

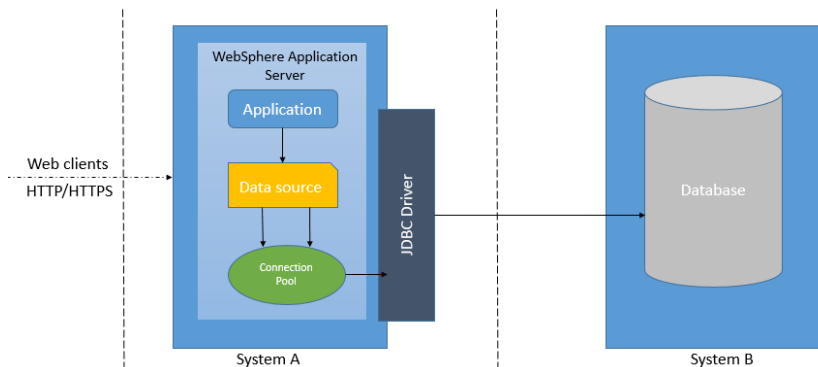
CHAPTER 7: CONNECT TO A DATABASE

Theory

Business applications running on WebSphere Application Server requires access to database systems. In order to access to databases, we need to define a data source for each database.

For better understanding of the tasks required to access databases, we need to understand following terms:

- **JDBC**, Java Database Connectivity, is a Java-based API technology to access databases. You can connect to a database, query and change data in a database. There are 2 types of JDBC drivers in WebSphere Application Server, version 2 driver (requires database client to connect to the database server) and version 4 driver (can directly connect to the database).
- **Data source**, is referred to the name of the configuration properties of the database in order to connect and run queries.



- **Connection pool**, is a configuration object that provides a set of connections to databases for the applications. When an application requires access to a database, it will use an existing connection from the pool and connection pool will create a new connection if there is no pooled connections available. You can set minimum and maximum number of connections for the pool to prevent overhead related with database connection requests.

- **JDBC Provider**, supplies the specific JDBC driver class to a specific database vendor. To create a data source, we need to associate a data source with the JDBC provider.
- **JNDI**, Java Naming Directory Interface, is a Java API that gives applications access to database connections.
- **J2C authentication alias**, is a feature that encrypts the password used by the adapter to access to a database.

In order to provide access to a database from an application that runs on WebSphere Application Server, you need to follow 2 basic steps:

0. Create and J2C authentication alias to store and encrypt credentials which will be used to connect to the database.
1. Create a JDBC provider that contains information of database drivers, type of access and location of the files needed for the implementation.
2. Create a data source that defines which JDBC driver to use, database name and location, and other connection properties.

AIM

In this lab exercise, you will enable access to applications from the WebSphere Application Server. In order to complete the exercise, you need to have following information beforehand:

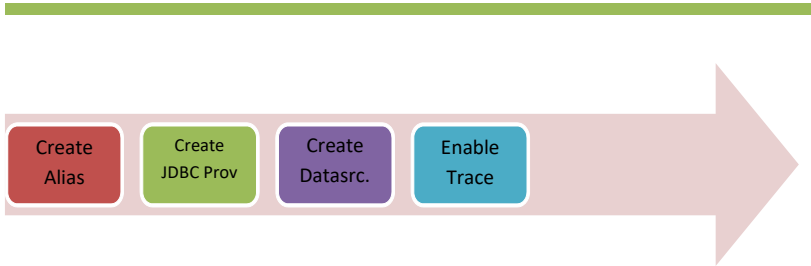
- A running database instance
- Hostname or IP address of the server where the database runs
- Port number to connect to the database
- Sample database name
- Username and password to connect to the database server and the database.

Note: If database is not available, still lab guide can be completed. Although, you won't be able to test connection with database at the end of this lab.

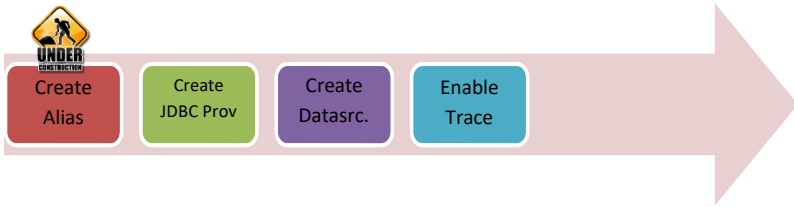
You need to follow the tasks below:

- Task 1: Create an authentication alias
- Task 2: Create JDBC provider
- Task 3: Create data source
- Task 4: Enable JDBC trace logs

