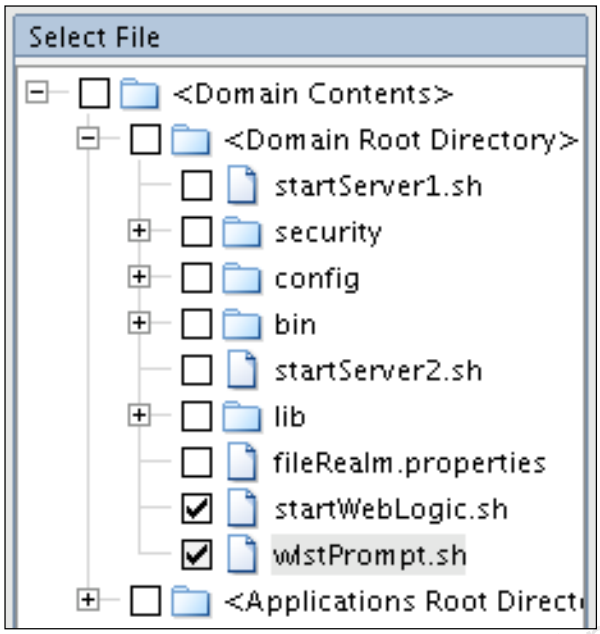
s. Click **Next**.

5. Configure variable replacement for custom template files.
   a. Click **Next** to skip the *Configure the Administration Server* step.
   b. Click **Next** to skip the *Configure Administrator User Name and Password* step.
   c. Click **Next** to skip the *Specify Start Menu Entries* step.
   d. Launch a text editor and inspect the contents of the file: `<LAB>/resources/files/wlstPrompt.sh`.
   e. Notice that the path to your WebLogic installation is hardcoded. Here is an example:

```
/u01/app/oracle/Middleware/11.1.1/wlserver_10.3
```

f. Close the file and return to the Domain Template Builder.

g. If not already selected, select the check box for the file `wlstPrompt.sh`:



h. Select `wlstPrompt.sh` and click the **Edit** button.

i. Notice that the preceding hardcoded text has automatically been replaced with a token (`@token_name`). A new file is generated that contains these updates: `<LAB>/resources/files/_edit_wlstPrompt.sh`.

j. Click **Next**.

k. Review the template's contents and click **Create**. When finished, click **Done**.

6. Test the custom template using the Configuration Wizard.

a. Move the domain directory `MedRecDomain` to some backup location of your choosing. For example:

```
mv <WORK>/domains/MedRecDomain <WORK>/backup/MedRecDomain
```

b. Create the following folder if it does not already exist: `<WORK>/libs`.

c. Launch the Configuration Wizard (`config.sh`) again.

d. Click **Next**.

e. Select the **Base this domain on an existing template** option. Browse to and select your new template: `<WORK>/templates/MedRecDomain.jar`. Click **Next**.

f. Enter the following values:

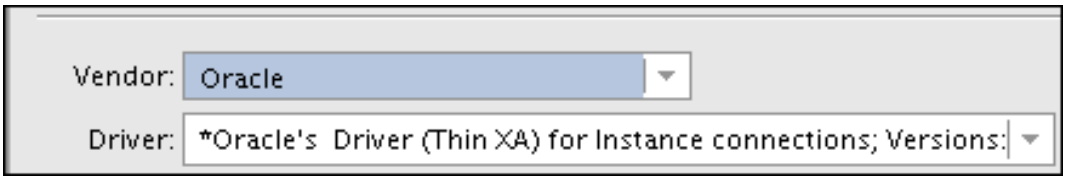
Field	Value
Domain Name	MedRecDomain
Domain Location	**Browse** to `<WORK>/domains`
Application Location	**Browse** to `<WORK>/libs`

Click **Next**.

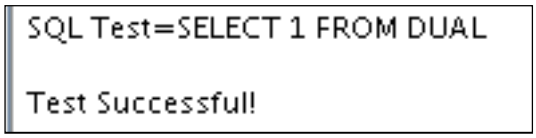
g. Click **Next** to skip the *Configure Administrator User Name and Password* step.

h. Select the **Production Mode** option and click **Next**.

i. Confirm that the domain contains a single data source. Select the check box and verify that its **Vendor** field is set to **Oracle**. Click **Next**:

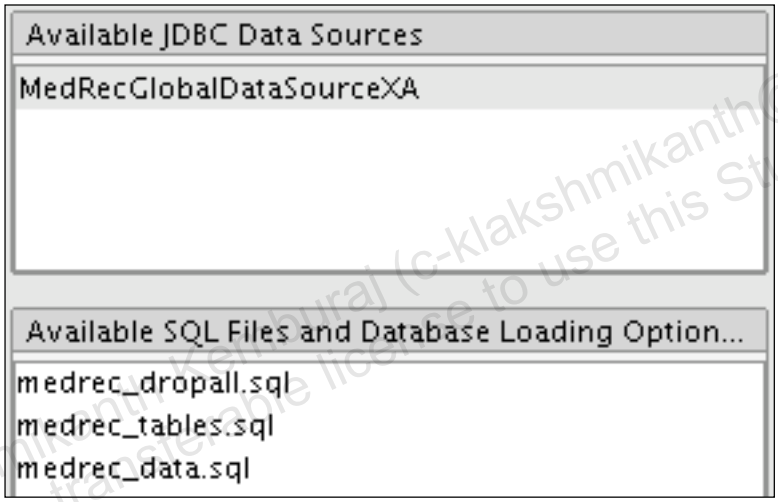


j. The data source will automatically be tested. Confirm that the test was successful:



Click **Next**.

k. Verify that the same data source is selected and that three SQL files were found for the same database vendor:



l. Click **Run Scripts**.

m. Confirm that the "Database Load Successful" message is displayed. Click **Next**.

n. Click **Next** to skip the *Select Optional Configuration* steps.

o. Click **Create**. Click **Done**.

7. Deploy the MedRec application.
   a. Locate the `<LAB>/solution/applications/medrec.ear` file. Copy it to the `<WORK>/applications` location. Create this folder if it does not already exist.
   b. Start your Administration Server by using the `<WORK>/domains/MedRecDomain/startWebLogic.sh` script.
   c. Start `MedRecSvr1` using the script `<WORK>/domains/MedRecDomain/startServer1.sh`.
   d. After the servers have started, return to your Lab Framework prompt.
   e. Execute the following WLST script: `<LAB>/resources/apps/deployApp.py`.
   f. Confirm that the script executed successfully:

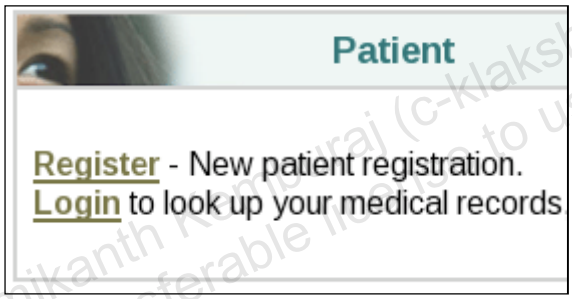
```
Application medrec deployed.
```

   Tip: To avoid Work Manager warnings, you can also optionally run the `<LAB>/resources/apps/createWorkManager.py` script. Work Managers will be addressed in a later exercise.

8. Test the MedRec application.
   a. Direct your Web browser to the following URL:

```
http://localhost:7021/medrec/index.action
```

   b. In the **Patient** area of the home page, click the **Login** link:



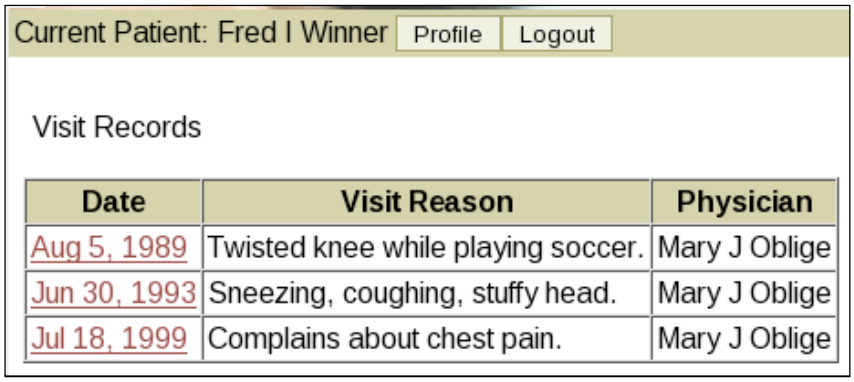
   c. Enter the following:

Field	Value
Email	fred@golf.com
Password	weblogic

   Click **Submit**.

   Tip: This version of the MedRec applications does not use the WebLogic security realm. Users are authenticated against a custom database table.

   d. Click the **Successfully logged in! Click here to continue** link.
   e. Click one of the links in the **Visit Records** area:



f. Click your browser's back button to return to the patient's main page. Test the remaining records.

g. Click the **Profile** button.

h. Edit the following fields:

Field	Value
Middle Name	Igor
Phone	9998887777

Click **Save**.

i. Click the **Logout** button.

Practice Solution

Perform the following tasks if you did not complete this practice and want to use the finished solution.

Solution Tasks

1. Launch the Lab Framework command shell by executing the `<STUDENT>/bin/prompt.sh` file.
2. Change the current directory to `<LAB>`.
3. Execute the following:

```
ant setup_solution
```

4. The Lab Framework performs the following:
   a. Makes a backup copy of your current work
   b. Starts the database
   c. Updates the database with the MedRec schema
   d. Shuts down any running servers
   e. Creates the MedRec domain from a template
   f. Starts the administration server and managed server 1
   g. Deploys the MedRec application
5. Locate the `<LAB>/solution/domains/MedRedDomain.jar` file. Copy it to `<WORK>/templates` for future exercises.

Practices for Lesson 4

Chapter 4

Overview of Practices for Lesson 4

Practices Overview

In the practices for this lesson, you will work with the WLS configuration MBeans of an offline domain, using WLST commands.

Practice 4-1: Work with Templates from the Command Line

Duration: 25 minutes

Objectives

After completing this practice, you should be able to:

- Open a domain template by using WLST
- Update an offline domain's configuration by using WLST
- Generate a template file by using WLST
- Enable configuration auditing and backups

Overview

Although the WebLogic Scripting Tool (WLST) is often used to monitor and update running domains and servers, it can also be used to update the configuration of offline domains and domain templates. In some instances, this offline approach can be more convenient than the alternative:

- Start your domain.
- Update the domain's configuration.
- Shut down the domain.
- Re-create a template.

In this practice, you will use WLST to open and edit the MedRec domain template created in prior exercises. You will perform several simple modifications to a server's SSL and log configurations. Lastly, you will use this opportunity to also enable domain configuration auditing and backups. These capabilities may help you recover from any configuration issues that you encounter in subsequent practices.

Dependencies

The following prior practice must be completed (or equivalent solutions run) before beginning this practice:

- *Create a Custom Domain Template*

Tasks

1. Open a domain template for editing from WLST.
   a. Shut down any running servers in your `MedRecDomain`.
   b. Execute the `<WORK>/domains/MedRecDomain/wlstPrompt.sh` script to launch WLST in interactive mode.
   c. Open your custom domain template:

```
readTemplate('<WORK>/templates/MedRecDomain.jar')
```

   d. Notice that the WLST prompt now indicates the name of the template:

```
wls:/offline/MedRecDomain>
```

   e. Enter the following commands:

```
cd('Servers')
ls()
```

   f. Confirm that there are three servers defined in the template.

2. Enable SSL for `MedRecSvr1`.
   a. Create a new SSL MBean for `MedRecSvr1`:

```
cd('MedRecSvr1')
create('MedRecSvr1','SSL')
cd('SSL/MedRecSvr1')
```

   b. Enable SSL and set its port number:

```
cmo.setEnabled(true)
cmo.setListenPort(7031)
```

   Tip: Remember that the built-in variable `cmo` is automatically set to the current MBean each time you use the `cd` command.
3. Configure logging for `MedRecSvr1`.
   a. Return to the main server MBean:

```
cd('../..')
```

   b. Create a new logging MBean for `MedRecSvr1`:

```
create('MedRecSvr1','Log')
cd('Log/MedRecSvr1')
```

   c. Tune the log rotation policies of `MedRecSvr1`:

```
cmo.setRotationType('bySize')
cmo.setFileMinSize(1000)
cmo.setNumberOfFilesLimited(true)
cmo.setFileCount(10)
cmo.setRotateLogOnStartup(true)
```

   d. Change the minimum severity of messages written to standard out:

```
cmo.setStdoutSeverity('Info')
```

4. Persist your template changes.
   a. Write the current in-memory domain configuration to a new template file named `NewMedRecDomain.jar`:

```
writeTemplate('<WORK>/templates/UpdatedMedRecDomain.jar')
```

   b. Exit WLST by using the `exit()` command.
   c. Open the new template JAR file by using the default archive browser and inspect the `config.xml` file. Verify your SSL and logging modifications to `MedRecSvr1`.
5. Enable configuration auditing and backups on your current domain.
   a. Restart your `MedRecDomain` administration server.
   b. Access the console and **Lock** it.
   c. In the Domain Structure panel, click the domain name.
   d. Click **Advanced**.
   e. Enter the following values:

Field	Value
Configuration Audit Type	Change Log

Field	Value
Configuration Archive Enabled	<checked>
Archive Configuration Count	10

Click **Save**.

f. Restart the administration server. During subsequent practices, configuration audit messages will be written to this server's log file and your configuration files will also be archived at `MedRecDomain/configArchive`.

Practice Solution

Perform the following tasks if you did not complete this practice and want to use the finished solution.

Solution Tasks

1. Launch the Lab Framework command shell by executing the file `<STUDENT>/bin/prompt.sh`.
2. Change the current directory to `<LAB>`.
3. Execute the following:

```
ant setup_solution
```

4. The Lab Framework executes a WLST script to read the domain template `MedRecDomain.jar` and create `UpdatedMedRecDomain.jar`.

Practices for Lesson 5

Chapter 5

Overview of Practices for Lesson 5

Practices Overview

In the practices for this lesson, you use WebLogic's network channel feature to customize the network addresses, ports, and protocols supported by a server.

Practice 5-1: Use Network Channels

Duration: 30 minutes

Objectives

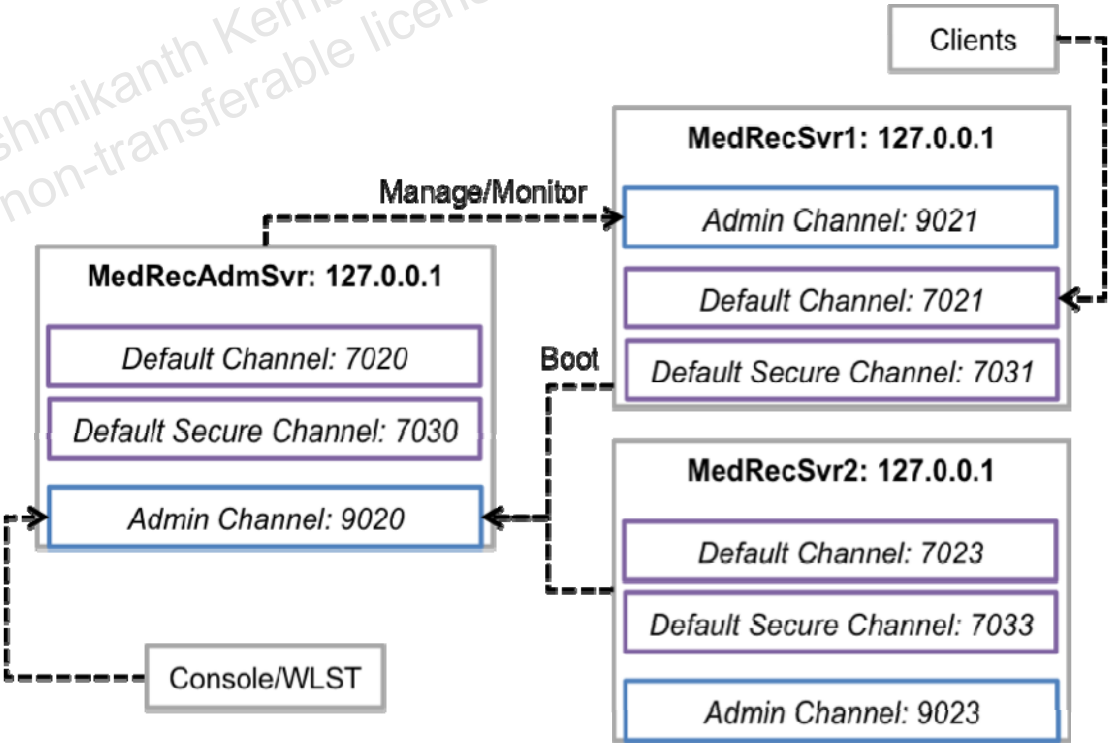
After completing this practice, you should be able to:

- Configure dedicated administration channels for servers
- Access an administration server by using its administration channel
- Use an administration channel to boot a managed server
- Start a managed server in standby mode

Overview

By default, an instance of WebLogic Server binds itself to all available network interfaces on a single port number. However, using WebLogic network channels, administrators have more fine-grained control over the specific interfaces used by a server and can bind a server to multiple ports. Administrators can also define the specific protocols supported on each network channel.

An administration channel is a special type of network channel that allows you to segregate administrative traffic from application traffic. These administrative channels also assume the use of SSL. In this practice, you will define administrative channels for each server in your domain and use these channels to boot, manage, and monitor the servers. You will also use this opportunity to experiment with WebLogic's standby mode feature. When in standby mode, a server's administration channel is open but all other channels are not. Because we are simulating an entire domain on a single machine, each server must use a different port number for its administration channel. This infrastructure is depicted as follows:



Dependencies

The following prior practice must be completed (or equivalent solutions run) before beginning this practice:

- *Create a Custom Domain Template*

Tasks

1. Configure SSL for all servers.
   a. Create a backup copy of the `<WORK>/domains/MedRecDomain` folder.
   b. Start your `MedRecDomain` administration server, if it is not already started. Kill any managed servers if they are running.
   c. Launch the administration console and log in: `http://localhost:7020/console`.
   d. **Lock** the console.
   e. In the **Domain Structure** panel, click **Environment > Servers**. Then, in the list of available servers, click **MedRecAdmSvr**.
   f. Enter the following values:

Field	Value
Listen Address	127.0.0.1
SSL Listen Port Enabled	<checked>
SSL Listen Port	7030

   Click **Save**.
   g. Repeat the previous steps to configure SSL for **MedRecSvr1**:

Field	Value
Listen Address	127.0.0.1
SSL Listen Port Enabled	<checked>
SSL Listen Port	7031

   h. Repeat the previous steps to configure SSL for **MedRecSvr2**:

Field	Value
Listen Address	127.0.0.1
SSL Listen Port Enabled	<checked>
SSL Listen Port	7033

2. Configure the default administration channel port.
   a. In the **Domain Structure** panel, click the domain name, **MedRecDomain**.
   b. Enter the following values:

Field	Value
Enable Administration Port	<checked>
Administration Port	9020

   Click **Save**.
3. Override the administration channel port for managed servers on the same machine.

a. Edit the configuration for `MedRecSvr1` again.
b. Click the **Advanced** link at the bottom of the page to view additional options.
c. Locate the field named **Local Administration Port Override**. Set its value to **9021** and click **Save**.
d. Repeat the previous steps to change the **Local Administration Port Override** for `MedRecSvr2`. Set its value to **9023**.
e. **Activate** your console changes.
f. Kill the administration server and your browser.

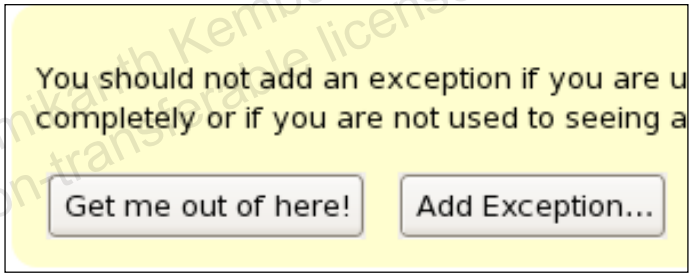
4. Administer the domain by using the administration channel.
a. Restart the administration server.
b. Confirm that the following message was written to the server log:

```
<Notice> <Server> <BEA-002613> <Channel "DefaultAdministration"
is now listening on 127.0.0.1:9020 for protocols admin, ldaps,
https.>
```

c. Restart your browser and attempt to access the console by using the administration server's default channel (port 7020).
d. Confirm that your browser displays the following message:

```
Console/Management requests ... can only be made through an
administration channel.
```

e. Access the console using the new secure administration channel (note the protocol change): **`https`**`://localhost:`**`9020`**`/console`.
f. When prompted about a bad certificate by the browser, click the **Add an Exception** link. Then click the **Add Exception** button:



g. Click the **Get Certificate** button, and then click **Confirm Security Exception**.
h. After logging in, **Lock** the console and edit the configuration for `MedRecSvr1` once again.
i. Click the **Advanced** link to view additional options.
j. Locate the field named **Startup Mode**. Set its value to **Standby**.
k. Click **Save** and then **Activate** your changes.

5. Connect managed servers to the Admin Server using the admin channel.
a. Edit the `<WORK>/domains/MedRecDomain/startServer1.sh` file.
b. Update the URL used to connect to the Administration Server:

```
./startManagedWebLogic.sh MedRecSvr1 https://127.0.0.1:9020
```

c. Repeat the previous steps to update the `startServer2.sh` file as well.
d. Use the `startServer1.sh` script to boot `MedRecSvr1`.
e. Confirm that the following messages were written to the `MedRecSvr1` log:

```
<Notice> <Server> <BEA-002613> <Channel "DefaultAdministration"
is now listening on 127.0.0.1:9021 for protocols admin, ldaps,
https.>

<Notice> <WebLogicServer> <BEA-000360> <Server started in
STANDBY mode>
```

6. Work with managed servers in Standby mode.
   a. Attempt to access the MedRec application on `MedRecSvr1`: `http://localhost:7021/medrec/index.action`
   b. Confirm that the request fails because `MedRecSvr1` is in Standby mode.
   c. Return to the administration console.
   d. In the **Domain Structure** panel, click **Environment > Servers**.
   e. Click the **Control** tab.
   f. Select the check box for **MedRecSvr1** and click the **Resume** button:

   g. When prompted, click **Yes**.
   h. Confirm that the following messages were written to the `MedRecSvr1` log:

```
<Notice> <Server> <BEA-002613> <Channel "Default" is now
listening on 127.0.0.1:7021 for protocols iiop, t3, ldap, snmp,
http.>

<Notice> <WebLogicServer> <BEA-000365> <Server state changed to
RUNNING>
```

   i. Verify that the MedRec application is now accessible.
7. Start an application in Admin mode
   a. Return to the administration console.
   b. In the **Domain Structure** panel, click **Deployments**.
   c. Select the check box next to the **medrec** application and click **Stop > Stop, but continue servicing administration requests**. Click **Yes** when asked if you are sure.
   d. Notice the state of the application is now "Admin."
   e. Attempt to access the MedRec application on `MedRecSvr1` in the usual way: `http://localhost:7021/medrec/index.action`

      Note: It should not work.
   f. Attempt to access the MedRec application on `MedRecSvr1` again through the server's admin port by using `https`:

      `https://localhost:9021/medrec/index.action`

g. Go through the previous steps to add a “bad certificate” exception to the browser.

h. When asked to log in, use `weblogic/Welcome1`.

Note: It should work this time. You are accessing the application through the server’s admin port.

i. Now place the application in its normal, active state. Return to the administration console. In the **Domain Structure** panel, click **Deployments**. Select the check box next to the **medrec** application and click **Start > Servicing all requests**. Click **Yes** when asked if you are sure. Notice the state of the application is now “Active.”

8. Clean up.

a. Kill all of your servers.

b. Delete the `<WORK>/domains/MedRecDomain` folder.

c. Move the backup copy of `MedRecDomain` that you created at the start of the exercise to `<WORK>/domains`.

d. Close any open Linux terminals.

Why? Servers started from existing terminals may continue to use the deleted copy of your domain in the trash.

Practice Solution

No solution exists for this practice. You must complete all of the instructions.

Practices for Lesson 6

Chapter 6

Overview of Practices for Lesson 6

Practices Overview

In the practices for this lesson, you update your existing MedRec WebLogic infrastructure to support database clustering.