Lab Exercise 7: CONNECT TO A DATABASE

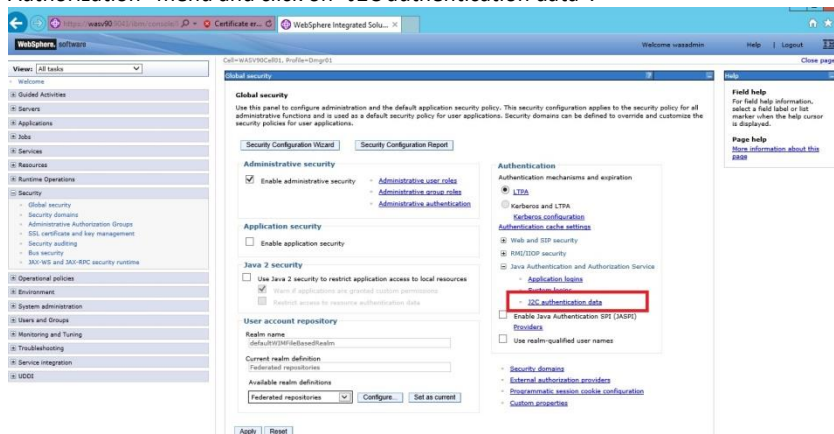


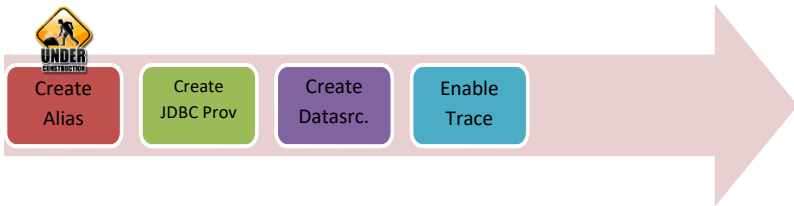
1. **Create an authentication alias**
2. **Create JDBC provider**
3. **Create data source**
4. **Enable JDBC trace logs**



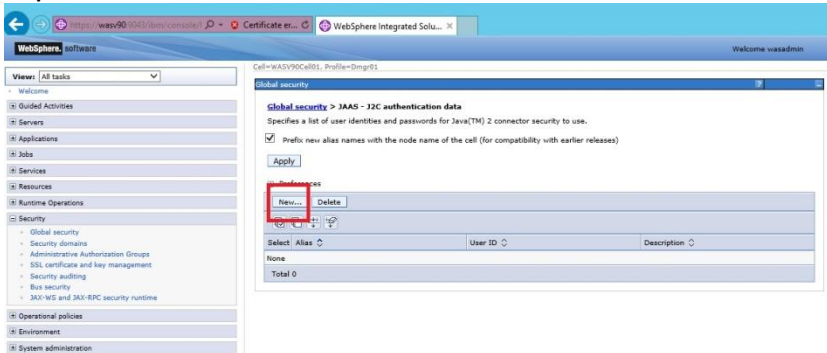
Task 1: Create an authentication alias

Step 1: Login to admin console and navigate to “Security>Global security”. Under the “Authentication” part, located on the right, expand “Java Authentication and Authorization” menu and click on “J2C authentication data”.

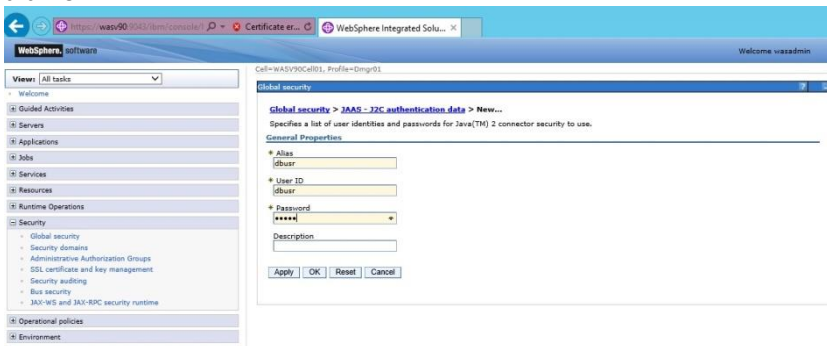


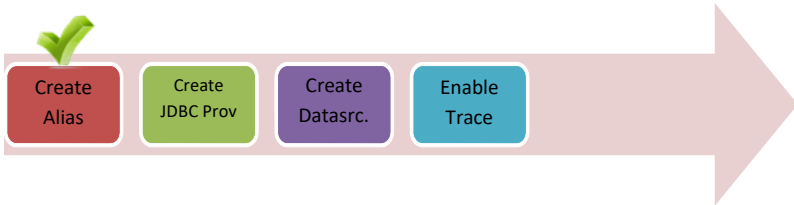


Step 2: Click on “New” to add a new authentication alias.

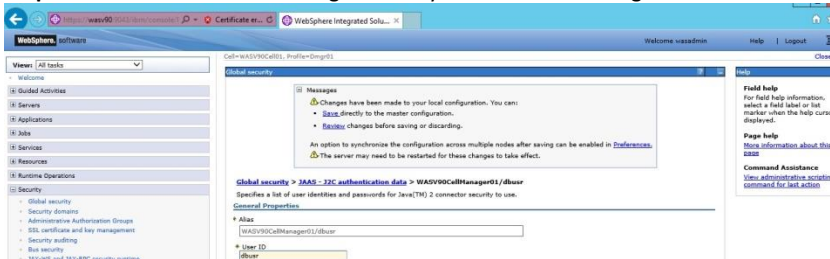


Step 3: Give an alias name and enter the credentials for database connection, then click “OK”.

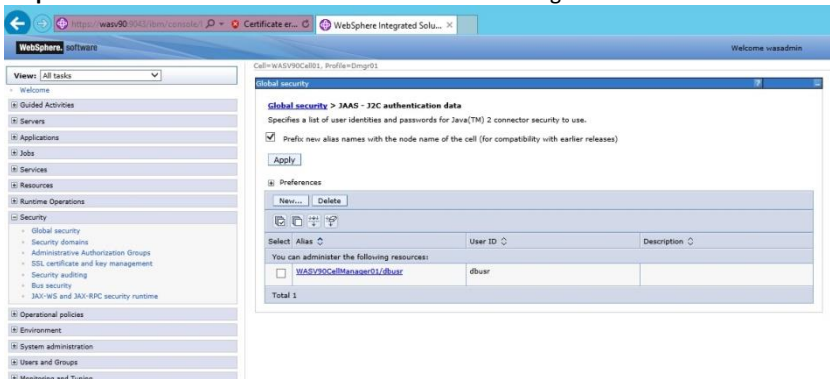




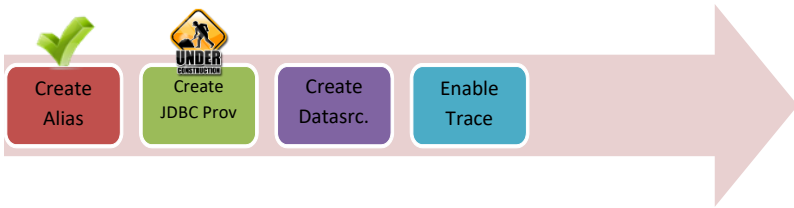
Step 4: Click “Save” to write changes directly to the master configuration.



Step 5: You should see the new alias listed as below image.



Task 1 is complete!



Task 2: Create JDBC provider

Step 1: Login to admin console and navigate to “Resources>JDBC>JDBC provides”.

JDBC providers

Use this page to edit properties of a JDBC provider. The JDBC provider object encapsulates the specific JDBC driver implementation class for access to the specific vendor database of your environment. Learn more about this task in a [guided activity](#). A guided activity provides a list of task steps and more general information about the topic.

Scopes: **All scopes**

☒ Show scope selection drop-down list with the all scopes option

Scope specifies the level at which the resource definition is visible. For detailed information on what scope it and how to create, see the [scope definition](#) page.

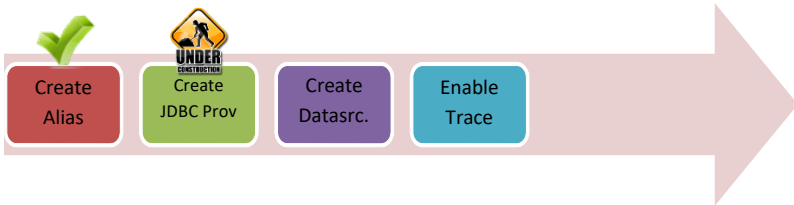
All scopes

Preferences

New... Delete

Select	Name	Scope	Description
<input type="checkbox"/>	Derby JDBC Provider	Node=WASV90Node1,Server=server1	Derby embedded non-XA JDBC Provider
<input type="checkbox"/>	QT3DataSource	Node=WASV90CallManager01	

Total 2



Step 2: You need to change the scope depending on your needs. For this example, we will use the cell as scope. Then click “New” to define new JDBC provider.

JDBC providers

Use this page to edit properties of a JDBC provider. The JDBC provider object encapsulates the specific JDBC driver implementation class for access to the specific vendor database of your environment. Learn more about this task in a [guided activity](#). A guided activity provides a lot of task steps and more general information about the topic.