Practice 6-1: Use a Multi Data Source for Failover

Duration: 30 minutes

Objectives

After completing this practice, you should be able to:

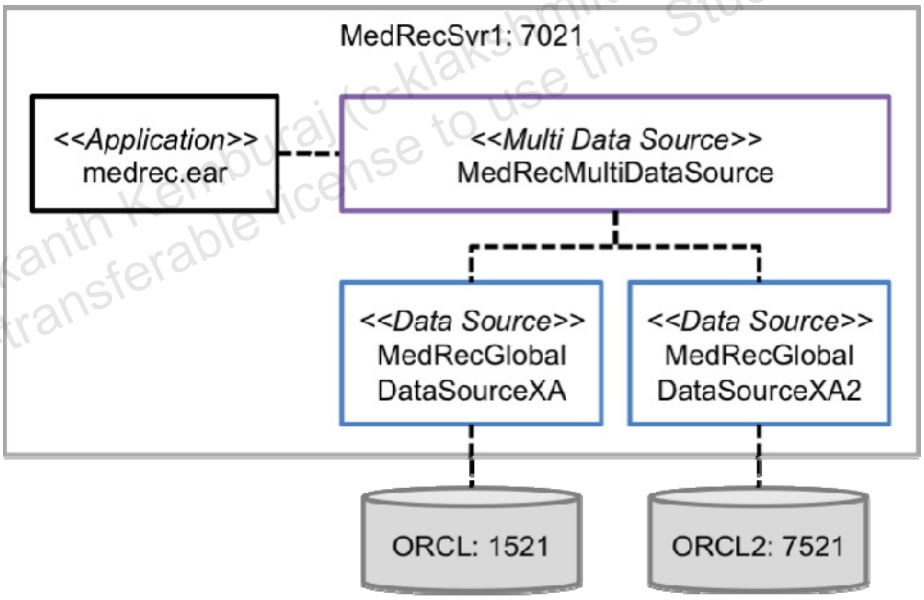
- Enable data source connection testing
- Define and target a multi data source
- Manage multi data source membership

Overview

Using WebLogic multi data sources, applications transparently connect to a grid of redundant database instances. Multi data sources are used to distribute database connection requests across these instances, or they can simply provide failover when one instance becomes unavailable.

The MedRec environment has been upgraded to include a backup database instance, which is synchronized with the primary instance every 12 hours to provide a higher level of availability. If the primary database becomes unavailable and the MedRec application requests a database connection, WebLogic will begin using the backup instance.

The following shows the MedRec multi data source architecture:



Dependencies

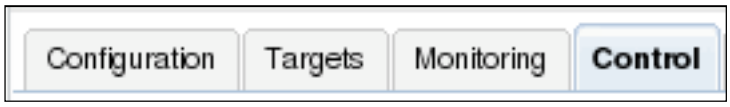
The following prior practice must be completed (or equivalent solutions run) before beginning this practice:

- *Create a Custom Domain Template*

Tasks

1. Configure connection testing on the original data source and update its JNDI name.
   a. Start the backup database by using `<STUDENT>/bin/startDB2.sh`.
   b. Start your `MedRecAdmSvr` and `MedRecSvr1`, if they are not already started.

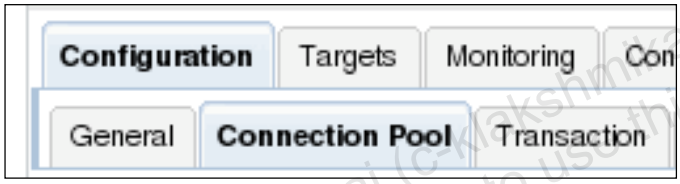
c. Log in to the administration console and **Lock** it.
d. In the **Domain Structure** panel, navigate to **Services > JDBC > Data Sources**.
e. Click **MedRecGlobalDataSourceXA** to edit its configuration.
f. Click the **Control** tab:



g. Select the check box for **MedRecSvr1**, and click **Shutdown > Force Shutdown**. When prompted, click **Yes**.
h. Click the **Targets** tab.
i. Clear all check boxes and click **Save**.
j. **Activate** your changes.
k. **Lock** the console once again and return to the **Configuration** tab.
l. Edit the **JNDI Name** field. Change the value to the following:

```
jdbc/MedRecGlobalDataSourceXA-Node1
```

Click **Save**.
m. Click the **Configuration > Connection Pool** tab:



n. Click the **Advanced** options.
o. Enter the following values:

Field	Value
Test Connections on Reserve	<checked>
Test Table Name	WLSDATA.PATIENTS

Click **Save**.
p. Return to the **Targets** tab. Select the check box for **MedRecSvr1** to target the data source back to it. Click **Save**.
q. **Activate** your changes.

2. Create a second data source.
a. If one is not already open, launch a Lab Framework command shell by executing the `<STUDENT>/bin/prompt.sh` file.
b. Navigate to `<LAB>/resources/jdbc` and execute the `createDataSource2.py` WLST script.
c. Confirm that the script executed successfully:

```
Data Source created successfully.
```

d. Return to the console and inspect the new data source:

	Name	JNDI Name
☐	MedRecGlobalDataSourceXA	jdbc/MedRecGlobalDataSourceXA-Node1
☐	MedRecGlobalDataSourceXA2	jdbc/MedRecGlobalDataSourceXA-Node2

3. Create a multi data source.
   a. **Lock** the console.
   b. In the **Domain Structure** panel, navigate to **Services > JDBC > Multi Data Sources**.
   c. Click the **New** button.
   d. Enter the following values:

Field	Value
Name	MedRecMultiDataSource
JNDI Name	jdbc/MedRecGlobalDataSourceXA
Algorithm Type	Failover

   Click **Next**.
   e. Target the resource to **MedRecSvr1** and click **Next**.
   f. Select the **XA Driver** option and click **Next**.
   g. Use the  button to add the following data sources to the **Chosen** list:
      – `MedRecGlobalDataSourceXA` (be sure that it is listed first)
      – `MedRecGlobalDataSourceXA2`
   h. Click **Finish**. Then **Activate** your changes.
   i. Kill and restart `MedRecSvr1`.
   Why? The restart is necessary to clear out any cached JNDI objects in the MedRec application.
4. Test data source failover.
   a. Launch the MedRec application: `http://localhost:7021/medrec/index.action`.
   b. Log in as the `fred@golf.com` patient, using the password `weblogic`.
   c. Click the **Successfully logged in! Click here to continue** link to view a list of medical records:

Current Patient: Fred I Winner | Profile | Logout

Visit Records

Date	Visit Reason	Physician
Aug 5, 1989	Twisted knee while playing soccer.	Mary J Oblige
Jun 30, 1993	Sneezing, coughing, stuffy head.	Mary J Oblige
Jul 18, 1999	Complains about chest pain.	Mary J Oblige

d. Use the `<STUDENT>/bin/stopDB1.sh` script to shut down the original database.
e. Return to the MedRec application. Select one of the medical records in the list.
f. Click the **Profile** button.
g. Click **Save**. The application should continue to function as normal.
h. Confirm that an error message is added to the `MedRecSvr1` log:

```
<Warning> <JDBC> <BEA-001129> <Received exception while creating
connection for pool "MedRecGlobalDataSourceXA": The Network
Adapter could not establish the connection>.
```

5. Decommission the backup database.
   a. Restart the database by using the `<STUDENT>/bin/startDB1.sh` script.
   b. Return the console and **Lock** it.
   c. Edit the multi data source.
   d. Click **Configuration > Data Sources**.
   e. Use the  button to remove **MedRecGlobalDataSourceXA2** from this multi data source, and click **Save**.
   f. Edit the **MedRecGlobalDataSourceXA2** data source.
   g. Click the **Targets** tab.
   h. Clear all check boxes and click **Save**.
   i. **Activate** your changes.
   j. Use the `<STUDENT>/bin/stopDB2.sh` script to shut down the backup database.

Practice Solution

Perform the following tasks if you did not complete this practice and want to use the finished solution.

Solution Tasks

1. Launch the Lab Framework command shell by executing the `<STUDENT>/bin/prompt.sh` file.
2. Change the current directory to `<LAB>`.
3. Execute the following:

```
ant setup_solution
```

4. The Lab Framework performs the following:
   a. Makes a backup copy of your current work
   b. Runs the solutions for any prerequisite practices, if they have not been run previously
   c. Starts both databases if they are not already started
   d. Starts the administration server and managed server 1 if they are not already started
   e. Uses WLST to create the data sources
   f. Uses WLST to create the multi data source
5. When finished with this practice, complete the section entitled “Decommission the backup database.”

Practices for Lesson 7

Chapter 7

Overview of Practices for Lesson 7

Practices Overview

There are no practices for this lesson.

Practices for Lesson 8

Chapter 8

Overview of Practices for Lesson 8

Practices Overview

There are no practices for this lesson.

Practices for Lesson 9

Chapter 9

Overview of Practices for Lesson 9

Practices Overview

In the practices for this lesson, you customize the default persistent store behavior for WebLogic JMS resources.

Practice 9-1: Configure JMS Persistence

Duration: 35 minutes

Objectives

After completing this practice, you should be able to:

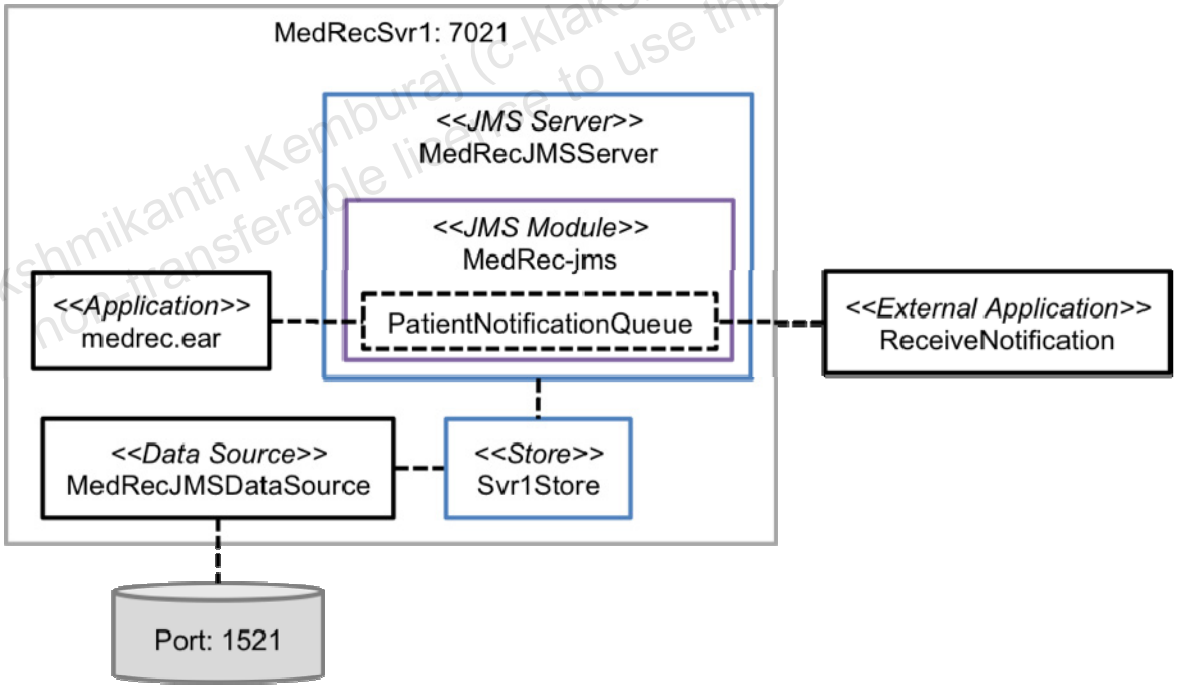
- Create a new JDBC persistent store for a server
- Associate a JMS server with a persistent store
- Configure persistence settings for JMS resources
- Verify message persistence across server restarts

Overview

The MedRec application allows patients to register themselves. However, before patients can access their records, the site administrator must approve the registration request. MedRec uses JMS to asynchronously send email and other types of notifications to patients, including the results of registration requests.

These notifications are of a critical nature and therefore must be delivered despite any interruptions in the JMS service or application server. WebLogic Server has the ability to persist JMS messages to either file systems or relational databases to help guarantee message delivery.

The following is the MedRec JMS architecture:



Dependencies

The following prior practice must be completed (or equivalent solutions run) before beginning this practice:

- *Create a Custom Domain Template*

Tasks

1. Create a data source for the persistent store.
   a. Start your `MedRecAdmSvr` and `MedRecSvr1`, if they are not already started.
   b. If one is not already open, launch a new Lab Framework command shell by executing the `<STUDENT>/bin/prompt.sh` file.
   c. Navigate to `<LAB>/resources/jdbc` and execute the `createJMSStoreDataSource.py` WLST script.
   d. Confirm that the script executed successfully:

```
Data Source created successfully.
```

   Tip: The account used by this data source has create privileges, so the required table for the JDBC persistent store will not need to be created manually.
2. Create a JDBC persistent store.
   a. Log in to the administration console and **Lock** it.
   b. In the **Domain Structure** panel, navigate to **Services > Persistent Stores**.
   c. Click **New > Create JDBCStore**.
   d. Enter the following values:

Field	Value
Name	Svr1Store
Target	MedRecSvr1
Data Source	MedRecJMSDataSource
Prefix Name	WLSDATA.SVR1_ (note the underscore)

   Click **OK**.
   e. **Activate** your changes.
   f. Locate the following new message in the `MedRecSvr1` log:

```
<Notice> <Store> <BEA-280067> <JDBC store "Svr1Store" did not
find a database table at "WLSDATA.SVR1_WLStore", so it created
one using the commands in file
"/weblogic/store/io/jdbc/ddl/oracle.ddl".>
```

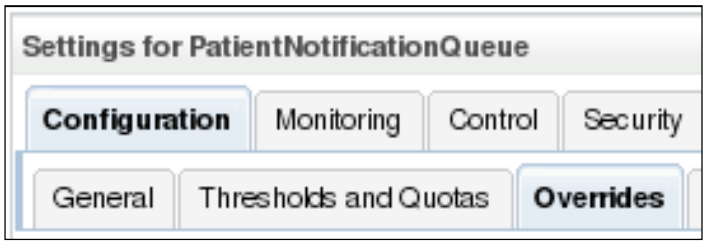
   g. Launch a new Linux terminal.
   h. Use **sqlplus** to confirm the presence of the new `SVR1_WLStore` table. For example:

```
> sqlplus / as sysdba
SQL> SELECT COUNT(*) FROM WLSDATA.SVR1_WLSTORE;

COUNT(*)
--------
17
```

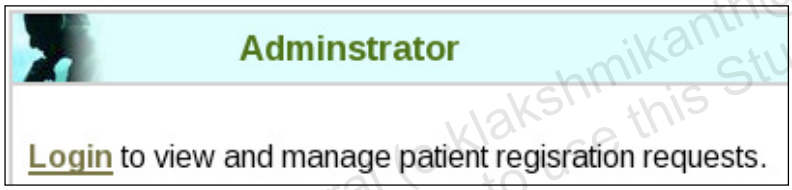
3. Configure persistence for JMS resources.
   a. Return to the console and **Lock** it.
   b. In the **Domain Structure** panel, navigate to **Services > Messaging > JMS Servers**.
   c. Click **MedRecJMSServer**.
   d. Locate the **Persistent Store** field, and select the value **Svr1Store**. Then click **Save**.

e. In the **Domain Structure** panel, navigate to **Services > Messaging > JMS Modules**.
f. Click **MedRec-jms**.
g. In the **Summary of Resources** table, click **PatientNotificationQueue**.
h. Click the **Configuration > Overrides** tab:



i. Locate the **Delivery Mode Override** field, and select the **Persistent** option. Then click **Save**.
j. **Activate** your changes.
k. Kill and restart `MedRecSvr1`.

4. Produce and consume a JMS message.
a. Launch the MedRec application.
b. In the *Administrator* section of the home page, click **Login**:



c. Log in using **admin@avitek.com** as the username and **weblogic** as the password.
d. Click the **View Pending Requests** link.
e. Click the registration request for the new patient, **charlie@star.com**.
f. Click the **Approve** button.
g. Return to the console. Locate and select the **PatientNotificationQueue** JMS destination once again.
h. Click the **Monitoring** tab.
i. Notice that the **Messages Current** column should indicate that a message is waiting in the queue to be consumed:

	Name	Messages Current	Messages Pending	Messages Total
☐	MedRec-jms!PatientNotificationQueue	1	0	1

j. Return to your Lab Framework command shell.
k. Navigate to `<LAB>/resources/client` and execute the following:

```
java com.bea.medrec.ReceiveNotification t3://localhost:7021
```

l. Confirm that a message was received by the client application:

```
Message Received: (Charlie,charlie@star.com,APPROVED)
```

m. Return the console and refresh your browser. The **Messages Current** column should have decreased by a value of 1.

n. Press **Enter** to quit the client application.

5. Test message persistence across server restart.

a. From a Linux terminal execute the `<LAB>/resources/updateStatus.sh` script. This action resets the status of the `charlie@star.com` account so that it can be approved again.

b. Return to the MedRec application, and click the **View Pending Requests** link again:

c. Repeat the prior steps to approve the patient registration request again.

d. Return to the console. Repeat the prior steps to confirm that a new message is waiting in the queue.

e. Kill and restart `MedRecSvr1`.

f. Use the console to confirm that the message is still available in the queue.

g. Repeat the prior steps to run the `ReceiveNotification` client application again.

Practice Solution

Perform the following tasks if you did not complete this practice and want to use the finished solution.

Solution Tasks

1. Launch the Lab Framework command shell by executing the `<STUDENT>/bin/prompt.sh` file.
2. Change the current directory to `<LAB>`.
3. Execute the following:

```
ant setup_solution
```

4. The Lab Framework performs the following:
   a. Makes a backup copy of your current work
   b. Runs the solutions for any prerequisite practices, if they have not been run previously
   c. Starts the database if it is not already started
   d. Removes any existing JDBC persistent store tables from the database
   e. Starts the administration server and managed server 1 if they are not already started
   f. Uses WLST to create a new JDBC data source
   g. Uses WLST to create the JDBC persistent store
   h. Uses WLST to update the JMS server
   i. Uses WLST to update `PatientNotificationsQueue` in the JMS module
5. Kill and restart `MedRecSvr1`.

Practices for Lesson 10

Chapter 10

Overview of Practices for Lesson 10

Practices Overview

There are no practices for this lesson.

Practices for Lesson 11

Chapter 11

Overview of Practices for Lesson 11

Practices Overview

In the practices for this lesson, you use WebLogic's Store and Forward JMS feature to automatically move messages produced on one server to another server.

Practice 11-1: Store and Forward JMS Messages

Duration: 45 minutes

Objectives

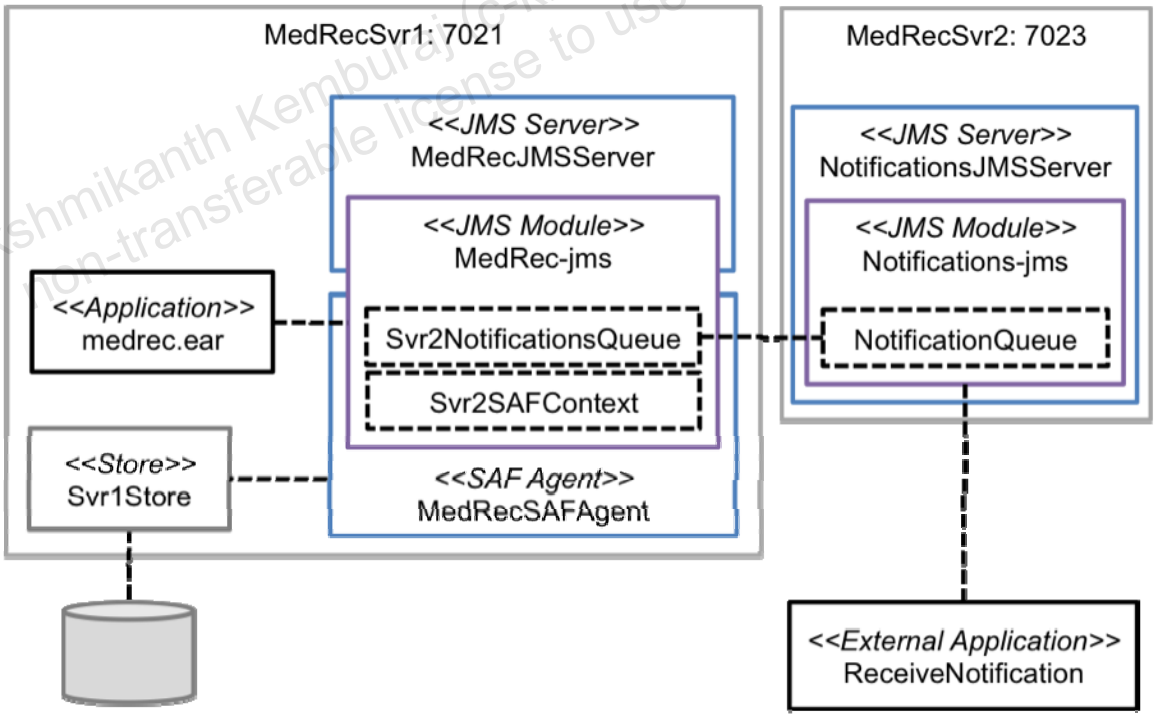
After completing this practice, you should be able to:

- Create and monitor an SAF agent
- Configure SAF resources in a JMS module
- Define an SAF context to a remote server
- Forward messages to a remote queue
- Verify message storage across server restarts

Overview

Currently in our testing environment any patient notification messages are published by the MedRec application to a local JMS queue, from which they are also consumed. But in the final production environment, messages must instead be routed to a dedicated notifications server, in this case `MedRecSvr2`. To avoid any modifications to our existing MedRec application or notification consumer applications, you will use WebLogic Server's store and forward (SAF) capabilities. The MedRec application will continue to look up the required JMS queue from the local JNDI tree, but any published messages will be automatically forwarded to a remote queue on `MedRecSvr2`.

The following shows the MedRec JMS store and forward architecture:



Dependencies

The following prior practice must be completed (or equivalent solutions run) before beginning this practice:

- *Configure JMS Persistence*

Tasks

1. Initialize JMS resources on `MedRecSvr2`.
   a. Start your `MedRecAdmSvr` and `MedRecSvr1`, if they are not already started.
   b. Start `MedRecSvr2` by using the `<WORK>/domains/MedRecDomain/startServer2.sh` script.
   c. If one is not already open, launch a new Lab Framework command shell by executing the `<STUDENT>/bin/prompt.sh` file.
   d. Navigate to `<LAB>/resources/jms` and execute the `createJMSServerAndModule.py` WLST script.
   e. Confirm that the script executed successfully:

```
JMS Server and Module created successfully.
```

   f. Launch the administration console.
   g. Navigate to **Services > Messaging > JMS Modules**. Click the **Notifications-jms** module.
   h. Scroll down to the **Summary of Resources** section and inspect this module's contents:

Name	Type	JNDI Name
NotificationFactory	Connection Factory	com.bea.medrec.email.NotificationFactory
NotificationQueue	Queue	com.bea.medrec.email.NotificationQueue

2. Create an SAF agent on `MedRecSvr1`.
   a. **Lock** the console.
   b. Navigate to **Services > Messaging > Store-and-Forward Agents**.
   c. Click **New**.
   d. Enter the following values:

Field	Value
Name	MedRecSAFAgent
Persistent Store	Svr1Store
Agent Type	Sending-only

   Click **Next**.
   e. Select the **MedRecSvr1** check box and click **Finish**.
3. Remove the local queue from the JMS module.
   a. Navigate to **Services > Messaging > JMS Modules**. Click the **MedRec-jms** module.
   b. Select the check box for the **PatientNotificationQueue** resource and click **Delete**. When prompted if you are sure, click **Yes**.
   c. **Activate** your changes.
4. Add an SAF context to a JMS module for `MedRecSvr2`.
   a. **Lock** the console again.

b. Click **New**.

Tip: You should still be under **Services > Messaging > JMS Modules > MedRec-jms**

c. Select the **Remote SAF Context** option and click **Next**.

d. Enter the following values:

Field	Value
Name	Svr2SAFContext
URL	t3://localhost:7023
User Name	weblogic
Password, Confirm Password	Welcome1

Click **OK**.