☒ Scope: Cell=wasV90Cell01

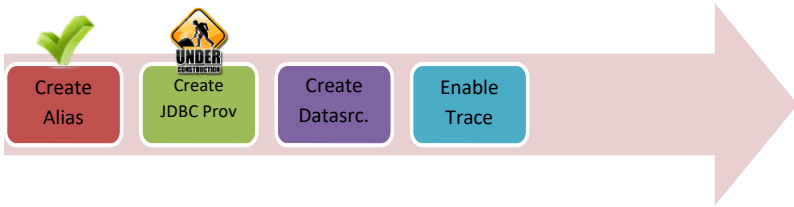
☒ Show scope selection drop-down list with the all scopes option

Scope specifies the level at which the resource definition is visible. For detailed information on what scope is and how it works, [see the scope selection help](#).

Cell=wasV90Cell01

New... **Delete**

Select	Name	Scope	Description
	None		
Total 0			



Step 3: In this step you need to configure following properties:

Database type: DB2

Provider type: DB2 Universal JDBC Driver Provider

Implementation type: Connection pool data source

Click “Next” to continue.

WebSphere Integrated Solutions console - Welcome wasadmin

Cell=WASV90Cell01_Profile=Dmgr01

Create a new JDBC Provider

Step 1: Create new JDBC provider

Step 2: Enter database class path information

Step 3: Summary

Create new JDBC provider

Set the basic configuration values of a JDBC provider, which encapsulates the specific vendor JDBC driver implementation classes that are required to access the database. The value of fields in the name and the description fields, but you can type different values.

Scope: cells\WASV90Cell01

Database type: DB2

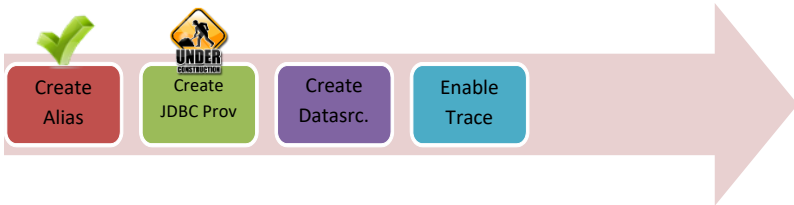
Provider type: DB2 Universal JDBC Driver Provider

Implementation type: Connection pool data source

Name: DB2 Universal JDBC Driver Provider

Description: One-phase commit DB2 JDBC provider that supports JDBC 3.0. Data sources that use this provider support only 1-phase commit processing, unless you use driver type 2 with the application server for x/OS. If you use the application server for x/OS, driver type 2 uses RRS and supports 2-phase commit processing.

Next Cancel



Step 4: You need to copy database drivers to the server where we have the deployment manager installed. As an example, we stored DB2 drivers under **“/headless/Desktop/websphere/jars”**. In this step, we need to configure the location of the drivers as shown below and click **“Next”**.

Create a new JDBC Provider

Step 1: Create new JDBC provider
 Step 2: Enter database class path information
 Step 3: Summary

Enter database class path information

Set the class path for the JDBC driver class files, which WebSphere® Application Server uses to define your JDBC provider. This wizard page displays a default list of jars and allows you to set the environment variables that define the directory locations of the files. Use complete directory paths when you type the JDBC driver file locations. For example, C:\SQLSRV\bin on Windows® or /home/db2inst1/bin on Linux™.

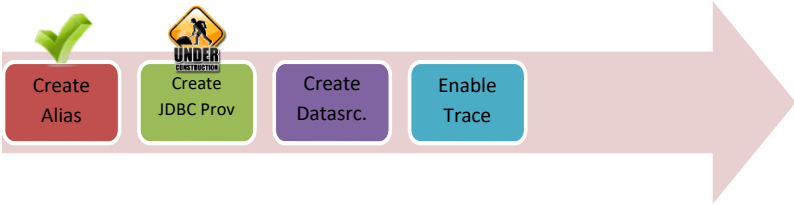
Entries are separated by using the ENTER key and must not contain path separator characters (such as \ or /). If a value is specified for you, you may click Next to accept the value.

Class path

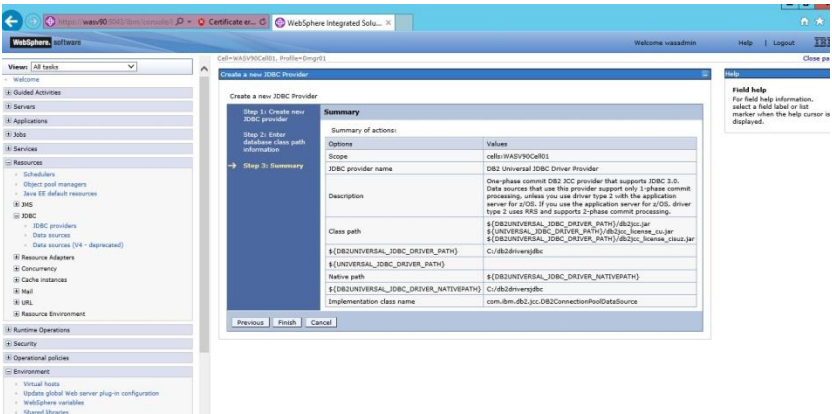
Directory location for "db2jar.jar; db2jar_license_ico.jar" which is saved as WebSphere variable \$[ORUNIVERSAL_JDBC_DRIVER_PATH]

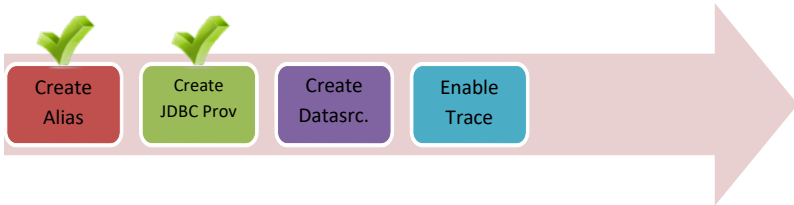
Native library path

Directory location which is saved as WebSphere variable \$[ORUNIVERSAL_JDBC_DRIVER_NATIVEPATH]

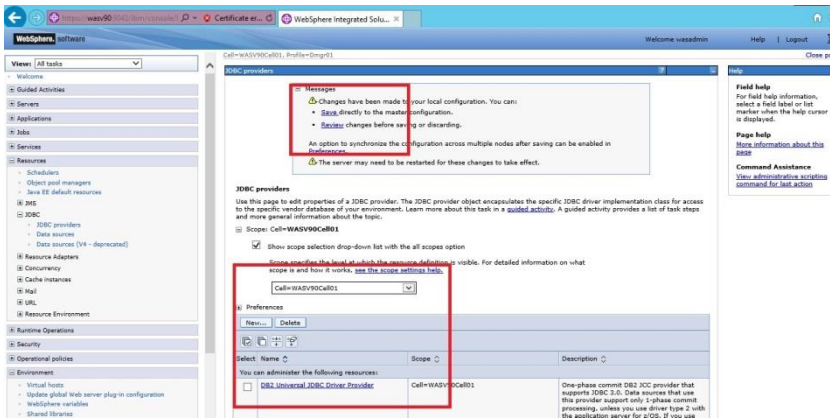


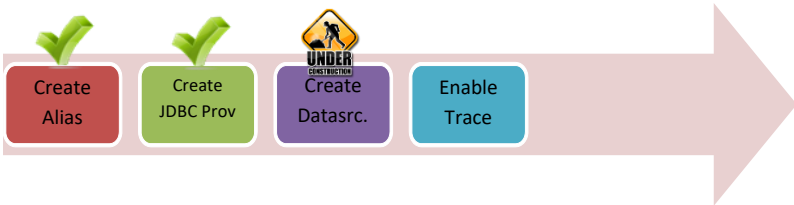
Step 5: Review the summary of options and then click “Finish”.





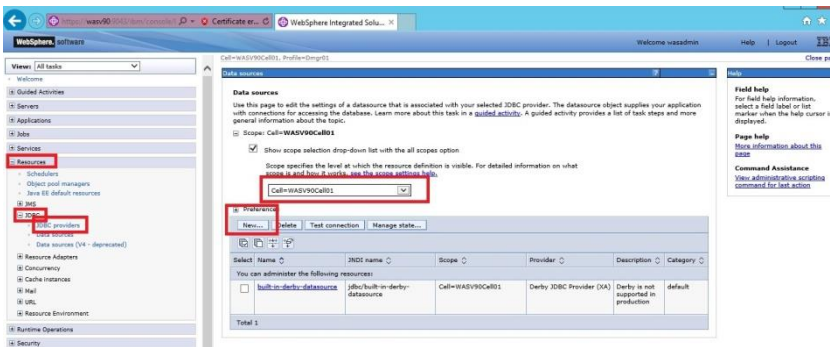
Step 6: Click “Save” to write the changes to master repository. You should see the newly created JDBC provider.