5. Add an SAF destination container to a JMS module.
   a. Click **New** once again.
   b. Select the **SAF Imported Destinations** option and click **Next**.
   c. Enter the following values:

Field	Value
Name	Svr2SAFDestinations
Remote SAF Context	Svr2SAFContext

Click **Next**.

   d. Click **Advanced Targeting**.
   e. Click **Create a New Subdeployment**.
   f. Name the new subdeployment **DeployToMedRecSAFAgent** and click **OK**.
   g. Select the **MedRecSAFAgent** check box and click **Finish**.
6. Define an SAF queue within the SAF destination container.
   a. Select your new **Svr2SAFDestnations** resource.
   b. Click the **Configuration > Queues** tab.
   c. Click **New**.
   d. Enter the following values:

Field	Value
Name	Svr2NotificationQueue
Remote JNDI Name	com.bea.medrec.email.NotificationQueue

Click **OK**.

   e. Select your new **Svr2NotificationQueue**.
   f. Locate the field **Local JNDI Name** and set its value to **com.bea.medrec.jms.PatientNotificationQueue**. This step allows the MedRec application to continue using the same JNDI names to access JMS resources.
   g. Click **Save**.
   h. **Activate** all of your changes.
7. Generate a patient notification message.

a. From a Linux terminal execute the `<LAB>/resources/updateStatus.sh` script. This action resets the status of the `charlie@star.com` account so that it can be approved again.
b. Launch the MedRec application.
c. In the **Administrator** section of the home page, click **Login**.
d. Log in using **admin@avitek.com** as the username and **weblogic** as the password.
e. Click the **View Pending Requests** link.
f. Click the registration request for the new patient **charlie@star.com**.
g. Click the **Approve** button.

8. Monitor and consume forwarded messages.
   a. Return to the console.
   b. Navigate to **Services > Messaging > Store-and-Forward Agents**. Click **MedRecSAFAgent**.
   c. Click the **Monitoring** tab.
   d. Click **Customize this table**. Add the following columns from the **Available** list to the **Chosen** list:
      – Messages Current
      – Messages Received
   e. Click **Apply**.
   f. Confirm that the **Messages Received** value has increased by 1:

	Name	Messages Received	Remote Endpoints Current
☐	MedRecSAFAgent	1	1

   g. Navigate to **Services > Messaging > JMS Modules**. Click the **Notifications-jms** module.
   h. Click **NotificationQueue**.
   i. Click the **Monitoring** tab.
   j. Confirm that the **Messages Current** value has increased by 1:

	Name	Messages Current	Messages Pending	Messages Total
☐	Notifications-jms!NotificationQueue	1	0	1

   k. Return to your Lab Framework command shell.
   l. Navigate to `<LAB>/resources/client` and execute the following to consume messages from `MedRecSvr2`:

```
java com.bea.medrec.ReceiveNotification t3://localhost:7023
```

   m. Confirm that a message was received:

```
Message Received: (Charlie,charlie@star.com,APPROVED)
```

n. Press **Enter** to quit the client application.

9. Test SAF high availability.
   a. Kill `MedRecSvr2`.
   b. Repeat the section of the instructions entitled “Generate a patient notification message.”
   c. Return to the console. Repeat the prior steps to monitor the `MedRecSAFAgent` again.
   d. Using the **Messages Current** column, confirm that a new message is waiting to be forwarded:

	Name	Messages Received	Messages Current
☐	MedRecSAFAgent	2	1

   e. Click the **Monitoring > Remote Endpoints** tab.
   f. Click the link found in the **Name** column of the table.
   g. Inspect the **Timestamp** of the stored message:

	ID	Type	CorrId	Priority	Timestamp
☐	ID:<706147.1237831405807.0>			4	Mon Mar 23 14:03:25 EDT 2009

   Tip: Because the message payload is a Java object and not text, you will not be able to view it in the console by using the link in the **ID** column.
   h. Kill `MedRecSvr1`.
   Why: Because the SAF agent is associated with a persistent store, the pending message will not be lost.
   i. Restart `MedRecSvr2`. When it is running, restart `MedRecSvr1`.
   Why: Now that both the SAF agent and remote destination are available again, the pending message can be delivered.
   j. Return to the console and monitor your SAF agent once again.
   k. Verify that the **Messages Current** column is now 0, because all messages have been forwarded.
   l. Repeat the prior steps to monitor the `NotificationQueue` again. Confirm that a new message is waiting to be consumed.
   m. Repeat the prior steps to run the `ReceiveNotification` client application a final time.
   n. When you are finished with the practice, you can shut down `MedRecSvr2`.

Practice Solution

Perform the following tasks if you did not complete this practice and want to use the finished solution.

Solution Tasks

1. Launch the Lab Framework command shell by executing the `<STUDENT>/bin/prompt.sh` file.
2. Change the current directory to `<LAB>`.
3. Execute the following:

```
ant setup_solution
```

4. The Lab Framework performs the following:
   a. Makes a backup copy of your current work
   b. Runs the solutions for any prerequisite practices, if they have not been run previously
   c. Starts the database if it is not already started
   d. Starts the administration and both managed servers if it is not already started
   e. Uses WLST to create a new JMS server and module on `MedRecSvr2`
   f. Uses WLST to create the SAF agent
   g. Uses WLST to remove a queue from the `MedRecSvr1` JMS module
   h. Uses WLST to create SAF resources in the `MedRecSvr1` JMS module
5. When finished with the practice, you can shut down `MedRecSvr2`.

Practices for Lesson 12

Chapter 12

Overview of Practices for Lesson 12

Practices Overview

In the practices for this lesson, you will use the WebLogic JMS bridge feature to automatically transfer messages between WLS and a third-party JMS product.

Naming Conventions

The practices in this lesson use the following variable names to refer to commonly used locations on your file system:

Variable	Path
<MQ_HOME>	/u01/app/opensource/MessageQueue

Practice 12-1: Bridge JMS Providers

Duration: 40 minutes

Objectives

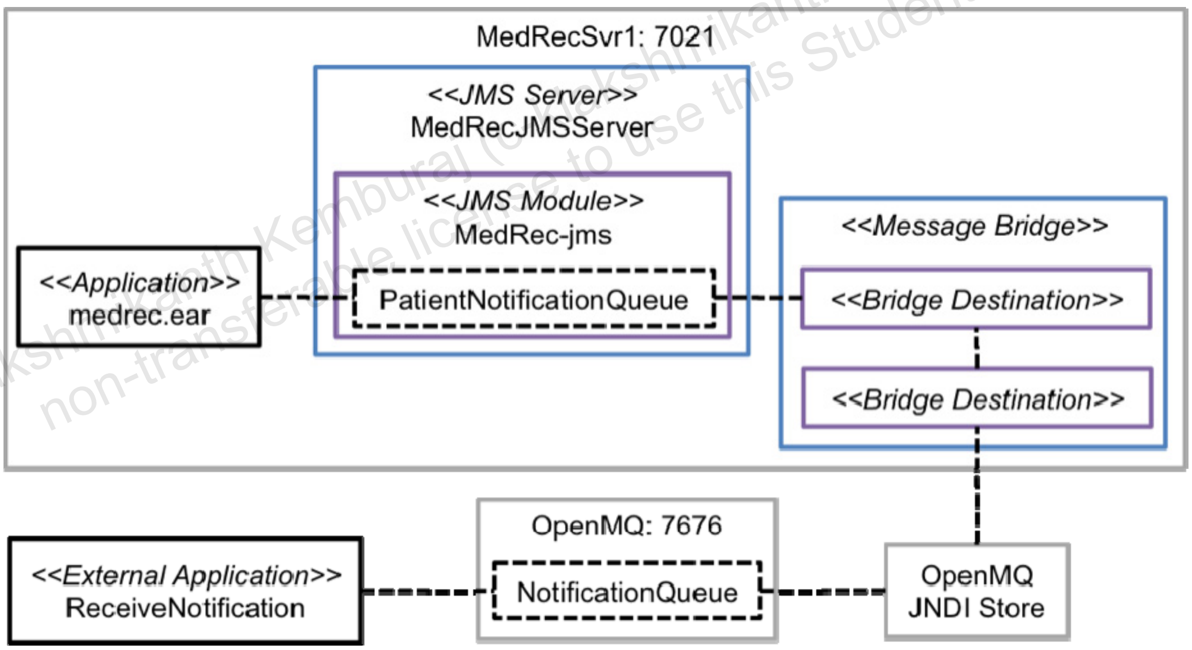
After completing this practice, you should be able to:

- Define a bridge destination for WebLogic Server JMS
- Define a bridge destination for a third-party JMS provider
- Configure a message bridge between destinations

Overview

Currently, in our testing environment any patient notification messages are published by the MedRec application to a local JMS queue. However, MedRec has already invested in an open source JMS provider to support many other applications. Although IT desires to eventually migrate this existing JMS infrastructure to WebLogic, this is not feasible in the short term. Therefore, we will use WebLogic's message bridging feature to integrate the two JMS providers. The MedRec application will continue to publish notification messages to the local queue, but they will be automatically forwarded to the production queue on the open source JMS provider.

The following is the MedRec architecture that is used to bridge WebLogic JMS and OpenMQ:



Dependencies

The following prior practice must be completed (or equivalent solutions run) before beginning this practice:

- *Create a Custom Domain Template*

Tasks

1. Start the OpenMQ message server.
   a. Launch two new Linux terminals.
   b. Within each terminal, navigate to `<MQ_HOME>/mq/bin`.

Tip: For convenience, `MQ_HOME` is also an OS environment variable.

c. In one terminal, execute the `imqbrokerd` script.

d. Confirm that the message server has started on port 7676.

e. In the other terminal, execute the following:

```
./imqcmd metrics dst -t q -n NotificationQueue
```

f. When prompted, enter the credentials `admin/admin`.

g. Note the current values of the **Msgs In** and **Msg Count Current** columns:

```
---------------------------------------
Msgs      Msg Bytes        Msg Count
In  Out    In     Out    Current   Peak
---------------------------------------
0     0     0       0        0        0
```

h. Terminate the process to end the display of metrics.

2. Include OpenMQ client libraries in the server classpath.

a. Shut down all `MedRecDomain` servers if they are running.

b. Locate the following files found at `<MQ_HOME>/mq/lib`:

- `fscontext.jar`
- `imq.jar`

c. Copy these files to `<WORK>/domains/MedRecDomain/lib`.

d. Restart `MedRecAdmSvr` and `MedRecSvr1`.

3. Disable any Store and Forward (SAF) resources.

a. If one is not already open, launch a new Lab Framework command shell by executing the `<STUDENT>/bin/prompt.sh` file.

b. Navigate to `<LAB>/resources/jms` and execute the `deleteSAFDestinations.py` WLST script.

Why? This script removes any Store and Forward (SAF) destinations created in prior exercises. In this exercise the message bridging features will be used instead to route messages produced by the MedRec application.

c. Execute the `createQueue.py` WLST script.

Why? This script restores the standard MedRec JMS queue if you deleted it as part of the SAF exercises.

4. Create the local bridge destination.

a. Launch the administration console and **Lock** it.

b. Navigate to **Services > Messaging > Bridges > JMS Bridge Destinations**.

c. Click **New**.

d. Enter the following values:

Field	Value
Name	Svr1PatientQueueBridgeDest
Adapter JNDI Name	eis.jms.WLSConnectionFactoryJNDIXA
Connection URL	t3://localhost:7021

Field	Value
Connection Factory JNDI Name	weblogic.jms.XAConnectionFactory
Destination JNDI Name	com.bea.medrec.jms.PatientNotificationQueue

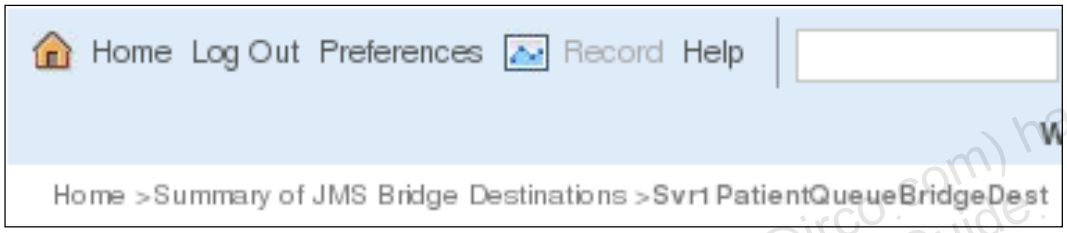
Click **OK**.

e. Click the new bridge destination.

f. Set the **User Name** and **User Password** to the domain's credentials (`weblogic/Welcome1`). Then click **Save**.

5. Create the OpenMQ bridge destination.

a. Using the locator link trail at the top of the console, click the **Summary of JMS Bridge Destinations** link:



b. Click **New**.

c. Enter the following values:

Field	Value
Name	OpenMQNotifyQueueBridgeDest
Adapter JNDI Name	eis.jms.WLSConnectionFactoryJNDIXA
Connection URL	file:///u01/app/opensource/MessageQueue/jndi-store (*note the three slashes "///"*)
Connection Factory JNDI Name	NotificationFactory
Destination JNDI Name	NotificationQueue

Click **OK**.

d. Click the new bridge destination.

e. Modify the following values:

Field	Value
Initial Context Factory	com.sun.jndi.fscontext.RefFSContextFactory
User Name	admin
User Password, Confirm User Password	admin

Click **Save**.

f. **Activate** your changes.

6. Create a message bridge.

a. **Lock** the console once again.

b. Navigate to **Services > Messaging > Bridges**.

c. Click **New**.

d. Enter the following values:

Field	Value
Name	OpenMQBridge
Quality of Service	Exactly-once
Started	<checked>

Click **Next**.

e. Select the **Svr1PatientQueueBridgeDest** option and click **Next**.

f. Use the default **WebLogic Server 7.0 or Higher** option and click **Next**.

g. Select the **OpenMQNotifyQueueBridgeDest** option and click **Next**.

h. Select the **Other JMS** option and click **Next**.

i. Target the bridge to **MedRecSvr1** and click **Next**.

j. Click **Finish**. Then **Activate** your changes.

7. Test the bridge.

a. Repeat the prior steps to run the `imqcmd` utility and view the latest metrics for the `NotificationQueue`. Leave the program running.

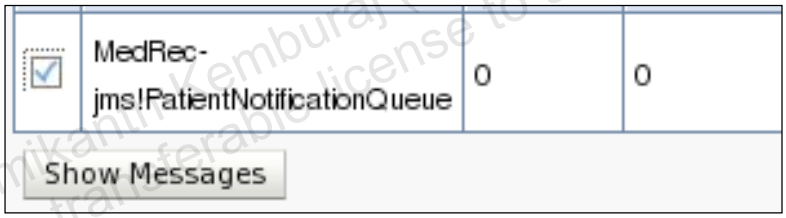
b. Return to the console.

c. Navigate to **Services > Messaging > JMS Modules**. Click **MedRec-jms**.

d. Click **PatientNotificationQueue**.

e. Click the **Monitoring** tab.

f. Select the check box for the queue and click **Show Messages**:



g. Click **New**.

h. Enter the following values:

Field	Value
Type	Text
Delivery Mode	Persistent
Body	Bridge Test

Click **OK**.

i. Return to the running `imqcmd` program. Confirm that a new message has been received. The values of the **Msgs In** and **Msg Count Current** columns should be incremented by 1:

```
-------------------------------------
Msgs      Msg Bytes        Msg Count
In  Out   In    Out    Current  Peak
-------------------------------------
1     0     0       0        1       0
```

j. Return to the Lab Framework command prompt.
k. Navigate to `<LAB>/resources/client`.
l. Execute the following to consume a message from the OpenMQ destination:

```
source setEnv.sh

java com.bea.medrec.ReceiveNotification
   file:///u01/app/opensource/MessageQueue/jndi-store
```

m. Confirm that the simple text message was received:

```
Message Received: (Bridge Test)
```

n. Press **Enter** to exit the client application.

8. Shut down the bridge.
   a. Kill `imqcmd` along with the OpenMQ server.
   b. Return to the console and **Lock** it.
   c. Edit your **OpenMQBridge** configuration and clear the **Started** check box.
   d. **Save** and **Activate** your changes.
   e. Stop `MedRecSvr1`.

Practice Solution

Perform the following tasks if you did not complete this practice and want to use the finished solution.

Solution Tasks

1. Launch the Lab Framework command shell by executing the `<STUDENT>/bin/prompt.sh` file.
2. Change the current directory to `<LAB>`.
3. Execute the following:

```
ant setup_solution
```

4. The Lab Framework performs the following:
   a. Makes a backup copy of your current work
   b. Runs the solutions for any prerequisite practices, if they have not been run previously
   c. Starts the administration and managed servers if they are not already started
   d. Uses WLST to remove any SAF destinations from prior exercises
   e. Uses WLST to create the MedRec queue
   f. Uses WLST to create bridge destinations
   g. Uses WLST to create a message bridge
   h. Deploys the XA bridge adapter
   i. (Re)Deploys the MedRec application
   j. Copies the OpenMQ client libraries to the domain
5. If you wish to try some of the practice you must first kill and restart the servers.
6. When finished with this practice, perform the section entitled "Shut down the bridge."

Practices for Lesson 13

Chapter 13

Overview of Practices for Lesson 13

Practices Overview

In the practices for this lesson, you configure a domain and its node managers to automatically detect machine failure and restart servers on other available machines.

Practice 13-1: Migrate Failed Servers

Duration: 50 minutes

Objectives

After completing this practice, you should be able to:

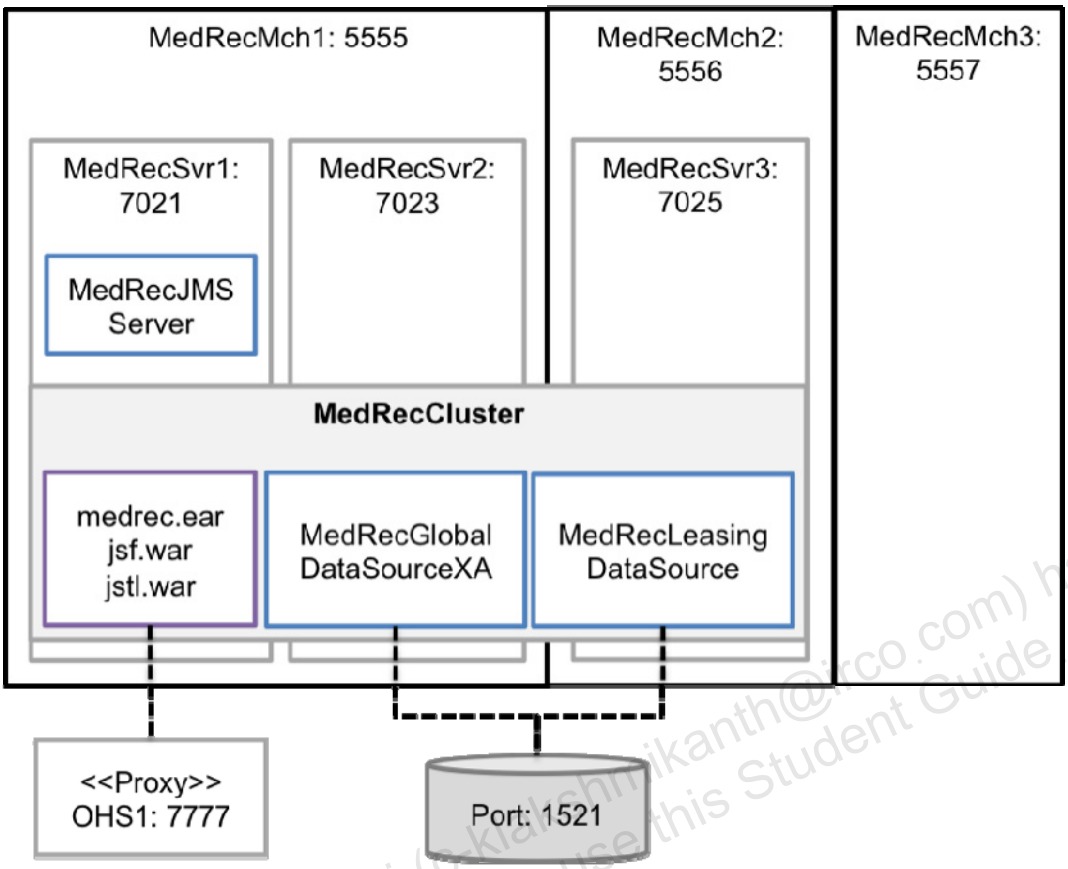
- Define machines and node managers
- Configure cluster database leasing
- Enable automatic server migration
- Verify successful cluster configuration
- Confirm server migration upon machine failure

Overview

WebLogic Server node managers automatically restart failed server instances on the same machine and can optionally kill unhealthy server instances. However, these features are not applicable in cases where the node manager fails or the entire machine fails. Instead, when running in a cluster, you can take advantage of WebLogic Server's whole server migration capability. At run time, one of the cluster members is elected as a master, whose responsibilities include monitoring the health of the other servers. If a failed server is detected and its node manager is also unavailable, the master will instruct another node manager on another machine to restart the server. For high availability purposes, this status or "leasing" information can be maintained in an external database.

In this lab, you will update the current MedRec single-server infrastructure to be a cluster of three servers. You will also define three machines, two of which will be used by default and one that will be available as a backup candidate for whole server migration. To simulate multiple machines in this lab environment, you will run three node managers concurrently, each using a separate port number.

The following is MedRec's initial cluster environment:



Dependencies

The following prior practice must be completed (or equivalent solutions run) before beginning this practice:

- *Create a Custom Domain Template*

Tasks

1. Define servers and machines.
   a. Start the `MedRecAdmSvr`, if it is not already running.
   b. Stop all `MedRecDomain` managed servers, if any are running.
   c. If one is not already open, launch a new Lab Framework command shell by executing the `<STUDENT>/bin/prompt.sh` file.
   d. Navigate to `<LAB>/resources/domain`.
   e. Execute the following WLST scripts:
      - `createMachines.py`
      - `createServer3.py`
   f. Copy the `<LAB>/resources/domain/startServer3.sh` file to `<WORK>/domains/MedRecDomain`.
   g. Launch the administration console.
   h. Select **Environment > Servers**. Confirm the presence of `MedRecSvr3` and that each managed server is assigned a machine:

	Name	Cluster	Machine
	MedRecAdmSvr(admin)		
	MedRecSvr1		MedRecMch1
	MedRecSvr2		MedRecMch1
	MedRecSvr3		MedRecMch2

2. Create a cluster.
   a. **Lock** the console.
   b. Select **Environment > Clusters**. Click **New**.
   c. Enter the following values:

Field	Value
Name	MedRecCluster
Messaging Mode	Unicast

   Click **OK.**
   d. Select the new cluster.
   e. Click the **Configuration > Servers** tab.
   f. Click **Add**.
   g. Select **MedRecSvr1** and click **Finish**.
   h. Repeat the previous steps to add the remaining two managed servers to the cluster.
   i. **Activate** your changes.
3. Initialize the database leasing infrastructure.
   a. Launch a Linux terminal and navigate to `<LAB>/resources/sql`.
   b. Use **sqlplus** to create the `ACTIVE` table (exit **sqlplus** when finished):

```
> sqlplus / as sysdba

SQL> @create_leasing.sql

Table created.

Commit complete.
```

   c. Return to your Lab Framework command shell and execute this WLST script: `<LAB>/resources/jdbc/createLeasingDataSource.py`.
   d. Return to the console. Confirm the presence of the new data source:

	MedRecLeasingDataSource	jdbc/MedRecLeasingDataSource	MedRecCluster

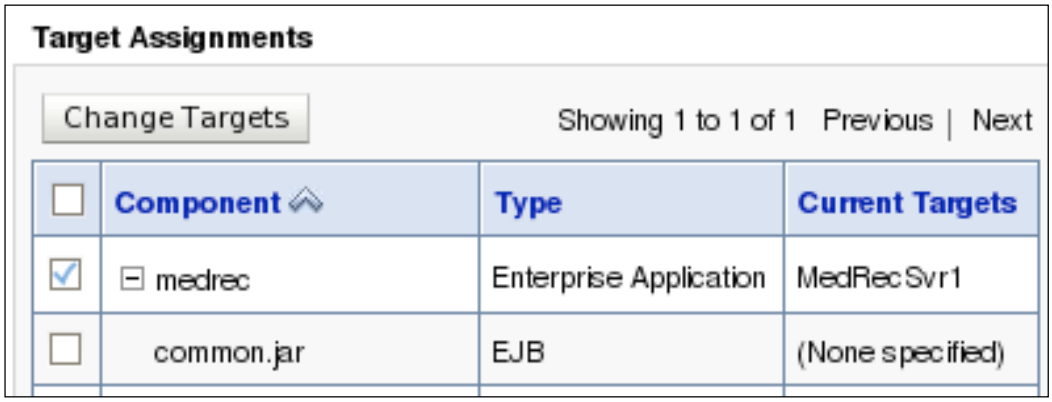
4. Configure servers and cluster for automatic migration.
   a. **Lock** the console.

b. Locate and select **MedRecSvr1**.
c. Click the **Configuration > Migration** tab.
d. Select the **Automatic Server Migration Enabled** check box, and click **Save**.
e. Repeat the previous steps on the remaining two managed servers.
f. Locate and select the **MedRecCluster**.
g. Click the **Configuration > Migration** tab.
h. Locate the **Candidate Machines For Migratable Servers** field. Move **MedRecMch3** from the **Available** column to the **Chosen** column.
i. Complete the remaining fields:

Field	Value
Migration Basis	Database
Data Source For Automatic Migration	MedRecLeasingDataSource
Auto Migration Table Name	WLSDATA.ACTIVE

Click **Save**.