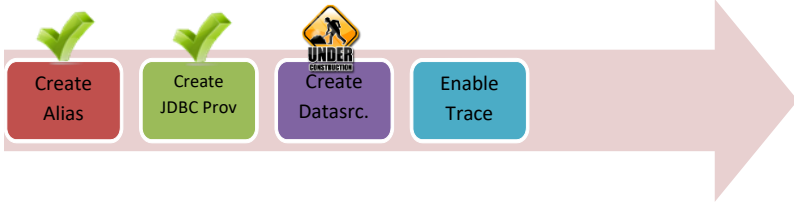




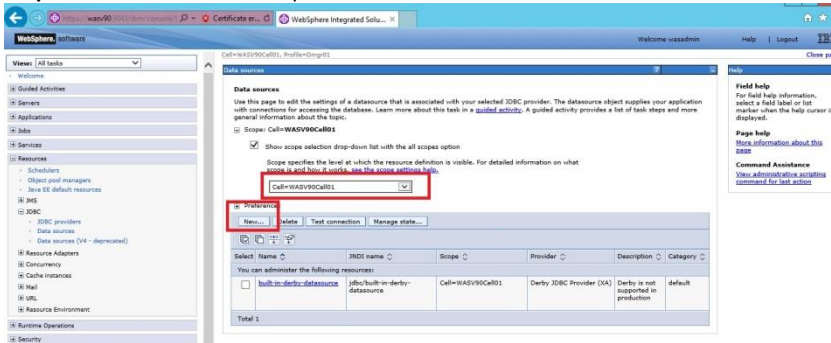
Task 3: Create data source

Step 1: Navigate to “Resources>JDBC>Data sources”. Change the scope according to your needs.

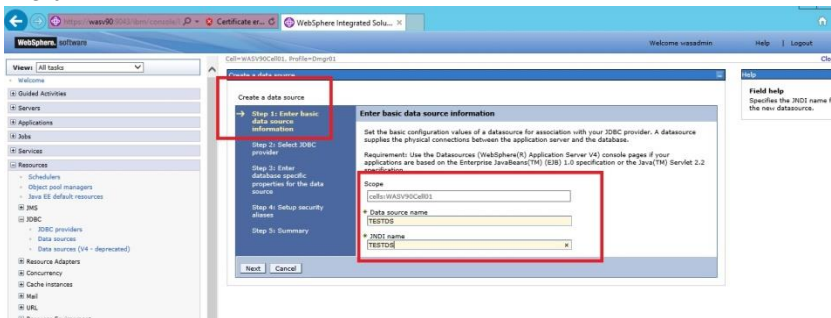


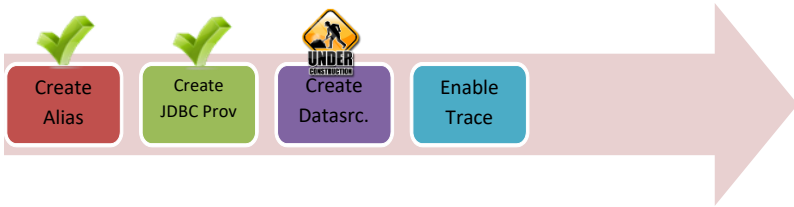


Step 2: We will use cell as scope and then click “New”.

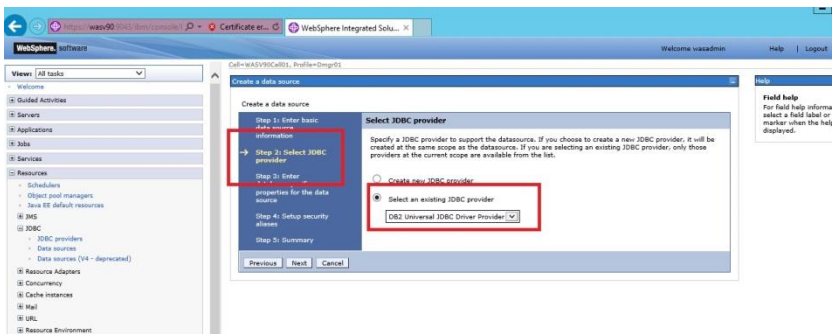


Step 3: Enter the data source (TESTDS) and JNDI (TESTDS) names and then click “Next”.