5. Retarget applications and resources.
a. Locate and select the data source named **MedRecGlobalDataSourceXA**.
b. Click the **Targets** tab.
c. Under **Clusters**, select the **All servers in the cluster** button. Click **Save**.
d. Repeat these steps to update the targets for the multi data source named **MedRecMultiDataSource**, if it exists.
e. In the **Domain Structure** panel, select **Deployments**.
f. Click the **jsf** library.
g. Click the **Targets** tab.
h. Select the **All servers in the cluster** button. Click **Save**.
i. Repeat the previous steps to retarget the **jstl** library to the entire cluster.
j. Locate the select the **medrec** application.
k. Click the **Targets** tab.
l. Select the check box for the entire enterprise application:



m. Click **Change Targets**.
n. Once again, select the **All servers in the cluster** button. Click **Yes**.
o. **Activate** all of your changes.

p. Kill and restart the Administration Server, `MedRecAdmSvr`.

6. Start the node managers.
   a. Copy the `<LAB>/solution/files/nodemanager` folder to `<WORK>`.
   b. Browse the contents of `<WORK>/nodemanager`. Inspect the supplied `nodemanager.properties` and `nodemanager.domains` files.
   c. Launch a Linux terminal and navigate to `<WORK>/nodemanager/MedRecMch1`.
   d. Run the script `startNM.sh`.
   e. Verify that the first node manager started successfully:

```
<INFO> <Plain socket listener started on port 5555>
```

   f. Launch two additional terminals and start the remaining two node managers.
7. Start the cluster.
   a. Return to the console.
   b. Locate and select the **MedRecCluster**.
   c. Click the **Control** tab.
   d. Select the check boxes for all cluster members and click **Start**. Then click **Yes**.
   e. Return to the terminals running the node managers. Browse the output. Notice the following:
      – `MedRecSvr1` and `MedRecSvr2` are starting on `MedRecMch1`.
      – `MedRecSvr3` is starting on `MedRecMch2`.
      – No servers are starting on `MedRecMch3`.

```
<INFO> <MedRecDomain> <MedRecSvr1> <Boot identity properties
saved to ...>
...
<INFO> <MedRecDomain> <MedRecSvr1> <Rotated server output log to
...>
```

   f. Return to the console and click the **Monitoring** tab.
   g. Refresh your browser until the **State** of all cluster members is **RUNNING**.
      Tip: In addition to the standard server log files, the node manager redirects the server's standard output stream to a file named `<server>.out`. Both can be useful for troubleshooting.
   h. Use the **Master** column to determine the current lease master. For example:

Name	State	Master
MedRecSvr1	RUNNING	
MedRecSvr2	RUNNING	
MedRecSvr3	RUNNING	True

8. Test the application via the proxy.

a. Launch another Linux terminal.

b. Start your proxy and verify that it is running:

```
> opmnctl start
> opmnctl startproc ias-component=ohs1
> opmnctl status
...
ohs1    | OHS     |    2541 | Alive
```

Why? Clusters always require a proxy sitting in front of them to manage load balancing and failover. We are using Oracle Process Manager and Notification Server (OPMN) to proxy the WebLogic Server cluster. It will run on port `7777`.

c. Launch a new Web browser window and access the MedRec application via the proxy: `http://localhost:7777/medrec/index.action`.

Tip: If the Web browser gives an error, it might be a browser cache issue. Clear the cache in Mozilla Firefox by selecting **Tools > Clear Private Data**. Then select every check box available and click **Clear Private Data Now**. Exit Firefox and launch it. Try the URL again.

Note: If you are interested, you can view the proxy server's configuration at `$INSTANCE_HOME/config/OHS/ohs1`. The configuration file is: `mod_wl_ohs.conf`.

d. Log in as a patient, such as `fred@golf.com/weblogic`, and view records (in the same way as was done in prior exercises).

Tip: The application has been configured to use in-memory session replication when deployed to a cluster.

e. Return to the console.

f. Locate and select the **medrec** application deployment.

g. Click the **Monitoring > Workload** tab.

h. If you would like to determine which server in the cluster the request was routed to, you can use the **Completed Requests** column. You may need to refresh your browser.

9. Simulate a machine failure and verify migration.

a. From a Linux terminal enter the following:

```
> ps –ef | grep MedRecSvr3
```

b. Make a note of the process ID for `MedRecSvr3`. For example:

```
oracle 31578 31539  ... java -jrockit –Xms512m -Xmx512m -
Dweblogic.Name=MedRecSvr3 ...
```

c. Locate the terminals running the node managers for `MedRecMch2` and `MedRecMch3`.

d. Perform the following actions at approximately the same moment:

- Kill the node manager for `MedRecMch2`.
- Kill `MedRecSvr3` (`kill -9 <process_id>`).

Why? MedRecSvr3 is on MedRecMch2, so if the server goes down and the node manager for that machine is unavailable, the server cannot be brought back up by its node manager, so server migration occurs.

e. Locate the terminal running the node manager for `MedRecMch3`.

f. Within a few minutes, this node manager will begin to start `MedRecSvr3`.

g. Return to the console and once again check the **States** of the clustered servers. Confirm that `MedRecSvr3` is restarted successfully.

10. Move a server back to its original machine.
    a. Select the **MedRecCluster** and click the **Control** tab.
    b. Gracefully shut down `MedRecSvr3`. Select this server's check box and click **Shutdown > Force Shutdown Now**. When prompted, click **Yes**.
    c. Return to the terminal running the node manager for `MedRecMch3`. Verify that `MedRecSvr3` was shut down (or use the console):

```
<Info> <MedRecDomain> <MedRecSvr3> <Server was shut down
normally>
```

    d. Restart the node manager for `MedRecMch2`.
    e. Return to the console and start `MedRecSvr3`.
    f. Confirm that the server was restarted on `MedRecMch2`.
    g. When finished, stop all of your managed servers by using the administration console and kill all node managers. Note that unless explicitly indicated in subsequent practices, you will not use the node managers to start and stop servers; instead you will use the start scripts.

Practice Solution

Perform the following tasks if you did not complete this practice and want to use the finished solution.

Solution Tasks

1. Stop all managed servers, if any are running.
2. Launch the Lab Framework command shell by executing the `<STUDENT>/bin/prompt.sh` file.
3. Change the current directory to `<LAB>`.
4. Execute the following:

```
ant setup_solution
```

5. The Lab Framework performs the following:
   a. Makes a backup copy of your current work
   b. Runs the solutions for any prerequisite practices, if they have not been run previously
   c. Starts the database and (re)creates the cluster leasing table
   d. Starts the proxy server if it is not already started
   e. Copies a server start script file to the domain
   f. Copies node manager scripts and supporting files to your work area
   g. Starts the administration server if it is not already started
   h. Uses WLST to create machine definitions
   i. Uses WLST to create `MedRecSvr3`
   j. Uses WLST to create a cluster
   k. Uses WLST to retarget resources to the cluster
   l. Uses WLST to create the leasing data source and associate it with the cluster
   m. Uses WLST to enable auto migration on each server
6. Start the three node managers.
7. Use the console to start the three managed servers.
8. When finished working on this practice, stop all of your managed servers by using the administration console and kill all node managers. Note that unless explicitly indicated in subsequent practices, you will not use the node managers to start and stop servers; instead you will use the start scripts.

Practices for Lesson 14

Chapter 14

Overview of Practices for Lesson 14

Practices Overview

In the practices for this lesson, you configure JMS for a cluster, including service migration and distributed destinations.

Practice 14-1: Configure JMS High Availability

Duration: 40 minutes

Objectives

After completing this practice, you should be able to:

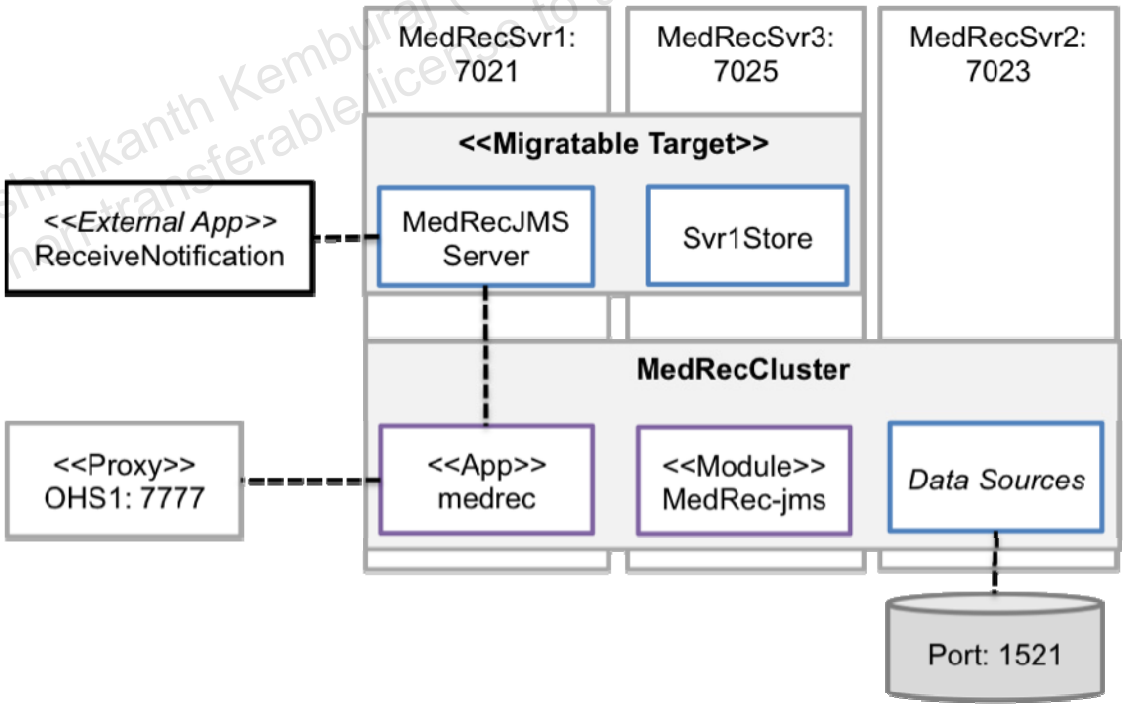
- Configure automatic JMS service migration in a cluster
- Define and use migratable targets
- Configure a connection factory for high availability
- Verify producer and consumer failover after JMS server failure

Overview

JMS servers cannot be deployed to an entire cluster. Instead, a JMS server is pinned to a specific server within the cluster. However, WebLogic Server can still provide transparent failover for JMS applications through either server-level migration or service-level migration. When a JMS server or its host server fails, service-level migration allows it to automatically restart on another available server in the cluster. The candidate servers to which a JMS server can be migrated may also be constrained to a subset of servers in the cluster.

In this practice, you will configure automatic service-level migration for the JMS server hosted on `MedRecSvr1`, along with its corresponding persistent store. You will then confirm that the MedRec producer and consumer applications continue to function correctly despite the loss of `MedRecSvr1`.

The following is the upgraded MedRec cluster, including its JMS migration policies:



Dependencies

No prior practices need to be completed before starting this practice.

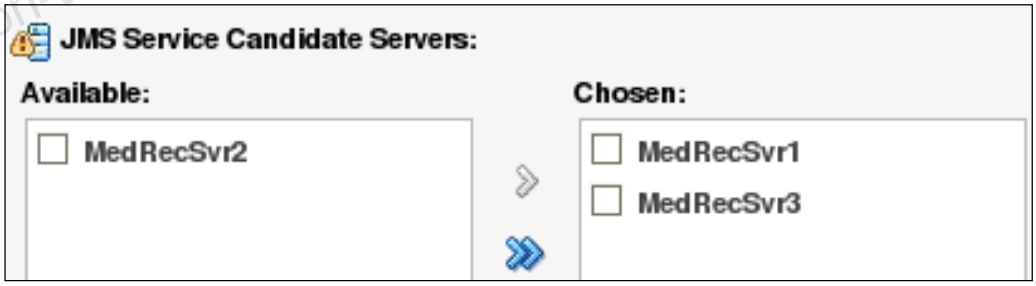
Tasks

1. Initialize a new clustered domain.
   a. Launch the Lab Framework command shell by executing the `<STUDENT>/bin/prompt.sh` file.
   b. Change the current directory to `<LAB>` and execute the following:

```
ant setup_exercise
```

   c. The Lab Framework performs the following:
      - Makes a backup copy of your current work.
      - Shuts down any running servers.
      - Re-creates `MedRecDomain`.
      - Starts the administration server and three managed servers.
      - Starts the OPMN proxy server if it is not already running.
      - Uses WLST to create data sources.
      - Uses WLST to create a JDBC persistent store.
      - Uses WLST to create a JMS server and module.
      - Uses WLST to configure database leasing for the cluster.
      - Deploys the MedRec application along with supporting libraries.

      Tip: If you need to restart a server at a later time, remember that the domain includes server start scripts for convenience.
2. Configure automatic JMS service migration.
   a. Launch the console and **Lock** it.
   b. Locate and select **MedRecSvr1**.
   c. Click the **Configuration > Migration** tab.
   d. Locate the **JMS Service Candidate Servers** field.
   e. Move **MedRecSvr1** and **MedRecSvr3** from the **Available** column to the **Chosen** column:



   f. Repeat the previous step for the **JTA Candidate Servers** field.
   g. Click **Save**.
   h. In the **Domain Structure** panel, select **Environment > Migratable Targets**.
   i. Select **MedRecSvr1 (migratable)**. This is the default generated migration policy for MedRecSvr1.
   j. Click the **Configuration > Migration** tab.
   k. Set the value of **Service Migration Policy** to **Auto-Migrate Failure Recovery Services**.
   l. Click **Save**. Then **Activate** your changes.

3. Target persistent stores and JMS servers to a migration policy.
   a. **Lock** the console once again.
   b. Locate and select the **Svr1Store** persistent store.
   c. Set the **Target** field to **MedRecSvr1 (migratable)** and click **Save**.
   d. Locate and select the **MedRecJMSServer** JMS server.
   e. Set the **Persistent Store** to `Svr1Store` and click **Save**.
   f. Click the **Targets** tab.
   g. Select the target **MedRecSvr1 (migratable)** and click **Save**.
   h. **Activate** your changes.
4. Create a connection factory for producer/consumer high availability.
   a. **Lock** the console once again.
   b. Locate and select the **MedRec-jms** JMS module.
   c. Click **New**.
   d. Select the **Connection Factory** option and click **Next**.
   e. Enter the following values:

Field	Value
Name	HAConnectionFactory
JNDI Name	com.bea.medrec.jms.HAConnectionFactory

   Click **Next**.
   f. Note that by default it is targeted to the entire cluster, so that JMS clients can initially connect to any server. Click **Finish**.
   g. Click the new connection factory to edit it.
   h. Click the **Configuration > Client** tab.
   i. Set the value of **Reconnect Policy** to **All**, and click **Save**.
   j. **Activate** your changes.
   k. Restart all of your managed servers.

   Tip: Remember, an easy way to kill a server is to go to the window in which it is running and click **Ctrl + c**. To then restart the managed servers recall that, for your convenience, the `startServer`*X*`.sh` scripts are found in your domain folder.
5. Post a message from a clustered application.
   a. Access the MedRec application via the proxy: `http://localhost:7777/medrec/index.action`.
   b. Return to the console.
   c. Locate and select the **medrec** application deployment.
   d. Click the **Monitoring > Workload** tab.
   e. Use the **Completed Requests** column to help determine which server the request was routed to. You may need to refresh your browser.
   f. From a Linux terminal execute the script `<LAB>/resources/updateStatus.sh`.
   g. Launch the MedRec application.
   h. In the **Administrator** section of the home page, click **Login**.
   i. Log in using **admin@avitek.com** as the username and **weblogic** as the password.
   j. Click the **View Pending Requests** link.

k. Click the registration request for the new patient, **charlie@star.com**. Then click the **Approve** button.

l. Leave the MedRec browser session open.

m. Return to the console.

n. In the **Domain Structure** panel, select **Services > Messaging > JMS Servers**.

o. Note that the **Current Server** column of `MedRecJMSServer` is set to MedRecSvr1:

	Name	Persistent Store	Target	Current Server	Health
☐	MedRecJMSServer	Svr1Store	MedRecSvr1 (migratable)	MedRecSvr1	OK

p. Select the JMS server.

q. Click the **Monitoring > Active Destinations** tab.

r. Use the **Messages Current** column to confirm that a message was added to the **PatientNotificationQueue**. The message was routed to `MedRecSvr1`, regardless of which server the producing application was running on.

6. Run the external consumer application.

a. Open your Lab Framework command shell.

b. Navigate to `<LAB>/resources/client` and execute the client application. Direct it to all cluster members to look up and establish the initial connection:

```
java com.bea.medrec.ReceiveNotification
t3://localhost:7021,localhost:7023,localhost:7025
```

c. Confirm that the message was received by the client application:

```
Message Received: (Charlie,charlie@star.com,APPROVED)
```

d. The application continues to check for messages. Leave it running.

7. Test JMS migration.

a. Return to the console and **Lock** it.

Tip: Although you will not modify your domain's configuration at this point, a domain lock is still required to perform a manual migration.

b. In the **Domain Structure** panel, select **Environment > Migratable Targets**.

c. Click the **Control** tab.

d. Select the check box for **MedRecSvr1 (migratable)** and click **Migrate**.

e. For the **New Hosting Server**, select **MedRecSvr3**. Click **OK**.

f. In the **Domain Structure** panel, select **Services > Messaging > JMS Servers**.

g. Use the **Current Server** column to confirm that `MedRecJMSServer` has been migrated to `MedRecSvr3`.

h. Return to the client application and confirm that it is still running without failure.

i. Similarly, locate and monitor the **PatientNotificationQueue** in the **MedRec-jms** module. The **Consumers Current** column should continue to be 1.

j. **Undo** the console lock.

8. Verify the high availability of JMS producers and consumers.

a. Return to the browser hosting your MedRec session. Repeat the prior steps to post another patient notification.

   Tip: The previous steps to create a patient notification are 5.f through 5.k.

   Note: When performing those steps, it may take the Web browser a long time to load (up to several minutes). This is because it takes time for the JMS migration to actually occur. Be patient and let the application run until the next page appears.

b. Return to the client application and confirm that it was successfully redirected to `MedRecSvr3` and received the next message.

c. When finished, press **Enter** to quit the client application.

d. From the console, return to the migratable target **MedRecSvr1 (migratable)**. Repeat the previous steps to manually migrate services back to `MedRecSvr1`.

   Tip: The previous steps for migration are 7.a through 7.g, except this time you select **MedRecSvr1** in step e and look for `MedRecSvr1` in step g.

Practice Solution

Perform the following tasks if you did not complete this practice and want to use the finished solution.

Solution Tasks

1. If not done previously, perform the section of the instructions entitled “Initialize a new clustered domain.”
2. Shut down all managed servers.
3. From your Lab Framework shell, change the current directory to `<LAB>`.
4. Execute the following:

```
ant setup_solution
```

5. The Lab Framework performs the following:
   a. Uses WLST to assign the persistent store to the JMS server
   b. Uses WLST to configure automatic JMS migration for `MedRecSvr1`
   c. Uses WLST to target JMS resources to the `MedRecSvr1` migration policy
   d. Uses WLST to add a new connection factory to the JMS module
6. Start the managed servers.

Practice 14-2: Load Balance JMS Messages

Duration: 40 minutes

Objectives

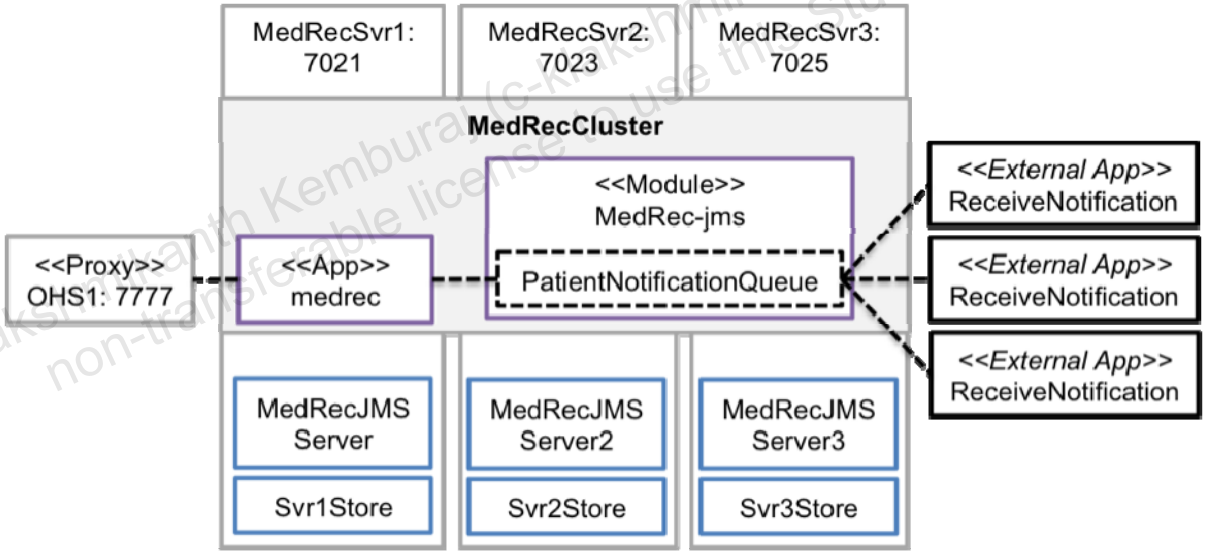
After completing this practice, you should be able to:

- Create a uniform distributed queue
- Monitor distributed queue members
- Verify server affinity for message producers
- Verify load balancing for message consumers

Overview

Currently, the MedRec infrastructure uses a single JMS server. However, load testing reveals that this single server may not be able to meet future throughput requirements. Therefore, to provide scalability, you will upgrade the current standard JMS queue to a uniform distributed queue. As remote applications produce and consume messages from this queue, their work will be load balanced across the JMS servers found on all cluster members. By default, local applications will demonstrate an affinity for their local queue instance to avoid unnecessary network communication.

The following shows MedRec's use of distributed destinations:



Dependencies

The following prior practice must be completed (or equivalent solutions run) before beginning this practice:

- *Configure JMS High Availability*

Tasks

1. Create persistent stores and JMS servers on each cluster member.
   a. Start all servers in your `MedRecDomain`, if not already started.
   b. From your Lab Framework command shell, navigate to `<LAB>/resources/jms`.
   c. Execute the following WLST scripts:

- `createStores.py`
- `createJMSServers.py`

d. Launch the administration console.

e. In the **Domain Structure** panel, select **Services > Messaging > JMS Servers**.

f. Confirm that a JMS server is deployed on each managed server:

	Name	Persistent Store	Target
☐	MedRecJMSServer	Svr1Store	MedRecSvr1 (migratable)
☐	MedRecJMSServer2	Svr2Store	MedRecSvr2 (migratable)
☐	MedRecJMSServer3	Svr3Store	MedRecSvr3 (migratable)

g. If you have another JMS server named **NotificationsJMSServer** from a previous exercise, edit it and set its **Targets** to `none`.

2. Create a distributed queue.

a. **Lock** the console.

b. Locate and select the JMS module named **MedRec-jms**.

c. Delete the existing queue named **PatientNotificationQueue**. Then **Activate** your changes.

d. **Lock** the console once again.

e. Click **New**.

f. Select the **Distributed Queue** option and click **Next**.

g. Enter the following values:

Field	Value
Name	PatientNotificationQueue
JNDI Name	com.bea.medrec.jms.PatientNotificationQueue
Load Balancing Policy	Round-Robin
Allocate Members Uniformly	<checked>

Click **Next**.

h. Click **Advanced Targeting**.

i. Click **Create a New Subdeployment**.

j. Enter `DeployToCluster` as the **Subdeployment Name** and click **OK**.

k. Select the check box for **MedRecCluster** and click **Finish**.

l. **Activate** your changes.

m. Select the new **PatientNotificationQueue**.

n. Click the **Configuration > Members** tab.

o. Confirm that the distributed queue is deployed to a JMS server on each cluster member:

Name
MedRecJMSServer2@PatientNotificationQueue
MedRecJMSServer3@PatientNotificationQueue
MedRecJMSServer@PatientNotificationQueue

3. Post messages using the distributed queue and verify server affinity.
   a. Access the MedRec application via the proxy: `http://localhost:7777/medrec/index.action`.
   b. Return to the console.
   c. Locate and select the **medrec** application deployment.
   d. Click the **Monitoring > Workload** tab.
   e. Use the **Completed Requests** column to help determine which server the request was routed to.
   f. Generate a patient notification message using the MedRec application.

      Tip: From a Linux terminal execute the script `<LAB>/resources/updateStatus.sh`. Launch the MedRec application. In the **Administrator** section of the home page, click **Login**. Log in using **admin@avitek.com** as the username and **weblogic** as the password. Click the **View Pending Requests** link. Click the registration request for the new patient **charlie@star.com**. Then click the **Approve** button.
   g. Return to the console.
   h. Locate and select the **PatientNotificationQueue**.
   i. Click the **Monitoring** tab.
   j. Click **Customize this table**. Move the **Messages Current** column from **Available** to **Chosen**. Click **Apply**.
   k. Verify that a message has been posted to the queue that resides on the same server that received the application request. For example:

☐	MedRec-jms!MedRecJMSServer@PatientNotificationQueue	1	0

   l. Repeat the previous steps until you generate another message on a different server in the cluster.

      Tip: The previous steps to generate a message can be found in the tip on step 3.f.

      Important Note: If you find that all the messages go to one server in the cluster, then the proxy is not load balancing. To force a message to go to another server, use the MedRec application directly on that managed server, bypassing the proxy. Do this by changing the port in the URL from `7777` to one of the managed server ports: `7021`, `7023`, or `7025`. For example, to call MedRec directly on `MedRecSvr3` use: `http://localhost:7025/medrec/index.action`.
4. Consume messages from a distributed queue.
   a. Launch four new Lab Framework command shells.

b. In the first shell, navigate to `<LAB>/resources/client` and execute the client application. Direct it to all cluster members to look up and establish the initial connection:

```
java com.bea.medrec.ReceiveNotification
t3://localhost:7021,localhost:7023,localhost:7025
```

c. Return to the console and monitor the **PatientNotificationQueue**. Use the **Consumers Current** column to determine which server the consumer was load balanced to.

d. Repeat the previous steps to run three additional client applications.

Tip: Perform step 4.b. in the other three Lab Framework shells, so they are also running the client application.

e. Verify that all produced messages are received:

```
Message Received: (Charlie,charlie@star.com,APPROVED)
```

Note: Which client receives the message does not matter. Remember, with a queue all that matters is that exactly one of the clients receives the message.

f. When finished, press **Enter** to quit each client application.

Practice Solution

Perform the following tasks if you did not complete this practice and want to use the finished solution.

Solution Tasks

1. If not done so previously, perform the section of the instructions entitled “Initialize a new clustered domain” in the “Configure JMS High Availability” practice.
2. Stop all managed servers, if any are running. Be sure that your administration server is still running.
3. From your Lab Framework shell, change the current directory to `<LAB>`.
4. Execute the following:

```
ant setup_solution
```

5. The Lab Framework performs the following:
   a. Makes a backup copy of your current work
   b. Runs the solutions for any prerequisite practices, if they have not been run previously
   c. Uses WLST to create persistent stores for `MedRecSvr2` and `MedRecSvr3`
   d. Uses WLST to create JMS servers for `MedRecSvr2` and `MedRecSvr3`
   e. Uses WLST to delete a standard queue
   f. Uses WLST to create a distributed queue
6. Start the managed servers.

Practices for Lesson 15

Chapter 15

Overview of Practices for Lesson 15

Practices Overview

In the practices for this lesson, you work with some of WebLogic's advanced HTTP session replication capabilities.

Naming Conventions

The practices in this lesson use the following variable names to refer to commonly used locations on your file system:

Variable	Path
<WT_HOME>	/u01/app/oracle/Middleware/11.1.1/webtier
<INSTANCE_HOME>	/u01/app/oracle/instances/webtier
<DIST_HOME>	/u01/app/opensource/Distributor

Practice 15-1: Replicate Sessions Across Two Clusters

Duration: 45 minutes

Objectives

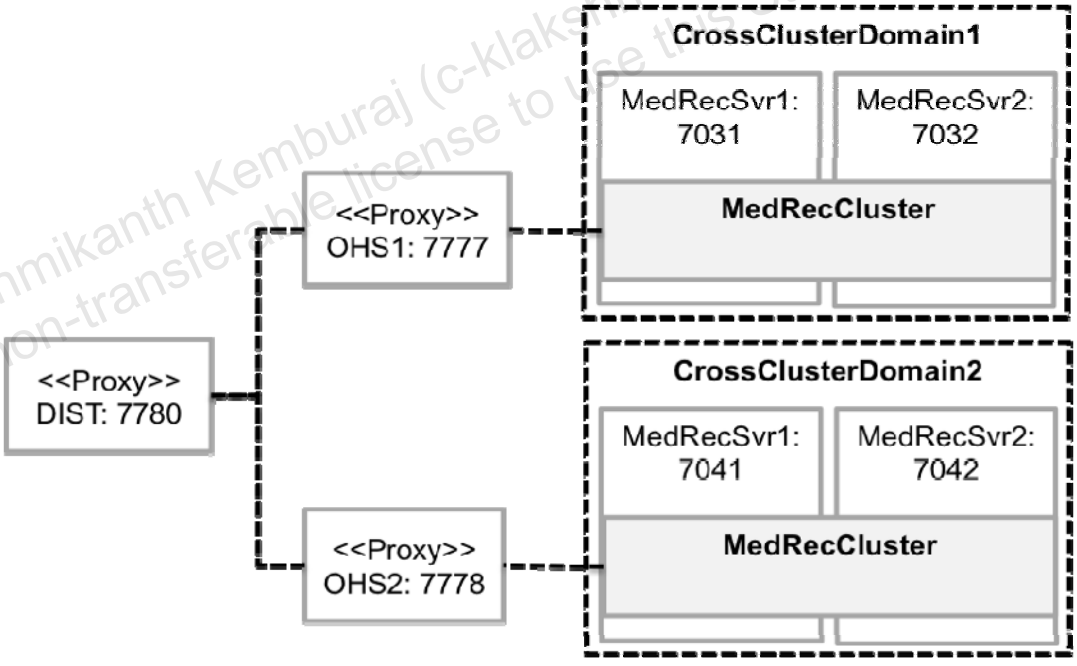
After completing this practice, you should be able to:

- Create two new domains, each with a cluster of two servers
- Start additional load balancers
- Configure clusters for MAN session replication
- Configure trust for cross-domain communication
- Verify failover after primary cluster failure

Overview

On MedRec's main corporate campus there are several available data centers, each on a separate subnet. To date, applications have been hosted using a cluster of WebLogic Servers within a single data center. For mission critical applications, however, IT would like to distribute the load across multiple data centers to provide better scalability and availability in the event that an entire data center fails. Due to the low latency between these subnets, IT has opted to employ WebLogic's Metropolitan Area Network (MAN) session replication feature.

The following is the MedRec MAN cluster architecture:



Dependencies

No prior practices need to be completed before starting this practice.

Tasks

1. Create two new domains.
   a. Stop any running `MedRecDomain` servers.
   b. Within the Lab Framework command shell, navigate to `<LAB>/resources/domains`.

c. Execute the following WLST scripts:
   - createDomain1.py
   - createDomain2.py

d. Confirm the presence of two new domains at <WORK>/domains:
   - CrossClusterDomain1
   - CrossClusterDomain2

   Tip: Both of these domains are configured with an administrative account weblogic/Welcome1.

2. Configure local and global load balancers.

a. Make a backup copy of the following files:
   - <INSTANCE_HOME>/config/OHS/ohs1/mod_wl_ohs.conf
   - <INSTANCE_HOME>/config/OHS/ohs2/mod_wl_ohs.conf

   Tip: For convenience, INSTANCE_HOME is also an OS environment variable.

b. Copy the following file to <INSTANCE_HOME>/config/OHS/ohs1:
   <LAB>/resources/proxy/ohs1/mod_wl_ohs.conf

   Overwrite the existing file. This file configures the ohs1 Web server so that requests are proxied to the cluster defined in CrossClusterDomain1.

c. Copy the following file to <INSTANCE_HOME>/config/OHS/ohs2:
   <LAB>/resources/proxy/ohs2/mod_wl_ohs.conf

   Overwrite the existing file. This file configures the ohs2 Web server so that requests are proxied to the cluster defined in CrossClusterDomain2.

d. Launch a Linux terminal.

e. Restart both proxy servers:

```
> opmnctl stopproc
> opmnctl startproc
```

f. Verify that both proxy servers are running:

```
> opmnctl status
...
ohs2    | OHS     |    8265 | Alive
ohs1    | OHS     |    8264 | Alive
```

g. Navigate to <DIST_HOME>.

   Tip: For convenience, DIST_HOME is also an OS environment variable.

h. Start the global proxy server:

```
> ./distributor.init start
```

3. Configure cross-domain trust on domain 1.

a. Start the administration server for CrossClusterDomain1.

   Tip: Open a terminal window. Navigate to <WORK>/domains/CrossClusterDomain1. Run ./startWebLogic.sh. When asked for a username and password enter weblogic/Welcome1.

b. Access the console for this domain: http://localhost:7030/console.

c. **Lock** the console.

d. In the **Domain Structure** panel, select the domain **CrossClusterDomain1**.
e. Click the **Security** tab.
f. Select the **Cross Domain Security Enabled** check box and click **Save**.
g. **Activate** your changes.
h. In the **Domain Structure** panel, click **Security Realms**.
i. Click **myrealm**.
j. Click the **Users and Groups** tab.
k. Click **New**. Create a new user, `crossdomain/Welcome1`.
l. Select the new user.
m. Click the **Groups** tab. Add the user to the group **CrossDomainConnectors** and click **Save**.
n. Return to the **myrealm** security realm.
o. Click the **Credential Mappings** tab.
p. Click **New**. Enter the following values:

Field	Value
Use Cross-Domain Protocol	<checked>
Remote Domain	CrossClusterDomain2

Click **Next**.

q. Enter the following values:

Field	Value
Remote User	crossdomain
Remote Password, Confirm Password	Welcome1

Click **Finish**.

4. Configure MAN session replication for cluster 1.
   a. Return to the console and lock it once again.
   b. Locate and select the **MedRecCluster**.
   c. Click the **Configuration > Replication** tab.
   d. Edit the following fields:

Field	Value
Cross-cluster Replication Type	MAN (Synchronous) HTTP Session State Replication
Remote Cluster Address	t3://localhost:7041,localhost:7042

Click **Save**.

   e. **Activate** your changes.
5. Configure trust and replication on domain 2.
   a. Start the administration server for `CrossClusterDomain2`.
   b. Access the console for this domain: `http://localhost:7040/console`.
   c. Repeat the steps in the previous section, “Configure cross-domain trust on domain 1,” but this time on `CrossClusterDomain2`.

      Be sure to set the **Remote Domain** field to `CrossClusterDomain1`.

d. Repeat the previous steps to configure replication on this domain's **MedRecCluster**:

Field	Value
Cross-cluster Replication Type	MAN (Synchronous) HTTP Session State Replication
Remote Cluster Address	t3://localhost:7031,localhost:7032

e. **Activate** your changes.

6. Deploy a test application to both clusters.
   a. Start the two managed servers for `CrossClusterDomain1`.

      Tip: For convenience, the domain contains the `startServer1.sh` and `startServer2.sh` scripts.
   b. Edit the `startServer1.sh` and `startServer2.sh` scripts for `CrossClusterDomain2`. Change the administration server port to 7040.
   c. Start the two managed servers for `CrossClusterDomain2`.

      Tip: Notice that the server log messages include the text "Starting man-async replication service with remote cluster address…"
   d. Copy the following file to `<WORK>/applications`:

      `<LAB>/resources/apps/ShoppingCart.war`
   e. Return to the Lab Framework command shell.
   f. Execute the following WLST scripts located in `<LAB>/resources/apps`:
      - `deployDomain1.py`
      - `deployDomain2.py`
7. Test the application using cluster 1.
   a. Close all open Web browsers and then launch a new one.
   b. Access the application via the global proxy server: `http://localhost:7780/ShoppingCart`.
   c. Click **Go Shopping** and **Add** an item to your shopping cart. Do not close the browser.
   d. Inspect the standard output from your servers in both `CrossClusterDomain1` and `CrossClusterDomain2` to determine which one is hosting this user's session. The server hosting the session will have messages being printed out. Here are some examples:

```
within welcome.jsp
within viewshoppingcart.jsp
within shoppingcart.jsp
```

8. Verify failover to cluster 2 if the primary cluster is unavailable.
   a. Stop the local proxy server for `CrossClusterDomain1`:

```
> opmnctl stopproc ias-component=ohs1
```

   b. Return to the browser running the test application.
   c. Click **Back to Home Page**.

      Important: You may need to click the link more than once for the Distributor to fail over.
   d. Click **View Shopping Cart**. The contents of the cart should remain intact.

e. Once again, inspect the standard output for your servers to determine the new host in `CrossClusterDomain2`.

f. When you have finished the practice successfully, kill the shell running the global proxy, `distributor.init`.

g. Shut down the remaining local proxy server and management process:

```
> opmnctl stopall
```

h. Kill all servers in `CrossClusterDomain1` and `CrossClusterDomain2`.

Practice Solution

No solution exists for this practice. You must complete all of the instructions.

Practices for Lesson 16

Chapter 16

Overview of Practices for Lesson 16

Practices Overview

In the practices for this lesson, you customize the WebLogic security realm to support authentication from an external LDAP server and an external database schema.

Naming Conventions

The practices in this lesson use the following variable names to refer to commonly used locations on your file system:

Variable	Path
<LDAP_HOME>	/u01/app/opensource/OpenDS

Practice 16-1: Authenticate Using an External LDAP

Duration: 40 minutes

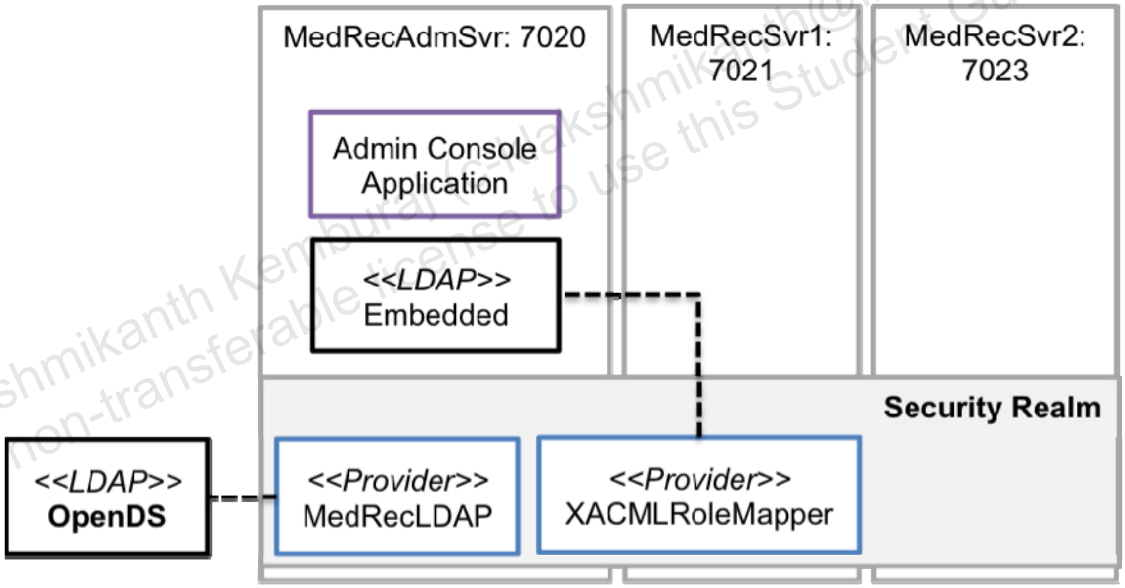
Objectives

After completing this practice, you should be able to:

- Configure an external LDAP with default WebLogic groups
- Add an LDAP authentication provider to a security realm
- Use authentication provider control flags

Overview

MedRec's WebLogic Server infrastructure must integrate with their corporate LDAP, which identifies standard administrative users and groups. WebLogic Server includes a Default Authentication provider that connects to an LDAP server embedded within the Administration Server for a domain. In this exercise, you will replace this default provider with an LDAP provider that connects to an external repository. You will then validate this configuration by logging in to the administration console as various LDAP users. For the purposes of this exercise, you will connect to a simple, open source LDAP server, as follows:



Dependencies

No prior practices need to be completed before starting this practice.

Instructions

1. Re-create the MedRec domain.
   a. Launch the Lab Framework command shell by executing the `<STUDENT>/bin/prompt.sh` file.
   b. Change the current directory to <LAB> and execute the following:

```
ant setup_exercise
```

   c. The Lab Framework performs the following:
      – Makes a backup copy of your current work

- Shuts down any running servers
- Re-creates `MedRecDomain`
- Starts the administration server
- Uses WLST to create a data source
- Uses WLST to create a JMS server and module
- Deploys the MedRec application along with supporting libraries

Tip: If you need to restart a server at a later time, remember that the domain includes server start scripts for convenience.

Why recreate the domain? This setup recreates the domain to be non-clustered to reduce the complexity of practice instructions.

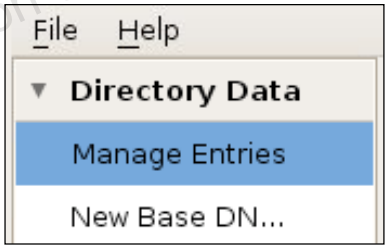
2. Initialize LDAP users.
   a. Launch a new Linux terminal.
   b. Navigate to `<LDAP_HOME>/bin`.

      Tip: For convenience, `LDAP_HOME` is also an OS environment variable.
   c. Launch the OpenDS control panel:

```
> ./control-panel
```

   d. A dialog box appears warning that the Local Server is not running. Click **OK** to close it.
   e. Click the **Start** button:

   f. When prompted, enter the password, `Welcome1`, and click **OK**.
   g. After the LDAP server has successfully started, close the dialog box.
   h. In the left menu, select **Directory Data > Manage Entries**:



   i. Click the base node, **dc=medrec,dc=com**. Then right-click and select **New User**:

   j. Enter the following values:

Field	Value
First Name	WLS

Last Name	System
Common Name	WLS System Admin
User ID	weblogic
Password, Password (Confirm)	Welcome1
Naming Attribute	uid

Click **OK**.

k. After the user has been successfully created, close the dialog box.

l. Repeat the previous steps to create a second user:

Field	Value
First Name	Oracle
Last Name	System
Common Name	Oracle System Admin
User ID	OracleSystemUser
Password, Password (Confirm)	Welcome1
Naming Attribute	uid

m. Create a third user:

Field	Value
First Name	Tom
Last Name	Hardum
Common Name	Tom Hardum
User ID	thardum
Password, Password (Confirm)	Welcome1
Naming Attribute	uid

n. Create a fourth user:

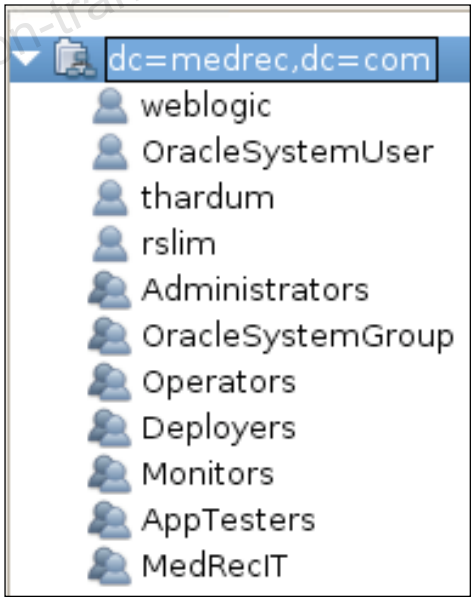
Field	Value
First Name	Rebecca
Last Name	Slim
Common Name	Rebecca Slim
User ID	rslim
Password, Password (Confirm)	Welcome1
Naming Attribute	uid

o. Confirm that you have four new users within the base DN:

3. Initialize LDAP groups.
   a. Right-click the base node, **dc=medrec,dc=com**, and select **New Group**.
   b. For **Name**, enter `Administrators`.
   c. Click the **Add Members** button.
   d. Select the **weblogic** user and click **OK**.
   e. Click **OK** once again.
   f. After the group has been successfully created, close the dialog box.
   g. Repeat the previous steps to create the following additional groups and initialize their members:

Group Name	Members
OracleSystemGroup	OracleSystemUser
Operators	thardum
Deployers	thardum
Monitors	thardum
AppTesters	thardum
MedRecIT	rslim

   h. Your final directory should resemble the following:



   i. Click **Close** to exit the **Manage Entries** dialog box.
   j. Exit the control panel.

Tip: Select **Exit** from the **File** menu.

4. Add an LDAP Authentication provider to the security realm.
   a. Launch the administration console: `http://localhost:7020/console`.
   b. **Lock** the console.
   c. In the **Domain Structure** panel, select **Security Realms**.
   d. Click **myrealm**.
   e. Click the **Providers > Authentication** tab. Then click **New**.
   f. Enter the following values:

Field	Value
Name	MedRecLDAP
Type	LDAPAuthenticator

   Click **OK**.
   g. Click the new provider.
   h. Click the **Configuration > Provider Specific** tab.
   i. Enter the following values:

Field	Value
Host	localhost
Port	7878
Principal	cn=Directory Manager
Credential, Confirm Credential	Welcome1
User Base DN	dc=medrec,dc=com
User Name Attribute	uid
Group Base DN	dc=medrec,dc=com
Static Group Name Attribute	cn

   Click **Save**.
5. Adjust the authentication processing order.
   a. Use the locator link trail to return back to the list of authentication providers:

   b. Click **Reorder**.
   c. Use the arrow  buttons to put the providers in the following order:
      - **MedRecLDAP**
      - **DefaultAuthenticator**
      - **DefaultIdentityAsserter**

      Click **OK**.
   d. Click the **MedRecLDAP** provider. Confirm that the **Configuration > Common** tab is selected.
   e. Set the **Control Flag** to **Sufficient**. Click **Save**.

f. Similarly, edit the **DefaultAuthenticator** and set its **Control Flag** to **Sufficient** as well.
Why? Running both providers concurrently allows you to continue administering the domain despite any potential problems with the new security provider.
g. **Activate** your console changes.
h. Kill and restart your administration server.

6. Test the LDAP provider using the console.
a. Return to the console.
b. When prompted to log in, enter `thardum/Welcome1`. Confirm that the user was authenticated successfully.
c. Log out of the console, and log back in as the `weblogic` user.
d. Locate and select **myrealm** again.
e. Click the **Users and Groups > Users** tab. Notice that users in both providers are shown, but only those in the `DefaultAuthenticator` are editable in the console:

	Name	Description	Provider
	OracleSystemUser		MedRecLDAP
☐	OracleSystemUser	Oracle application software system user.	DefaultAuthenticator
	rslim		MedRecLDAP

f. Click the **Users and Groups > Groups** tab. Verify the presence of the `MedRecIT` group in the `MedRecLDAP` provider.

7. Add an external group to a global role.
a. Click the **Roles and Policies > Realm Roles** tab.
b. In the **Roles** table, locate **Global Roles > Roles > Deployer**.
c. Click the **View Role Conditions** link:

Deployer	Global Role	View Role Conditions

d. Click **Add Conditions**.
e. Verify that the **Predicate List** is set to **Group**, and click **Next**.
f. For **Group Argument Name**, enter `MedRecIT` and click **Add**.
g. Click **Finish**. Then click **Save**.
h. Log out of the console, and log back in as `rslim/Welcome1`.
i. After successfully logging in, explore the console. Rebecca should be able to manage **Deployments**, but not create other types of resource such as servers and data sources.