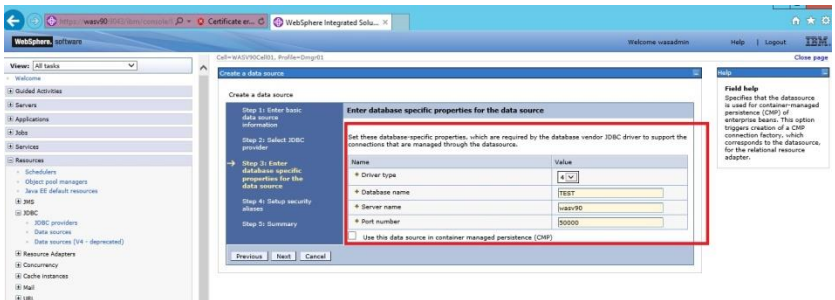


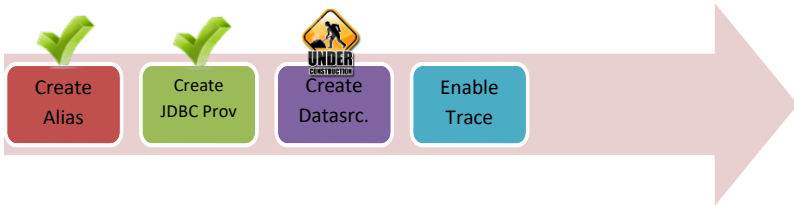


Step 4: Select “Select an existing JDBC Provider” and from the list sselect “DB2 Universal JDBC Driver Provider” then click “Next”.

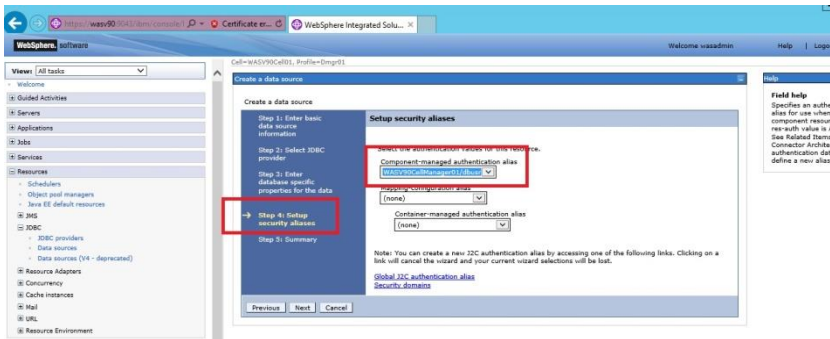


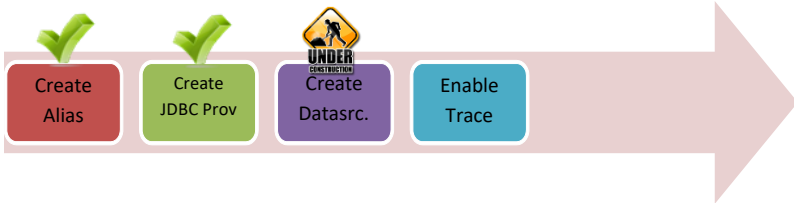
Step 5: You need to enter database properties (driver type should be 4, database name, database host, database port) and click “Next”.





Step 6: Select authentication alias we created in the first task for the “Component-managed authentication alias” and “Container-managed authentication alias”, then click “Next”.





Step 7: Review the summary and then click “Finish”.

WebSphere Integrated Solutions console

Create a data source

Summary

Options	Values
Scope	cell=WASV90Cell01
Data source name	TESTDS
JDBC name	TESTDS
Select an existing JDBC provider	DBZ Universal JDBC Driver Provider
Implementation class name	com.ibm.db2.jcc.DB2ConnectionPoolDataSource
Driver type	4
Database name	TEST
Server name	wasv90
Port number	50000
Use this data source in container managed persistence (CMP)	false
Component-managed authentication alias	WASV90CellManager01/dbvar
Mapping-configuration alias	(none)
Container-managed authentication alias	(none)

Previous **Finish** Cancel

Step 8: Click “Save” to write changes.

WebSphere Integrated Solutions console

Data sources

Messages

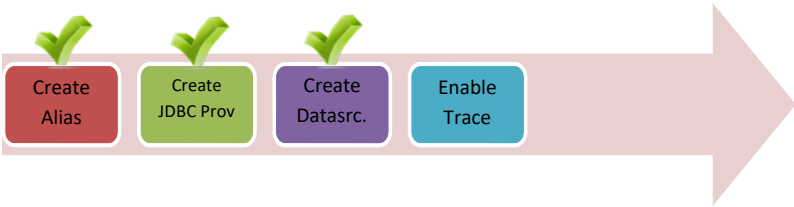
- Changes have been made to your local configuration. You can:
 - [Save](#) directly to the master configuration.
 - [Revert](#) changes before saving or discarding.
- An option to synchronize the configuration across multiple nodes after saving can be enabled in the **master** console.
- The server may need to be restarted for these changes to take effect.