8. Remove the default provider.
a. Log out of the console, and log back in as `weblogic`.
b. **Lock** the console.
c. Locate and select **myrealm** again.
d. Locate and delete the `DefaultAuthenticator`.

e. **Activate** your changes.
f. Restart the administration server.

Practice Solution

Perform the following tasks if you did not complete this practice and want to use the finished solution.

Solution Tasks

1. If not done previously, perform the section of the instructions entitled “Re-create the MedRec domain.”
2. Launch the Lab Framework command shell by executing the `<STUDENT>/bin/prompt.sh` file.
3. Change the current directory to `<LAB>`.
4. Execute the following:

```
ant setup_solution
```

5. The Lab Framework performs the following:
   a. Starts the LDAP server and imports solution data
   b. Uses WLST to create the LDAP authentication provider
   c. Uses WLST to delete the default authentication provider
   d. Uses WLST to update global roles
6. Restart the administration server.

Practice 16-2: Authenticate Using a Database

Duration: 30 minutes

Objectives

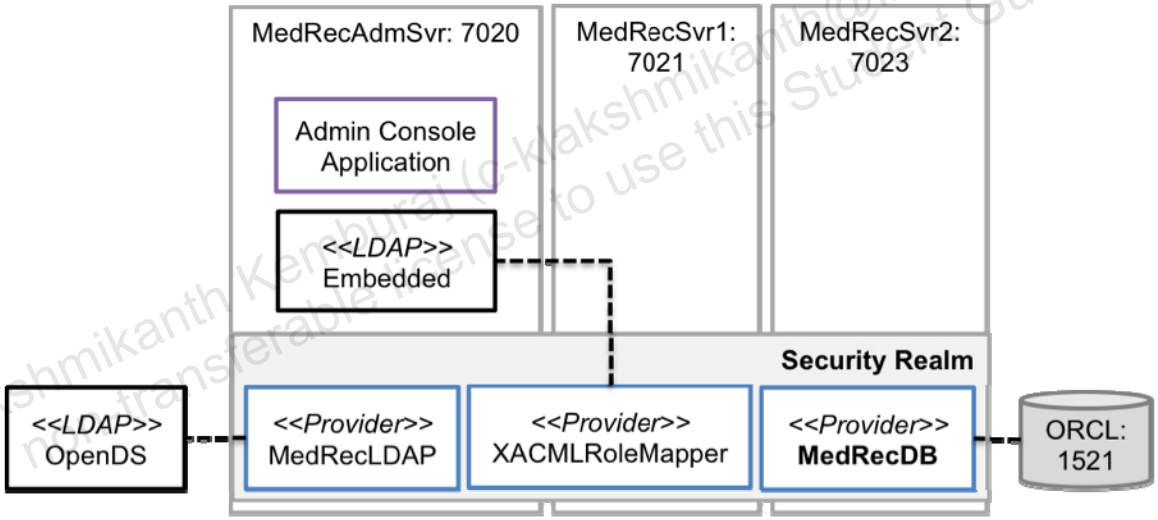
After completing this practice, you should be able to:

- Configure a database to support the SQL authenticator
- Add a SQL authenticator to a security realm
- Use the console to add users and groups to the database

Overview

MedRec's IT infrastructure includes an existing identity management database with tables that define IT users and groups. Fortunately, WebLogic Server supports multiple security providers, so IT can use our existing LDAP users as well as these database users. In this practice, you will use the WebLogic SQL Authentication provider to integrate your WebLogic domain with this database. You will then validate this configuration by logging in to the administration console as various users defined in this database.

The following is the updated WebLogic security realm:



Dependencies

The following prior practices must be completed (or equivalent solutions run) before beginning this practice:

- *Authenticate Using an External LDAP*

Tasks

1. Create the database schema for the SQL authenticator.
   a. Launch **sqlplus** from the command line:

```
> sqlplus / as sysdba
```

   b. Execute the `sqlauth_tables.sql` file, located in the folder for this exercise:

```
SQL> @<LAB>/resources/sql/sqlauth_tables.sql
```

   c. Confirm that three new (empty) tables were created:

```
SQL> SELECT * FROM WLSDATA.USERS;
SQL> SELECT * FROM WLSDATA.GROUPS;
SQL> SELECT * FROM WLSDATA.GROUPMEMBERS;
```

2. Add the SQL authenticator to the security realm.
   a. Launch the console and **Lock** it.
   b. Locate and select the **myrealm** security realm.
   c. Click the **Providers > Authentication** tab. Then click **New**.
   d. Enter the following values:

Field	Value
Name	MedRecDB
Type	SQLAuthenticator

   Click **OK**.
   e. Click the new provider.
   f. Set the **Control Flag** to **SUFFICIENT** and click **Save**.
   g. Click the **Configuration > Provider Specific** tab.
   h. Enter the following values:

Field	Value
Data Source Name	MedRecGlobalDataSourceXA
Group Membership Searching	limited
Max Group Membership Search Level	3

   Click **Save**.
   i. **Activate** your changes.
   j. Restart the administration server.
3. Add users and groups to the SQL provider.
   a. Return to the console.
   b. Select the **myrealm** security realm once again.
   c. Click the **Users and Groups > Users** tab. Then click **New**.

   Why? The SQL authentication provider supports both read and write interfaces. Therefore you can use the console to create new users and groups in the database.
   d. Enter the following values:

Field	Value
Name	bbendit
Description	Bill Bendit
Provider	MedRecDB
Password, Confirm Password	Welcome1

   Click **OK**.
   e. Click the **Users and Groups > Groups** tab. Then click **New**.
   f. Enter the following values:

Field	Value
Name	MedRecCC
Description	MedRec Command Center
Provider	MedRecDB

Click **OK**.

g. Locate and select the **bbendit** user.

h. Click the **Groups** tab.

i. Move the **MedRecCC** group from the **Available** column to the **Chosen** column. Click **Save**.

j. Repeat the previous steps to view the latest contents of the database. Here is an example:

```
SQL> SELECT * FROM WLSDATA.USERS;

U_NAME  U_PASSWORD          U_DESCRIPTION
--------------------------------------------------------
bbendit {SHA-1}YSIKC7Em1vo= Bill Bendit
```

4. Add an external database group to a global role.

a. Return to the **myrealm** security realm once again.

b. Click the **Roles and Policies > Realm Roles** tab.

c. In the **Roles** table, locate **Global Roles > Roles > Monitor**, and click its **View Role Conditions** link.

d. Click **Add Conditions**.

e. Verify that the **Predicate List** is set to **Group**, and click **Next**.

f. For **Group Argument Name**, enter `MedRecCC` and click **Add**.

g. Click **Finish**. Then click **Save**.

h. Log out of the console, and log back in as `bbendit/Welcome1`.

i. After successfully logging in, explore the console. Bill should be able to view and monitor resources, but not create or modify them. Also notice that the **Change Center** buttons are absent.

Practice Solution

Perform the following tasks if you did not complete this practice and want to use the finished solution.

Solution Tasks

1. If not done previously, perform the section of the instructions entitled “Re-create the MedRec domain” found in the “Authenticate Using an External LDAP” practice.
2. Launch the Lab Framework command shell by executing the file: `<STUDENT>/bin/prompt.sh`.
3. Change the current directory to `<LAB>`.
4. Execute the following:

```
ant setup_solution
```

5. The Lab Framework performs the following:
   a. Makes a backup copy of your current work
   b. Starts the database if it is not already started
   c. Updates the database with the authentication provider schema
   d. Starts the administration server if it is not already started
   e. Uses WLST to create the SQL Authentication provider
6. Kill the administration server.
7. Execute the following:

```
ant setup_security
```

8. The Lab Framework performs the following:
   a. Restarts the administration server
   b. Uses WLST to add users and groups via the SQL provider
   c. Uses WLST to update global roles

Practice 16-3: Define Password Rules

Duration: 20 minutes

Objectives

After completing this practice, you should be able to:

- Configure password validation rules
- Create users and verify password rules

Overview

MedRec corporate security policies dictate that all internal applications require passwords that meet the following requirements:

- Do not contain the username
- Consist of at least eight characters
- Have at least one lowercase character
- Have at least one uppercase character
- Have at least one numeric character
- Have at least one character that is neither alphabetic nor numeric

Dependencies

The following prior practices must be completed (or equivalent solutions run) before beginning this practice:

- *Create a Custom Domain Template* or *Authenticate Using an External LDAP*

Tasks

1. Initialize rules for password validation.
   a. Launch the console, log in as `weblogic/Welcome1`, and lock the console.
   b. Locate and select the **myrealm** security realm.
   c. Click the **Providers > Password Validation** tab.
   d. Select the default **SystemPasswordValidator**.
   e. Click the **Configuration > Provider Specific** tab.
   f. Update the following values:

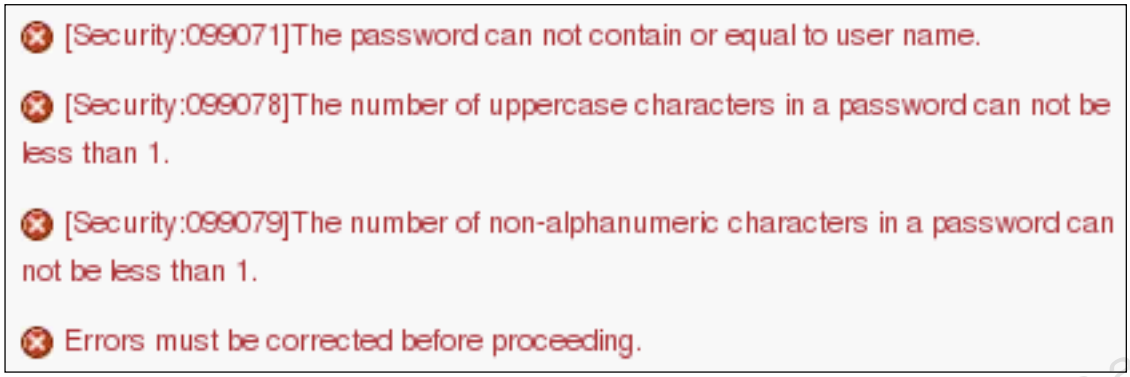
Field	Value
Reject if Password Contains the User Name	<checked>
Minimum Number of Numeric Characters	1
Minimum Number of Lower Case Characters	1
Minimum Number of Upper Case Characters	1
Minimum Number of Non-Alphanumeric Characters	1

   Click **Save**.
   g. **Activate** your changes.
2. Test the password validation rules.
   a. Return to the **myrealm** security realm.

b. Click the **Users and Groups > Users** tab. Then click **New**.

c. Enter the **Name** `asoggy` and the **Password** `asoggy111`. Select any available **Provider**. Click **OK**.

d. Confirm that the password failed validation:



e. Repeat the previous steps to create the same user with the following passwords. All of these attempts should fail as well:
   - `111222333`
   - `Welcome`
   - `Welcome1`

f. Attempt to create the user once again, using the password `Welc@me1`. Verify that user creation succeeded.

Practice Solution

Perform the following tasks if you did not complete this practice and want to use the finished solution.

Solution Tasks

1. If not done previously, perform the section of the instructions entitled “Re-create the MedRec domain” found in the “Authenticate Using an External LDAP” practice.
2. Launch the Lab Framework command shell by executing the `<STUDENT>/bin/prompt.sh` file.
3. Change the current directory to `<LAB>`.
4. Execute the following:

```
ant setup_solution
```

5. The Lab Framework performs the following:
   a. Makes a backup copy of your current work
   b. Starts the administration server if it is not already started
   c. Uses WLST to update the password validation provider

Practices for Lesson 17

Chapter 17

Overview of Practices for Lesson 17

Practices Overview

In the practices for this lesson, you perform some simple load tests of your WebLogic infrastructure, while attempting to increase performance by using several common techniques along with a standard methodology.

Naming Conventions

The practices in this lesson use the following variable names to refer to commonly used locations on your file system:

Variable	Path
<GRINDER_HOME>	/u01/app/opensource/Grinder

Practice 17-1: Tune a Server JVM

Duration: 40 minutes

Objectives

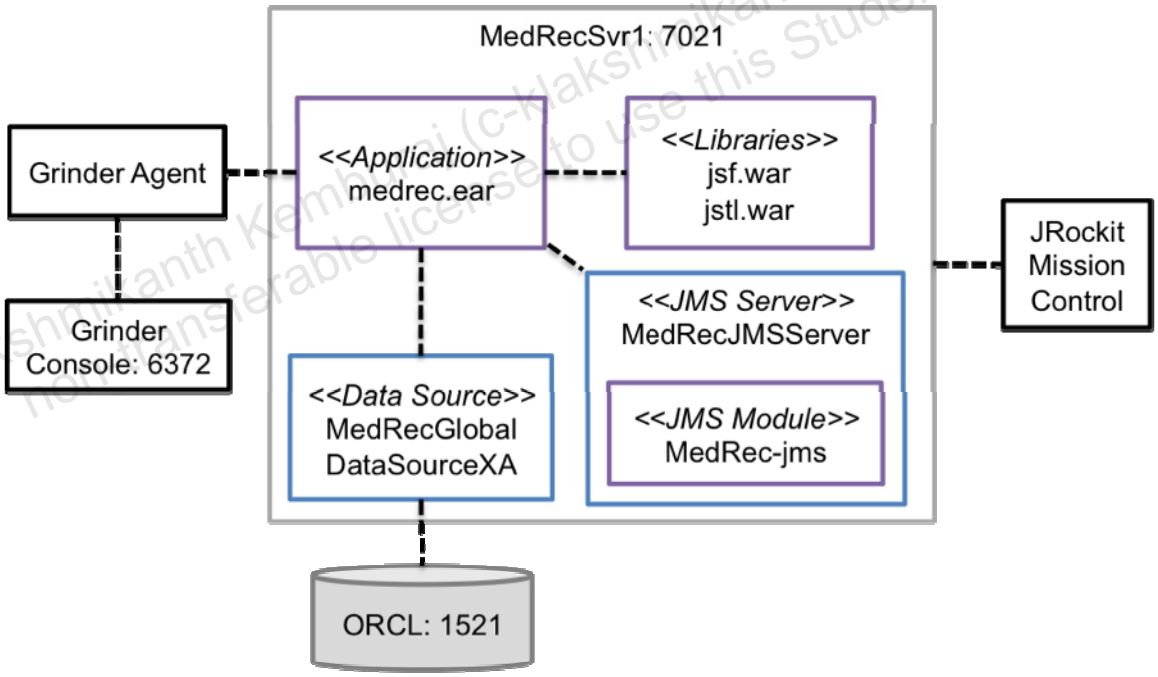
After completing this practice, you should be able to:

- Record a Grinder script by using the Grinder TCP proxy
- Start a Grinder agent from the Grinder console
- Modify server JVM heap settings
- Monitor heap usage with JRockit Mission Control

Overview

One of the easiest ways to tune the performance of WebLogic Server is through the properties of its host virtual machine. For production environments, Oracle recommends the JRockit JVM. Like all JVMs, JRockit supports several properties that affect how it allocates and garbage collects server memory at run time. The optimal values of these properties must be determined experimentally. MedRec IT has configured a simple performance testing environment for this purpose based on the Grinder open source load simulation tool.

The following is MedRec's performance testing environment:



Dependencies

The following prior practice must be completed (or equivalent solutions run) before beginning this practice:

- *Create a Custom Domain Template* or *Authenticate Using an External LDAP*

Tasks

1. Configure the Grinder proxy.
   a. Launch Firefox.

b. Select **Edit > Preferences**.
c. Click **Advanced**.
d. Click the **Network** tab.
e. Click the **Settings** button.
f. Copy and paste the current settings into a text file, so that they can be restored later.
g. Edit the following values:

Field	Value
HTTP Proxy	localhost
Port	8001
No Proxy For	<empty>

Click **OK**.

h. Click **Close**. Restart Firefox.

You will receive an error because the proxy server is not yet running.

i. Create a new folder, `<WORK>/grinder`. Copy the contents of `<LAB>/resources/grinder` to this location.
j. If one is not already open, launch a new Lab Framework command shell by executing the `<STUDENT>/bin/prompt.sh` file.
k. Navigate to `<WORK>/grinder` and execute the following:

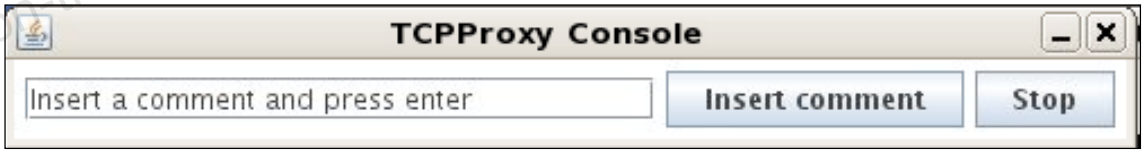
```
source setEnv.sh

java net.grinder.TCPProxy -console -http >
viewrecords-updateprofile.py
```

l. Confirm that the proxy started successfully:

```
(tcpproxy): Engine initialised, listening on port 8001
```

m. The TCPProxy Console window is also displayed:



2. Record the test case.
   a. Start your administration server if it is not already running.
   b. Start `MedRecSvr1` and use the JRockit JVM:

```
export JAVA_VENDOR="Oracle"
./startServer1.sh
```

   c. Inspect the terminal running `MedRecSvr1`. Determine the current server heap settings. For example:

```
JAVA Memory arguments: -Xms512m -Xmx512m
```

   Tip: You can also view server JVM settings by using the administration console.
   d. Direct your Web browser to the MedRec application: `http://localhost:7021/medrec/index.action`.
   e. Log in as the `fred@golf.com/weblogic` patient.

f. Click the **Successfully logged in! Click here to continue** link.

Tip: If you make a mistake, you will need to first kill and restart the proxy server, and then start over from the initial URL.

g. Click one of the links in the **Visit Records** table.

h. Click your browser's back button to return to the patient's main page. Test the remaining records.

i. Click the **Profile** button.

j. Click the **Logout** button.

k. In the TCPProxy Console, click the **Stop** button.

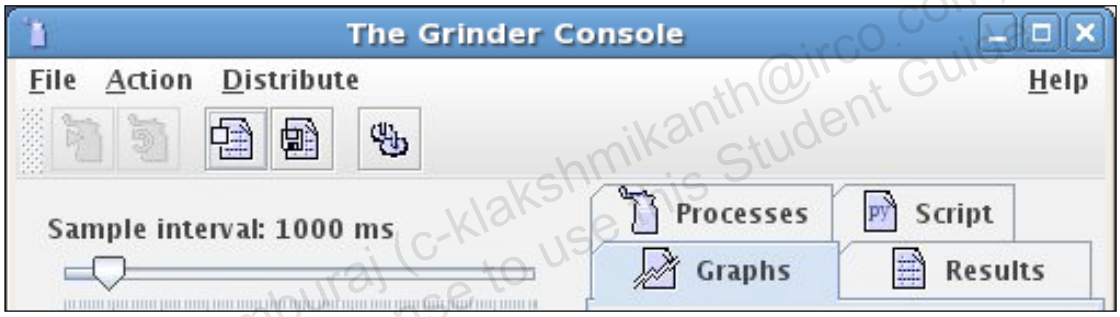
l. Inspect the generated test script, `viewrecords-updateprofile.py`.

m. Edit your Firefox proxy settings once again, and restore the previous values.

3. Start the Grinder console and agent.

a. From the Lab Framework command prompt you used earlier, ensure you are at `<WORK>/grinder`, and execute the following:

```
java net.grinder.Console
```



Tip: If you closed that Lab Framework prompt from earlier, open a new one, navigate to `<WORK>/grinder`, and before you execute the Java class shown above you must first execute:

```
source setEnv.sh
```

b. Locate the **Ignore 0 Samples** field and set its value to 10.

This setting gives WebLogic Server time to tune its own performance based on the simulated load.

c. Create a new file `<WORK>/grinder/grinder.properties`.

d. Edit the file and add the following:

```
grinder.processes=1
grinder.threads=50
grinder.runs=2
grinder.logDirectory=logs
grinder.numberOfOldLogs=2
grinder.script=viewrecords-updateprofile.py
```

e. Save your changes.

f. Launch a second Lab Framework command shell.

g. Navigate to `<WORK>/grinder` and execute the following:

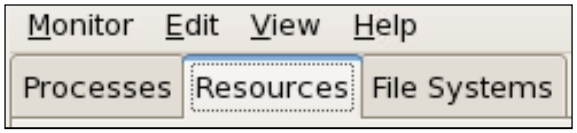
```
source setEnv.sh
```

```
java net.grinder.Grinder
```

h. Confirm that your agent started successfully and that it discovered your console:

```
(agent): connected to console at localhost/127.0.0.1:6372
(agent): waiting for console signal
```

4. Run the test case.
   a. On the Linux toolbar, select **System > Administration > System Monitor**.
   b. Click the **Resources** tab:

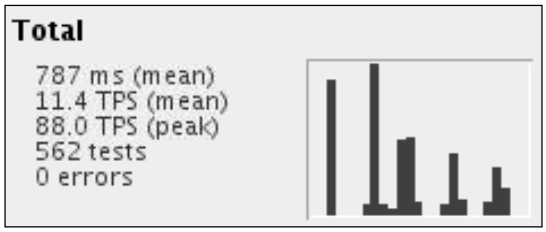


   c. Return to the Grinder console. Click the **Results** tab.
   d. On the main toolbar, click the **Start the worker processes** button. Click **OK** if prompted to run the agent's default property file. Also, if it is not already selected, click the **Start collecting statistics** button on the toolbar.
   e. As the agent runs, the left pane displays the number of **Collected Samples** along with the response time and throughput statistics:

   f. As the agent runs, use the Linux System Monitor to monitor CPU utilization.
   g. In the **Results** tab of the Grinder console, locate the **Successful Tests** column.
   h. When most of the rows are within the range of 50 to 100 (after about 3 to 5 minutes), click the **Stop Collecting** button in the left pane.
   i. Return to the Grinder agent and verify that the test script finished:

```
(agent): finished, waiting for console signal
```

   j. Locate the **Total** section in the lower left area of the console. For example:



   k. Record the following values:
      - Mean response time (ms): _____
      - Mean TPS: _____
   l. On the main toolbar, click the **Reset the worker processes** button. When prompted to reset the console statistics, click **Yes**.
5. Modify the server heap settings.
   a. Locate the terminal running `MedRecSvr1`. Kill the server.
   b. Within the same terminal, execute the following:

```
export USER_MEM_ARGS="-Xms768m -Xmx768m"
./startServer1.sh
```

Tip: If you start a new terminal, be sure to set the JVM to JRockit as before.

c. Once again, inspect the server's output to verify the new heap settings.

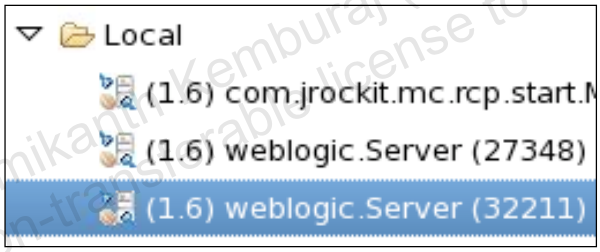
6. Run the performance test again.
   a. Repeat the previous section entitled "Run the test case." Be sure to re-enable data collection in the Grinder console.
   b. Did the performance (mean response time and mean TPS) improve?
   c. Kill the server again and restart it using these heap settings:

```
-Xms1g -Xmx1g
```

   d. Run the performance test a third time.
   e. Record and compare the results.
7. Monitor heap usage with JRockit Mission Control.
   a. Determine the process ID of `MedRecSvr1`:

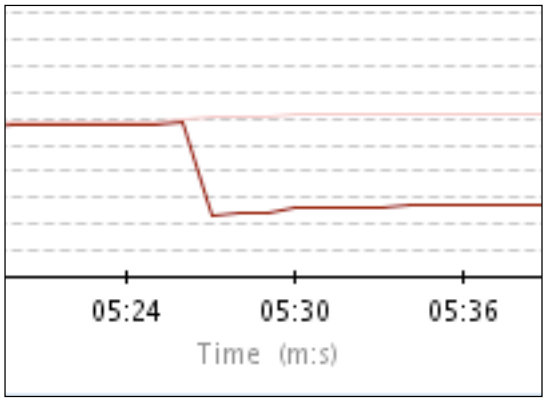
```
ps -ef | grep MedRecSvr1
```

   b. Shut down the Grinder agent and console.
   c. From a Lab Framework command shell, execute `jrmc`. The JRockit Mission Control application is displayed.
   d. Close the **Welcome** panel.
   e. In the **JVM Browser** panel, locate the JVM for `MedRecSvr1` using the process ID. Here is an example:



   f. Right-click this JVM and select **Start Console**.
   g. Locate the **Memory** section found at the bottom of **General > Overview** tab.
   h. Repeat the previous steps to run the Grinder agent again.

      Because the Grinder console is not running, the agent will immediately begin running the test case.
   i. Use Mission Control Console to monitor the heap statistics while the server is under load. Garbage collections present themselves as a drop in used heap:



j. After confirming that the Grinder agent finished, feel free to explore other features of the Mission Control Console if time permits. When finished, close the console.

Practice Solution

No major modifications are made to your domain during this practice. However, to perform subsequent tuning practices, you will need the Grinder scripts. If it is not already present, copy the `<LAB>/solution/grinder` folder to the location `<WORK>`.

Practice 17-2: Tune Server Performance

Duration: 30 minutes

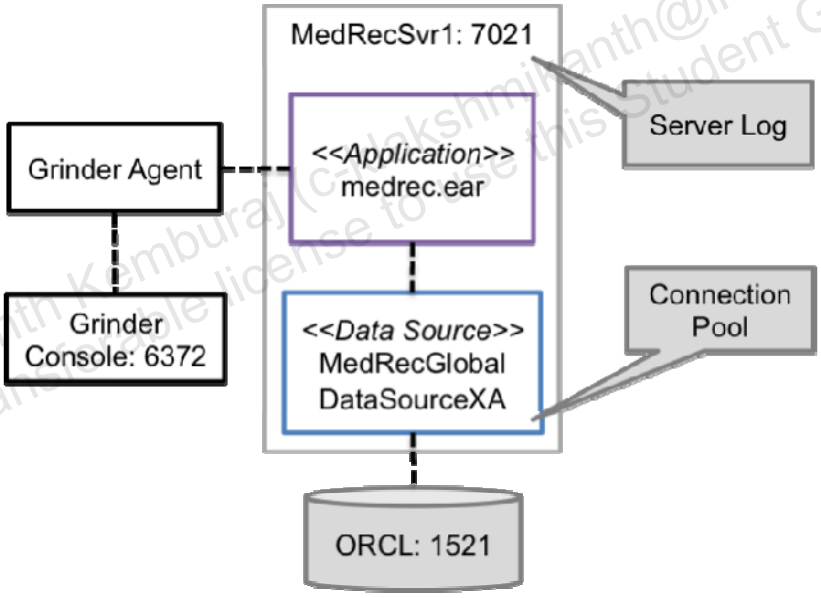
Objectives

After completing this practice, you should be able to:

- Monitor data source performance
- Tune data source connection pool settings
- Tune server log settings

Overview

Many simple methods are available to help increase the performance of your WebLogic Server environment. These methods exist at the domain, server, service (JDBC, JMS, and so on), and application levels. Generally speaking, many systems are "I/O bound," meaning that back-end file system and databases are the primary bottleneck. In this practice, you will tune two I/O-related variables: the size of MedRec's database connection pool as well as the amount of server logging. You will also continue to use the Grinder as your primary performance testing tool. See the following diagram:



Dependencies

The following prior practice must be completed (or equivalent solutions run) before beginning this practice:

- *Tune a Server JVM*

Tasks

1. Set the initial connection pool size.
   a. Kill and restart `MedRecSvr1` to reset its runtime statistics.
   b. Launch the administration console.
   c. Locate and select the data source named **MedRecGlobalDataSourceXA**.
   d. **Lock** the console.

e. Click the data source's **Configuration > Connection Pool** tab.
f. Edit the following values:

Field	Value
Initial Capacity	2
Maximum Capacity	2

Click **Save**.
g. **Activate** your changes.
h. Click the **Control** tab.
i. Select the check box for **MedRecSvr1** and click **Reset**. When prompted, click **Yes**.

2. Configure data source monitoring.
a. Click the **Monitoring > Statistics** tab.
b. Click **Customize this table**.
c. Move the following columns from **Available** to **Chosen**:
- **Active Connections High Count**
- **Current Capacity**
- **Curr Capacity High Count**
- **Reserve Request Count**
- **Wait Seconds High Count**
- **Waiting For Connection High Count**

Click **Apply**.
d. Note the value of **Current Capacity**.

3. Monitor a data source during a stress test.
a. Edit the file `<WORK>/grinder/grinder.properties`. Add or modify the following lines:

```
grinder.runs=10
grinder.sleepTimeFactor=0.1
```

b. Launch the Grinder console from a Lab Framework command prompt.
c. Launch the Grinder agent from a Lab Framework command prompt.
d. Use the Grinder console to **Start the worker processes** and **Start collecting statistics**.
e. After the agent finishes, record the **response time (mean)** and **TPS (mean)**.
f. Use the Grinder console to **Reset the Worker Processes**.
g. Return to the administration console and refresh your browser.
h. Inspect the value of **Active Connections High Count**.

Ideally, we want this value to be slightly less than the **Current Capacity**. Otherwise, we will need to increase the capacity.
i. Similarly, inspect the values of **Wait Seconds High Count** and **Waiting For Connection High Count**. Ideally, these values should be at or near 0.

4. Increase the connection pool size.
a. Repeat the previous steps to update the data source connection pool settings:

Field	Value
Initial Capacity	4
Maximum Capacity	4

Be sure to **Reset** the data source again as well.

b. Return to the **Monitoring > Statistics** tab.
c. Confirm that the new **Current Capacity** is 4.

5. Run the stress test again.
   a. Repeat the previous steps to run the Grinder again.
   b. Record the latest statistics and compare them to those of the prior test.
   c. Return to the administration console and refresh the data source statistics.
   d. If the value of **Active Connections High Count** is still equal to the data source capacity, repeat the previous steps to increase the data source's capacity by 2. Then run the test again.
6. Tune server log settings.
   a. In the administration console, locate and select the server **MedRecSvr1**.
   b. **Lock** the console.
   c. Click the **Logging > General** tab.
   d. Click the **Advanced** link.
   e. Edit the following values:

Field	Value
Log file: Severity level	Warning
Standard out: Severity level	Error
Domain log broadcaster: Severity level	Critical
Memory buffer: Severity level	Notice

Click **Save**.

   f. Click the **Logging > HTTP** tab.
   g. Clear the check box **HTTP access log file enabled**, and click **Save**.
   h. **Activate** your changes.
7. Test the latest performance.
   a. Repeat the previous steps to run the Grinder again.
   b. Record the latest statistics and compare them to those of the prior test.

Practice Solution

Perform the following tasks if you did not complete this practice and want to use the finished solution.

Solution Tasks

1. Launch the Lab Framework command shell by executing the file `<STUDENT>/bin/prompt.sh`.
2. Change the current directory to `<LAB>`.
3. Execute the following:

```
ant setup_solution
```

4. The Lab Framework performs the following:
   a. Makes a backup copy of your current work
   b. Starts the administration and managed server if they are not already started
   c. Uses WLST to update the JDBC data source connection pool settings
   d. Uses WLST to update the server log settings

Practice 17-3: Tune Performance Using Work Managers

Duration: 35 minutes

Objectives

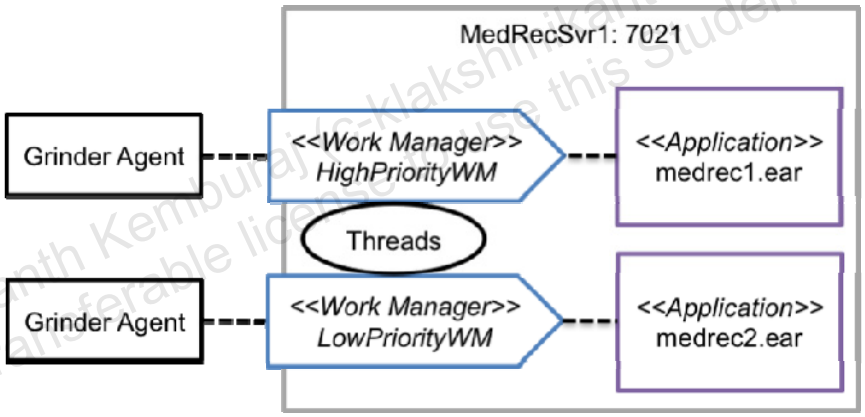
After completing this practice, you should be able to:

- Configure server work managers
- Assign request classes to work managers
- Use deployment plans to associate work managers with applications
- Verify Work Manager parameters during a stress test

Overview

Each WebLogic Server instance uses a self-tuning thread pool to process all requests, and by default all types of requests have the same level of service. Work managers allow administrators to prioritize different services, applications, and application components, using request classes and constraints.

The following diagram shows the test environment that you will use to experiment with the work manager feature. You will deploy multiple versions of the MedRec application and stress test both concurrently using the Grinder:



Dependencies

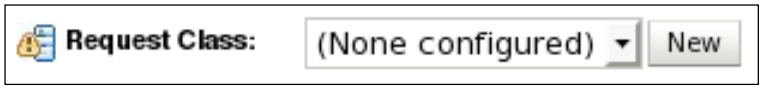
The following prior practice must be completed (or equivalent solutions run) before beginning this practice:

- *Tune a Server JVM*

Tasks

1. Create global work managers.
   a. Launch the administration console and **Lock** it.
   b. In the **Domain Structure** panel, select **Environment > Work Managers**.
   c. Click **New**.
   d. Select the **Work Manager** option and click **Next**.
   e. **Name** the work manager `HighPriorityWM` and click **Next**.
   f. Target the work manager to **MedRecSvr1** and click **Finish**.
   g. Edit the new work manager.

h. Locate the **Request Class** field and click **New**:



i. Select the option **Fair Share Request Class** and click **Next**.

j. Enter the following values:

Field	Value
Name	FairShare90
Fair Share	90

Click **Next**.

k. Target the request class to **MedRecSvr1** and click **Finish**.

l. Click **Save**.

m. Repeat these steps to create a second work manager and associated request class, using the details below:

Resource	Field	Value
Work Manager	**Name**	LowPriorityWM
	Target	MedRecSvr1
Fair Share Request Class	**Name**	FairShare10
	Fair Share	10
	Target	MedRecSvr1

n. **Activate** your changes.

2. Deploy a copy of the MedRec application for each work manager.

a. Launch the Lab Framework command shell by executing the `<STUDENT>/bin/prompt.sh` file.

b. Change the current directory to <LAB> and execute the following:

```
ant setup_exercise
```

c. The Lab Framework performs the following:
   - Makes a backup copy of your current work
   - Starts the administration and managed server if they are not already running
   - Copies Grinder configuration files to your work directory
   - Uses WLST to remove the current MedRec application
   - Deploys two new MedRec applications and associates them with deployment plans

d. Locate the deployment plan files at `<WORK>/applications`. Confirm that each refers to one of your work managers. For example:

```
<variable>
   <name>MedRecWeb_DispatchPolicy</name>
   <value>HighPriorityWM</value>
</variable>
```

e. Test that each application has deployed successfully using a browser:
   - `http://localhost:7021/`**`medrec1`**`/index.action`

- http://localhost:7021/**medrec2**/index.action

f. Return to the console.
g. In the **Domain Structure** panel, select **Deployments**.
h. Click the **Monitoring > Workload** tab.
i. Locate the **Work Managers** table. Use the **Completed Requests** column to confirm that each application is linked to the correct work manager:

Name	Server	Application	Pending Requests	Completed Requests
LowPriorityWM	MedRecSvr1	medrec1	0	0
LowPriorityWM	MedRecSvr1	medrec2	0	7
HighPriorityWM	MedRecSvr1	medrec1	0	7

3. Use Grinder agents to stress test the server.
   a. Close the Grinder console and agent if they are running.
   b. Copy the contents of the <LAB>/exercise/files/grinder folder to <WORK>/grinder. Click **Yes**, if asked if you want to overwrite files.
   c. Launch two Lab Framework command prompts. Navigate to <WORK>/grinder and execute the following:

```
source setEnv.sh
```

   d. Start a Grinder agent in each command prompt at approximately the same time:

```
java net.grinder.Grinder grinder-medrec1.properties

java net.grinder.Grinder grinder-medrec2.properties
```

   e. Wait for both agents to finish and exit.
   f. Return to the console and once again verify that the requests to each application were serviced by their dedicated Work Manager.
4. Inspect the performance results.
   a. Open the <WORK>/grinder/logs-**medrec1**/out_*nnn*.log file.
   b. At the end of the log file, locate the row starting with the text "Totals." Record the following performance data from this row:
      - Mean Test Time (ms): _____
      - TPS: _____

      Tip: You may wish to disable line wrapping in your text editor.
   c. Repeat the previous steps on the <WORK>/grinder/logs-**medrec2**/out_*nnn*.log file. Compare the results.
   d. When finished with the practice, execute the following WLST script:

```
<LAB>/resources/restoreApp.py
```

      This script removes the medrec1 and medrec2 applications from your domain and deploys the original application.

Practice Solution

Perform the following tasks if you did not complete this practice and want to use the finished solution.

Solution Tasks

1. Launch the Lab Framework command shell by executing the `<STUDENT>/bin/prompt.sh` file.
2. Change the current directory to `<LAB>`.
3. Execute the following:

```
ant setup_solution
```

4. The Lab Framework uses WLST to create two work managers.
5. If not done previously, perform the section of the instructions entitled “Deploy a copy of the MedRec application for each work manager.”
6. When finished with this practice, execute the following WLST script: `<LAB>/resources/restoreApp.py`.

Practices for Lesson 18

Chapter 18

Overview of Practices for Lesson 18

Practices Overview

In the practices for this lesson, you take several different approaches to monitoring and performing diagnostics on WebLogic Server.

Naming Conventions

The practices in this lesson use the following variable names to refer to commonly used locations on your file system:

Variable	Path
<LDAP_HOME>	/u01/app/opensource/OpenDS

Practice 18-1: Configure and Monitor Diagnostic Data

Duration: 40 minutes

Objectives

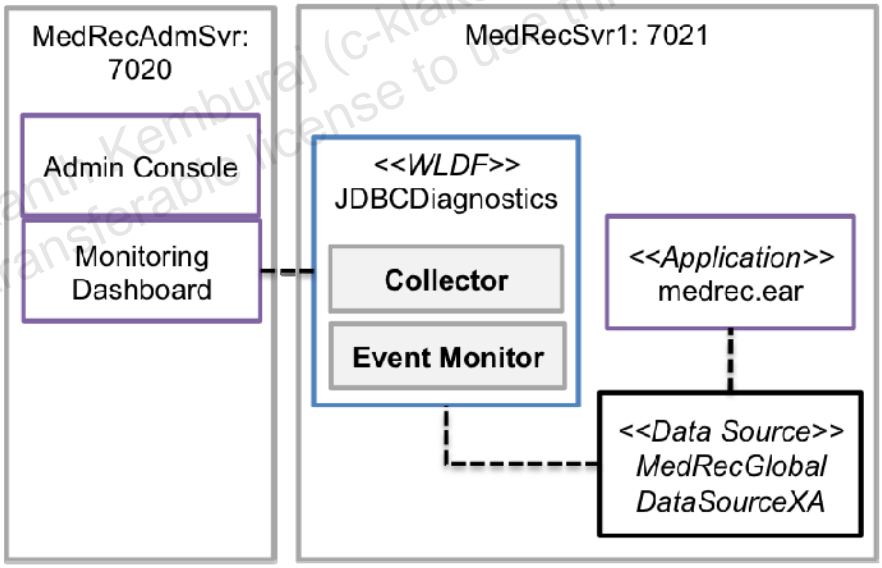
After completing this practice, you should be able to:

- Capture a server diagnostic image
- Collect and record MBean metrics by using WLDF
- Generate diagnostic events by using WLDF monitors
- Use standard diagnostic views in the WLS monitoring dashboard
- Create a custom view in the dashboard

Overview

Patients have reported intermittent problems with the MedRec application, particularly with the profile management features. Before contacting the development team and beginning troubleshooting, however, the IT group would like to get more insight into the server and application. To get started, you suspect that these issues may be due to the JDBC data source or the application components that interact with it during profile management. The WebLogic Diagnostics Framework (WLDF) allows you to monitor server subsystems in real time and to also collect and record statistics over time for later analysis.

In this exercise, you will create and configure a WLDF module as shown below:



Dependencies

The following prior practice must be completed (or equivalent solutions run) before beginning this practice:

- *Create a Custom Domain Template* or *Authenticate Using an External LDAP*

Tasks

1. Capture a server diagnostic image.
   a. Launch the console.
   b. In the **Domain Structure** panel, select **Diagnostics > Diagnostic Images**.

c. Select the radio button for **MedRecSvr1** and click the **Capture Image** button.
d. Note the default location of the image file and click **OK**.
e. Using the Linux File Browser, locate the new zip file found at the following location: `<WORK>/domains/MedRecDomain/servers/MedRecSvr1/logs/diagnostic_images`.
f. Right-click this file and select **Extract Here**.
g. Locate the file in the archive named `JVM.img`. Open it in any editor and browse its contents.
h. Similarly, browse the contents of the `JDBC.img` file. Notice the single data source targeted to `MedRecSvr1`. This file also includes the number of connections currently allocated to the data source along with some basic statistics for each connection. For example:

```
Resource Pool:MedRecGlobalDataSourceXA:dumpPool available[0] =
   autoCommit=true,enabled=true,isXA=true,isJTS=false,
   vendorID=0,connUsed=false, ...
```

2. Create a diagnostic module.
   a. Return to the console and **Lock** it.
   b. Select **Diagnostics > Diagnostic Modules**.
   c. Click **New**.
   d. **Name** the module `JDBCDiagnostics` and click **OK**.
   e. Edit the new module.
   f. Click the **Targets** tab.
   g. Select **MedRecSvr1** and click **Save**.
4. Define a metric collector (harvester).
   a. Click the **Configuration > Collected Metrics** tab.
   b. Set the **Sampling Period** to 30000 (30 seconds) and click **Save**.
   c. Click **New**.
   d. Click **Next**.
   Note: Because this module is targeted to a managed server, only the Server Runtime MBeans apply.
   e. From the select box, choose the **weblogic.management.runtime. JDBCConnectionPoolRuntimeMBean** option. Click **Next**.
   f. Move the following attributes from the **Available** column to the **Chosen** column:
      - **CurrCapacity**
      - **LeakedConnectionCount**
      - **WaitSecondsHighCount**

      Click **Finish**.
5. Define a diagnostic monitor and action (instrumentation).
   a. Click the **Configuration > Instrumentation** tab.
   b. Verify that the **Enabled** check box is *not* selected.
   c. Click the **Add/Remove** button.
   d. Move the **JDBC_Before_Connection_Internal** monitor type from the **Available** column to the **Chosen** column. Then click **OK**.

e. Edit the new monitor.
f. Locate the **Actions** field.
g. Move **StackDumpAction** action from the **Available** column to the **Chosen** column. Then click **Save**.
h. **Activate** your changes.

6. View collected metrics in the WLDF archive.
   a. Direct a separate Web browser to the MedRec application: `http://localhost:7021/medrec/index.action`.
   b. Log in as the `fred@golf.com/weblogic` patient.
   c. Click the **Profile** button.
   d. Enter `weblogic` for the password. Click **Save**.
   e. Click the **Logout** button.
   f. Return to the console.
   g. In the **Domain Structure** panel, select **Diagnostics > Log Files**.
   h. Select the radio button for **HarvestedDataArchive** for `MedRecSvr1`:

○	HarvestedDataArchive	Metric Data	MedRecAdmSvr
◉	HarvestedDataArchive	Metric Data	MedRecSvr1

   i. Click **View**.
   j. Click the **Customize this table** link.
   k. Remove the **Instance Name** column from the **Chosen** field. Click **Apply**.
   l. Confirm that the three selected attributes are collected approximately every 30 seconds. For example:

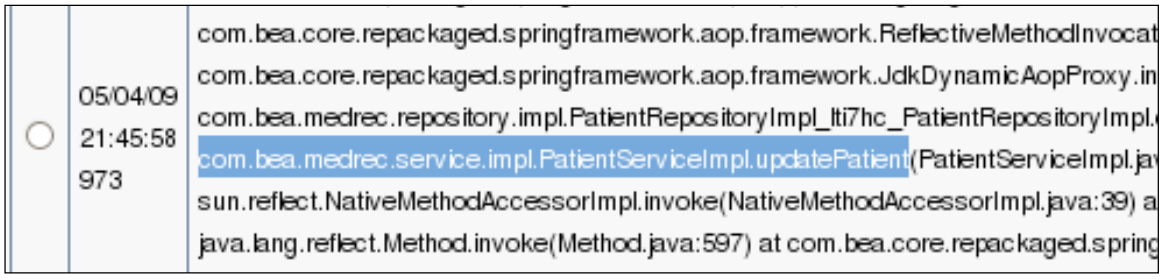
	Date	Attribute	Value
○	05/04/09 21:34:02 457	CurrCapacity	15
○	05/04/09 21:34:02 457	LeakedConnectionCount	0
○	05/04/09 21:34:02 457	WaitSecondsHighCount	0

7. View raw events in the WLDF archive.
   a. **Lock** the console once again.
   b. Edit the **JDBCDiagnostics** module. Return to the **Configuration > Instrumentation**.
   c. Select the **Enabled** check box and click **Save**. Then **Activate** your changes.
   d. Repeat the previous steps to test the MedRec application.
      Tip: The previous steps to test the MedRec application are 6.a through 6.e.
   e. Return to the console and click **Log Files** again.
   f. Select the radio button associated with the **EventsDataArchive** for `MedRecSvr1`. Click **View**.
   g. Click the **Customize this table** link.
   h. Remove all columns from the **Chosen** field, except **Date** and **Payload**. Click **Apply**.

i. Confirm that several events were posted. The **Payload** column includes the stack dump associated with the event.

j. Use the browser to perform a search for the text “com.bea.medrec,” which indicates MedRec application code.

k. Try to determine which application classes requested data source connections during your test:



You may also notice various frameworks being used by MedRec to help manage JDBC connections and persistence, such as `org.apache.openjpa` and `kodo.jdbc`.

l. Repeat the prior steps to edit the **JDBCDiagnostics** module and disable instrumentation.

8. Monitor a server using the console dashboard.

a. Click **Home** to return to the console’s home page.

b. On the far right of the page, click **Monitoring Dashboard**.

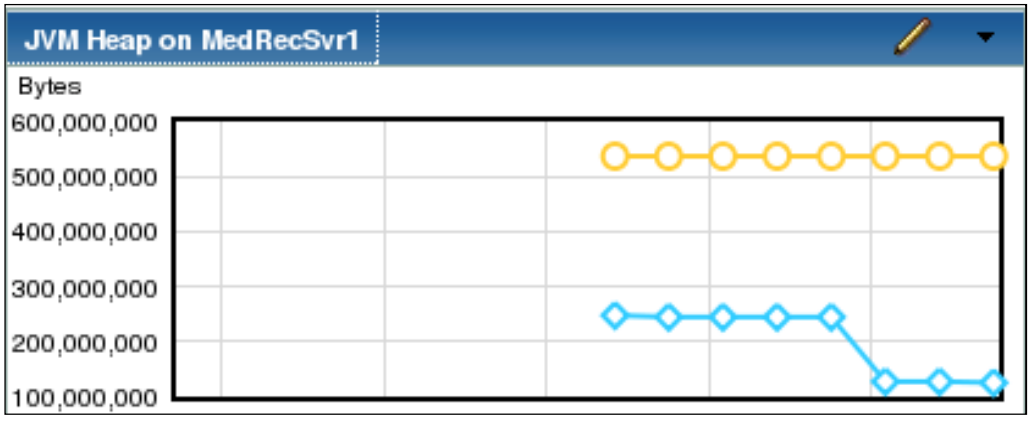
c. In the left panel, select the built-in view named **JVM Runtime Heap**:

d. Click the **Start** button found on the toolbar at the top of the dashboard to begin collecting data.

e. Test the MedRec application a third time.

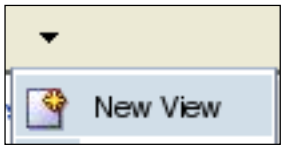
Tip: The steps to test the MedRec application are 6.a through 6.e.

f. Return to the dashboard and view the historical heap statistics for `MedRecSvr1`:

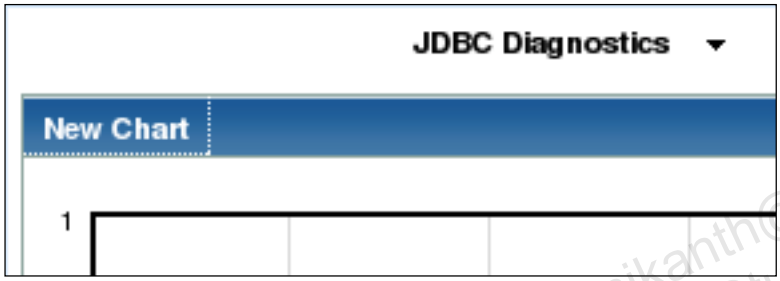


9. Create a custom view by using the dashboard.

a. Use the drop-down menu to select **New View**:

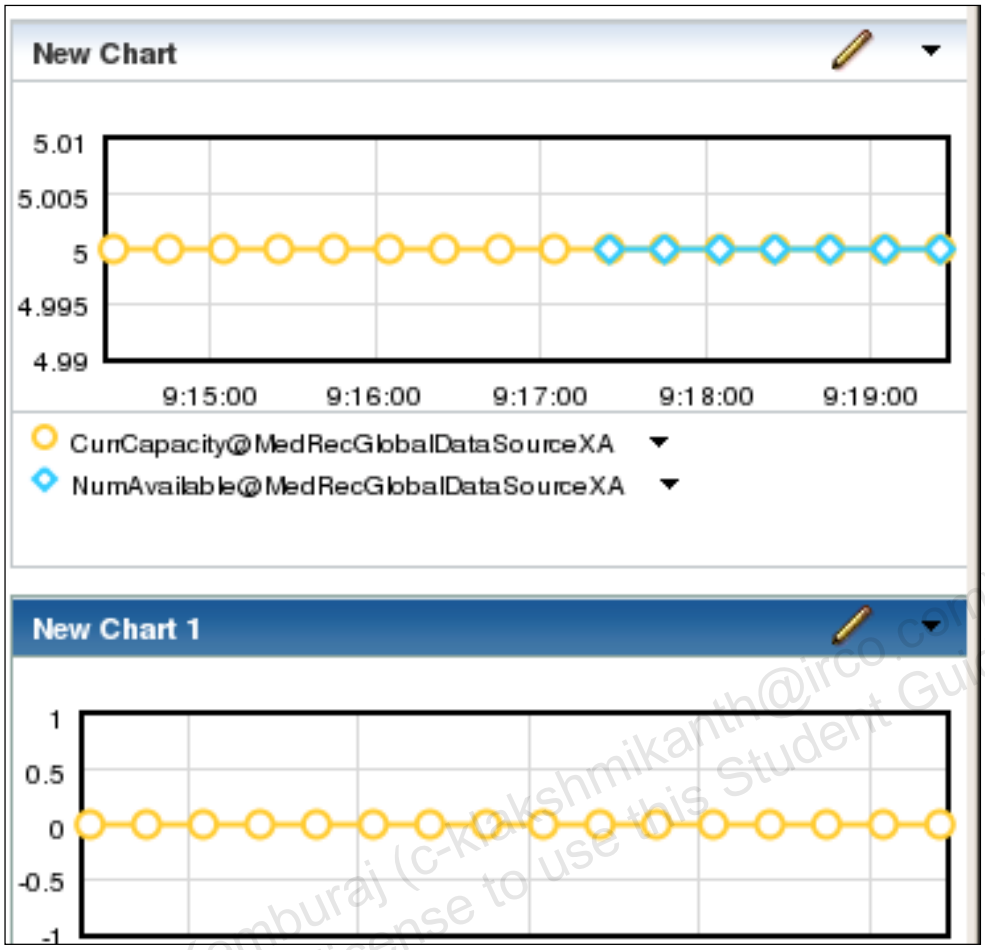


b. Name the view `JDBC Diagnostics`.
c. Click the **Metric Browser** tab.
d. Select the **MedRecSvr1** server and click **Go**.
e. In the list of available MBeans **Types** on this server, locate and select **JDBCConnectionPool**.
f. Under **Instances**, select **MedRecGlobalDataSourceXA**.
g. Drag and drop the **CurrCapacity** metric onto the right panel. A new chart with a single line graph is created:



h. **Start** collecting data for this view. Note that the time axis starts at the point when this attribute was first collected by your diagnostic module.
i. From the drop-down menu in the right panel, select **New Chart**:

j. Drag and drop the **LeakedConnectionCount** metric onto the second chart.
k. Drag and drop the **NumAvailable** attribute onto the same chart as **CurrCapacity**:



l. Retest the MedRec application.
m. Return to the dashboard and view the latest JDBC metrics.
n. When finished, click the **Stop All** button on the toolbar.

10. Deactivate WLDF.
    a. **Lock** the console.
    b. Edit the **JDBCDiagnostics** module.
    c. Click the **Targets** tab.
    d. Clear all check boxes and click **Save**. Then **Activate** your changes.

Practice Solution

Perform the following tasks if you did not complete this practice and want to use the finished solution.

Solution Tasks

1. Launch the Lab Framework command shell by executing the `<STUDENT>/bin/prompt.sh` file.
2. Change the current directory to `<LAB>`.
3. Execute the following:

```
ant setup_solution
```

4. The Lab Framework performs the following:
   a. Starts the administration and managed server if it is not already started
   b. Uses WLST to create a new WLDF module
   c. Uses WLST to add a metric collector to the module
   d. Uses WLST to add a diagnostic monitor to the module
5. When finished with this practice, perform the section of the instructions entitled “Deactivate WLDF.”

Practice 18-2: Monitor WLS by Using SNMP

Duration: 40 minutes

Objectives

After completing this practice, you should be able to:

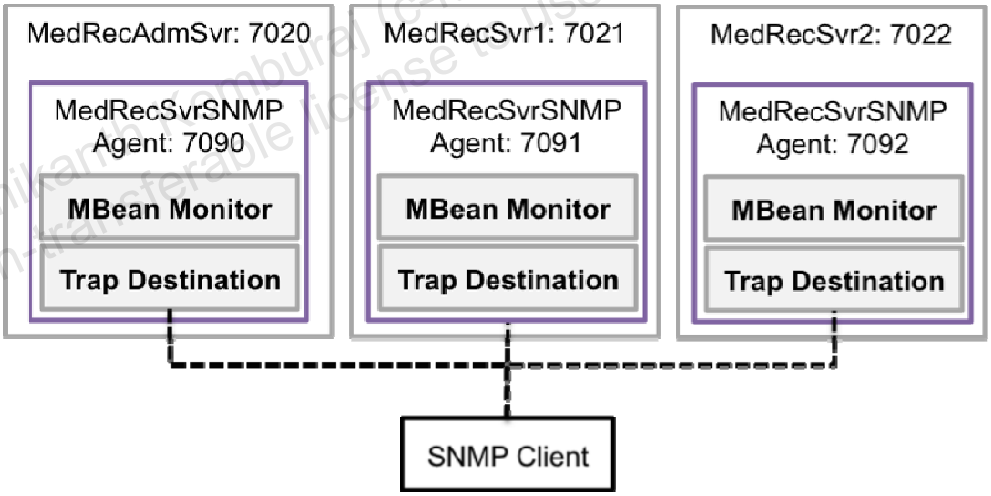
- Map WLS and SNMP credentials
- Configure, deploy, and monitor SNMP agents
- Configure an agent's trap destinations
- Use the WLS SNMP command line utilities
- Create a custom trap monitor based on MBean attributes

Overview

Several years earlier, MedRec made a significant investment in an SNMP infrastructure to centralize the management and monitoring of their IT resources, including hardware and software. Currently, the MedRec SNMP control center is based on version 3 of the protocol and also supports the INFORM message type.

Fortunately, WebLogic Server instances can host SNMP agents that integrate with its native MBean monitoring framework. These SNMP agents can respond to direct requests from SNMP management software, and can also asynchronously publish trap messages based on certain server conditions, such as a server startup or an application deployment.

MedRec's initial WebLogic Server SNMP environment is depicted below:



Dependencies

The following prior practice must be completed (or equivalent solutions run) before beginning this practice:

- *Create a Custom Domain Template* or *Authenticate Using an External LDAP*

Tasks

1. Create a WebLogic user for SNMP access.

   Note: If you did *not* complete the previous practice entitled "Authenticate Using an External LDAP," you will instead need to create the following user with the WLS console (embedded LDAP).

a. Kill any running managed servers in your `MedRecDomain`.
b. Launch a new Linux terminal.
c. Navigate to `<LDAP_HOME>/bin`.
d. Launch the OpenDS control panel:

```
> ./control-panel
```

e. When prompted, enter `Welcome1` as the password and click **OK**.
f. In the left menu, select **Directory Data > Manage Entries**.
g. Click the base node, **dc=medrec,dc=com**. Then right-click and select **New User**.
h. Enter the following values:

Field	Value
First Name	Monitoring
Last Name	Agent
Common Name	Monitoring Agent
User ID	monitoragent
Password, Password (Confirm)	Welc@me1
Naming Attribute	uid

Click **OK**.
i. After the user has been successfully created, **Close** the dialog box.
j. Locate and select the **Monitors** group.
k. Click the **Add Members** button.
l. Select the **monitoragent** user and click **OK**.
m. Click **Save Changes**.
n. Exit the OpenDS control panel.

2. Configure an SNMP credential mapping.
a. Launch the administration console.
b. From the **Domain Structure** panel, select **Diagnostics > SNMP**.
c. Click the **Security** tab.
d. Click **New**.
e. Enter the following values:

Field	Value
Credential Mapping Type	Authentication
WLS User	monitoragent
SNMP Password, Confirm SNMP Password	snmpmanager

Click **OK**.

3. Deploy an SNMP agent to servers.
a. **Lock** the console.
b. Click the **Agents** tab.
c. Locate the table named **Server SNMP Agents**. Click **New**.
d. **Name** the MedRecSvrSNMPAgent agent and click **OK**.

e. Select your new agent.
f. Edit the following values:

Field	Value
Enabled	<checked>
SNMP UDP Port	7090
Master AgentX Port	7099
Community Based Access Enabled	<unchecked>
Trap Version	V3
Authentication Protocol	MD5
Privacy Protocol	None
Inform Enabled	<checked>

Click **Save**.
g. Click the **Targets** tab.
h. Target the SNMP agent to all servers and click **Save**.

4. Configure a trap destination.
a. Click the **Configuration > Trap Destinations** tab.
b. Click **New**.
c. Enter the following values:

Field	Value
Name	MedRecSNMPManager
Host	localhost
Port	7095
Security Name	monitoragent
Security Level	Authentication Only

Click **OK**.
d. **Activate** your changes.

5. Test default traps.
a. From the **Domain Structure** panel, select **Diagnostics > SNMP**.
b. Click the **Monitoring** tab.
c. Note the initial values for **Server Start Traps** and **Server Stop Traps**.
d. Launch two Lab Framework command prompts.
e. In the first prompt, execute the following Java application:

```
java weblogic.diagnostics.snmp.cmdline.Manager
   SnmpTrapMonitor -v3 -p 7095
   -M /weblogic/diagnostics/snmp/mib -m BEA-WEBLOGIC-MIB
   -O l -u monitoragent -A snmpmanager -e MedRecSNMPManager
```

Tip: The argument `-O l` consists of letters only, not numbers.

Tip: For convenience, these commands are also found at `<LAB>/resources`.

f. Confirm that the trap monitor application started successfully:

```
Listening on port:7095
```

Tip: If you encounter problems with the SNMP utilities, trying running the application in debug mode using the **–d** argument.

g. Use **Ctrl + C** to force your administration server to shut down. Then restart it.

h. Verify that *INFORM* messages were sent to the trap monitor. For example:

```
--- Snmp Inform Received ---
    Version         : v3
    Source          : UdpEntity:127.0.0.1:7090
    Community       :
    Context         :
    TrapOID         : wlsServerShutDown
    Inform Objects  : {
   { trapTime=Thu May 07 16:05:51 EDT 2009 }
   { trapServerName=MedRecAdmSvr }
}
   ...
}

...
--- Snmp Inform Received ---
    Version         : v3
    Source          : UdpEntity:127.0.0.1:7090
    Community       :
    Context         :
    TrapOID         : wlsServerStart
    Inform Objects  : {
   { trapTime=Thu May 07 16:08:55 EDT 2009 }
   { trapServerName=MedRecAdmSvr }
}
   ...
}
```

i. Start `MedRecSvr1` and verify that similar messages are generated upon startup.

j. Return to the console and monitor the SNMP agents again. Confirm that the value for **Server Start Traps** has increased for both servers.

6. Poll an SNMP agent.

   a. Click the **Customize this table** link.

   b. Move the **UDP Listen Port** column from **Available** to **Chosen**. Click **Apply**.

   c. Note the port assignments for each server:

Location	Log Message Traps	Server Start Traps	Server Stop Traps	UDP Listen Port
MedRecAdmSvr	0	2	0	7090
MedRecDomain	0	2	0	7090
MedRecSvr1	0	1	0	7091

d. From the second Lab Framework command prompt, execute the following Java application:

```
java weblogic.diagnostics.snmp.cmdline.Manager SnmpWalk
   -h localhost -p 7090
   -M /weblogic/diagnostics/snmp/mib -m BEA-WEBLOGIC-MIB
   -O l -u monitoragent -A snmpmanager
   -e MedRecSvrSNMPAgent serverRuntimeState
```

e. Confirm that you received the current state of the administration server. For example:

```
serverRuntimeState.16.167.105.169.225.97.72.173.112.163.125.209.
5.160.16.209.21=RUNNING
```

f. Repeat the previous command, but get the same attribute from `MedRecSvr1`'s agent instead, running on port 7091.

g. Repeat the previous command, but instead get the following OID label from MedRecSvr1: `jdbcConnectionPoolRuntimeCurrCapacity`.

h. The output should resemble the following:

```
jdbcConnectionPoolRuntimeCurrCapacity.16.175.176.20.64.234.182.4
3.19.212.23.155.240.220.187.104.130=15
```

Tip: If time permits, feel free to inspect <WEBLOGIC_HOME>/server/lib/BEA-WEBLOGIC-MIB.asn1 and experiment with other OID labels.

7. Create a custom trap.

a. Return to the console and **Lock** it.

b. Locate and select your new SNMP agent, **MedRecSvrSNMPAgent**.

c. Click the **Configuration > String Monitors** tab.

d. Click **New**.

e. Enter the following values:

Field	Value
Name	DeploymentStateMonitor
Monitored MBean Type	WebAppComponentRuntime

Click **Next**.

f. For **Monitored Attribute Name**, select **Status**. Then click **Next**.

g. For **String to Compare**, enter `DEPLOYED`. Then click **Finish**.

h. Edit the new string monitor.

i. Edit the following values:

Field	Value
Monitored MBean Name	MedRecSvr1_/medrec
Polling Interval	30
Notify Match	<checked>

Click **Save**.

j. **Activate** your changes.

8. Test the custom trap.
   a. Return to the command prompt running the trap monitor application.
   b. Confirm that the application status trap was received:

```
--- Snmp Inform Received ---
    Version        : v3
    Source         : UdpEntity:127.0.0.1:7091
    Community      :
    Context        :
    TrapOID        : wlsMonitorNotification
    Inform Objects : {
   { trapTime=Thu May 07 16:49:27 EDT 2009 }
   { trapServerName=MedRecSvr1 }
   { trapMonitorType=jmx.monitor.string.matches }
   { trapMonitorThreshold=DEPLOYED }
   { trapMonitorValue=DEPLOYED }
   { trapMBeanName=com.bea:ApplicationRuntime=medrec,
     Name=MedRecSvr1_/medrec,ServerRuntime=MedRecSvr1,
     Type=WebAppComponentRuntime }
    { trapMBeanType=WebAppComponentRuntime }
    { trapAttributeName=Status }
}
...
```

   c. Return to the console.
   d. From the **Domain Structure** panel, select **Diagnostics > SNMP**.
   e. Click the **Monitoring** tab. Note the current value of the **String Monitor Traps** column.
   f. In the **Domain Structure** panel, click **Deployments**.
   g. **Stop** the **medrec** application.
   h. **Lock** the console. In **Deployments**, check the box next to **medrec**, and click **Update**. Then click **Finish**. Finally, **Activate** the changes.
   i. **Start** the application.
   j. Confirm that the custom trap is once again received by the monitor application.
9. Deactivate the SNMP agent.
   a. Kill the trap monitor application.
   b. **Lock** the console if not already locked.
   c. Edit the **MedRecSvrSNMPAgent**.

d. Clear the **Enabled** check box. Then click **Save**.
e. **Activate** your changes.

Practice Solution

Perform the following tasks if you did not complete this practice and want to use the finished solution.

Solution Tasks

1. If you previously performed the “Authenticate Using an External LDAP” practice, complete the “Create a WebLogic user for SNMP access” section of the current practice.
2. Launch the Lab Framework command shell by executing the `<STUDENT>/bin/prompt.sh` file.
3. Change the current directory to `<LAB>`.
4. Execute the following:

```
ant setup_solution
```

5. The Lab Framework performs the following:
   a. Makes a backup copy of your current work
   b. Starts the administration and managed server if not already started
   c. Uses WLST to create a new user, if not using an external LDAP
   d. Uses WLST to create a new SNMP agent
   e. Uses WLST to add a trap destination to the agent
   f. Uses WLST to add a monitor to the agent
6. Perform all of step 2 of this practice, “Configure an SNMP Credential Mapping.”
7. Enable the SNMP agent by performing steps 3.a, 3.b, and 3.e through 3.f. Notice that all the fields in 3.f. are correct except for **Enabled**. **Lock** the console, check **Enabled**, click **Save**. Then **Activate** the changes.

Practice 18-3: Debug WLS Subsystems

Duration: 35 minutes

Objectives

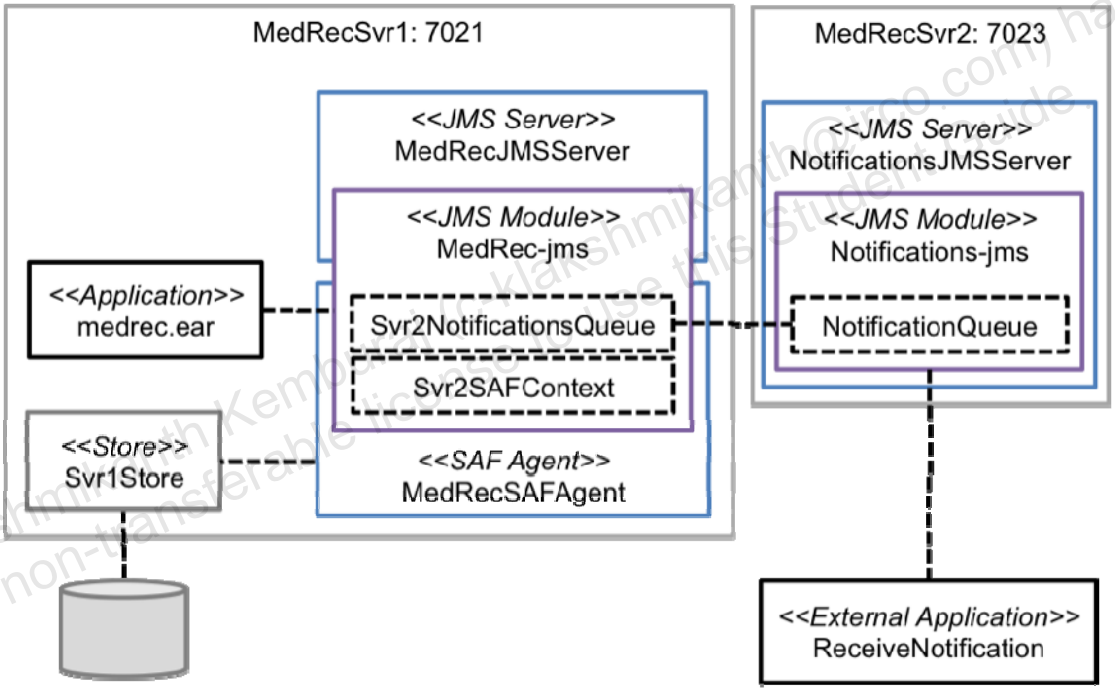
After completing this practice, you should be able to:

- Enable debugging on a server subsystem
- Configure server logs to include debug messages
- Interpret debug messages

Overview

In this exercise, you will intentionally incorporate a configuration error into a working WebLogic environment, to test server debugging log messages.

Recall the MedRec JMS Store and Forward (SAF) architecture:



Dependencies

The following prior practice must be completed (or equivalent solutions run) before beginning this practice:

- *Create a Custom Domain Template* or *Authenticate Using an External LDAP*

Tasks

1. Stage the troubleshooting scenario.
   a. Delete the `jsf.war` and `jstl.war` files from `<WORK>/applications`.
   b. Perform the **Solution Tasks** for **Practice 11-1: Store and Forward JMS Messages**.
      Why? This will put things in a known, working state before running the WLST script below to introduce bugs.
   c. Start both managed servers if not already started.

d. From your Lab Framework command prompt, navigate to `<LAB>/resources/jms`.
e. Execute the `addBugs.py` WLST script.
f. Confirm that the script executed successfully:

```
Resources updated successfully.
```

2. Test store and forward configuration.
   a. From a Linux terminal execute the `<LAB>/resources/updateStatus.sh` script.
   b. Launch the MedRec application.
      Tip: Use this URL: `http://localhost:7021/medrec/index.action`
   c. In the **Administrator** section of the home page, click **Login**.
   d. Log in using the **admin@avitek.com** username and **weblogic** password.
   e. Click the **View Pending Requests** link.
   f. Click the registration request for the new patient, **charlie@star.com**.
   g. Click the **Approve** button.
   h. Launch the administration console.
   i. From the **Domain Structure** panel, select **Services > Messaging > JMS Modules**.
   j. Click the **Notifications-jms** module.
   k. In the **Summary of Resources** table, locate the queue named **NotificationQueue**. Note its **JNDI Name**.
   l. Click this queue. Then click the **Monitoring** tab.
   m. The **Messages Current** column should be 0, indicating that the store and forward agent is not functioning correctly.
3. Generate store and forward debug messages.
   a. **Lock** the console.
   b. Locate and edit the **MedRecSvr1** server.
   c. Click the **Debug** tab.
   d. Locate the **weblogic > jms > saf** debug scope:

☐	⊞ pauseresume	Disabled
☐	⊞ saf	Disabled

   e. Select the check box for the **saf** scope and click the **Enable** button.
   f. Click the **Logging > General** tab.
   g. Click the **Advanced** link.
   h. Edit the following values:

Field	Value
Minimum severity to log	Debug
Log file: Severity level	Debug
Standard out: Severity level	Debug
Memory buffer: Severity level	Debug

   Click **Save**.

i. **Activate** your changes.

j. Locate the shell running `MedRecSvr1`. Inspect the generated debug messages, which should be similar to the following:

```
<Debug> <JMSSAF> <BEA-000000> < subforwarder to
com.bea.medrec.emails.NotificationQueue failed to reconnect, due
to javax.naming.NameNotFoundException: While trying to lookup
'com.bea.medrec.emails.NotificationQueue' didn't find subcontext
'emails'. Resolved 'com.bea.medrec' [Root exception is
javax.naming.NameNotFoundException: While trying to lookup
'com.bea.medrec.emails.NotificationQueue' didn't find subcontext
'emails'. Resolved 'com.bea.medrec']; remaining name
'emails/NotificationQueue'>
```

Tip: Alternatively, you can **tail** the server log file (`tail –f MedRecSvr1.log`).

k. Can you find the discrepancy between the JNDI name of the `NotificationQueue` and the JNDI named used by the store and forward agent?

4. Repair the configuration typo.

a. **Lock** the console once again.

b. Locate and edit the JMS module named **MedRec-jms**.

c. In the **Summary of Resources** table, select **Svr2SAFDestinations**.

d. Click the **Configuration > Queues** tab.

e. Edit **Svr2NotificationQueue**.

f. Correct the **Remote JNDI Name**:

```
com.bea.medrec.email.NotificationQueue
```

Click **Save**.

g. **Activate** your changes.

h. Verify that the prior debug message is no longer generated.

i. Repeat the previous steps to monitor the **NotificationQueue** in the **Notifications-jms** module. Confirm that a message was received.

Tip: Messages Current is now 1 instead of 0.

5. Deactivate debug messages.

a. **Lock** the console.

b. Repeat the prior steps to update the **Logging** configuration of **MedRecSvr1**:

Field	Value
Minimum severity to log	Info
Log file: Severity level	Info
Standard out: Severity level	Warning
Memory buffer: Severity level	Info

c. Repeat the prior steps to update the server's **Debug** configuration. Disable **saf** debugging.

d. **Activate** your changes.

Practice Solution

No solution exists for this practice. You must complete all of the instructions.