Scope: Cell=WASV90Cell01

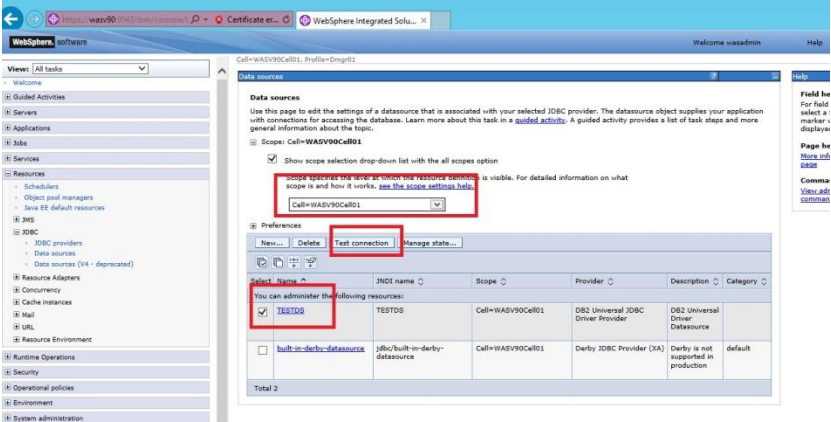
Show scope selection drop-down list with the all scope options visible. For detailed information on what scope is and how it works, [see the scope settings help](#).

Cell=WASV90Cell01

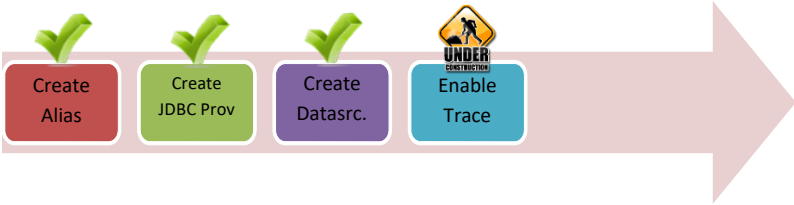
Select	Name	JDBC name	Scope	Provider	Description	Category
<input type="checkbox"/>	TESTDS	TESTDS	Cell=WASV90Cell01	DBZ Universal JDBC Driver Provider	DBZ Universal Driver Datasource	



Step 9: Select the data source recently added and click “Test connection”. You should have the success message for the connection test.

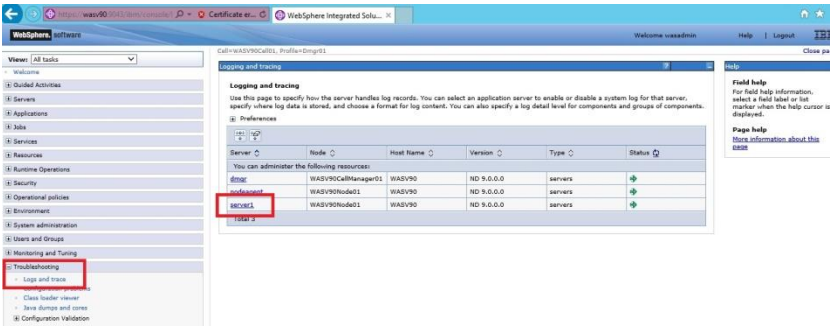


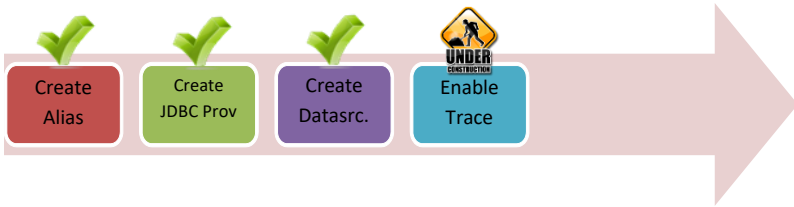
Task 3 is complete!



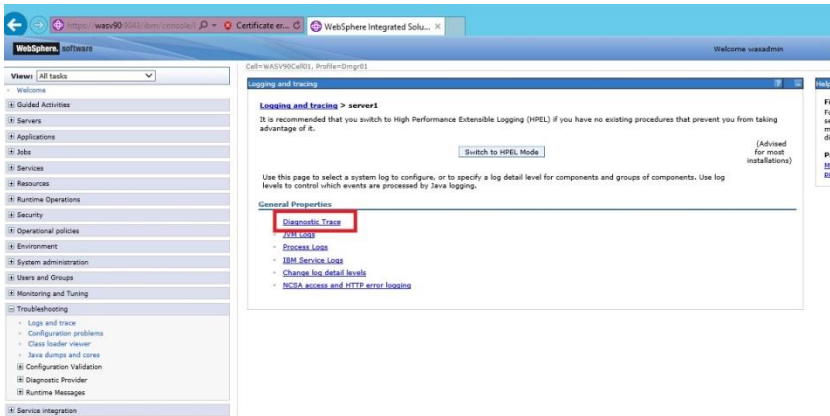
Task 4: Enable JDBC trace logs

Step 1: Navigate to “Troubleshooting>Logs and trace” and then click on “server1”.

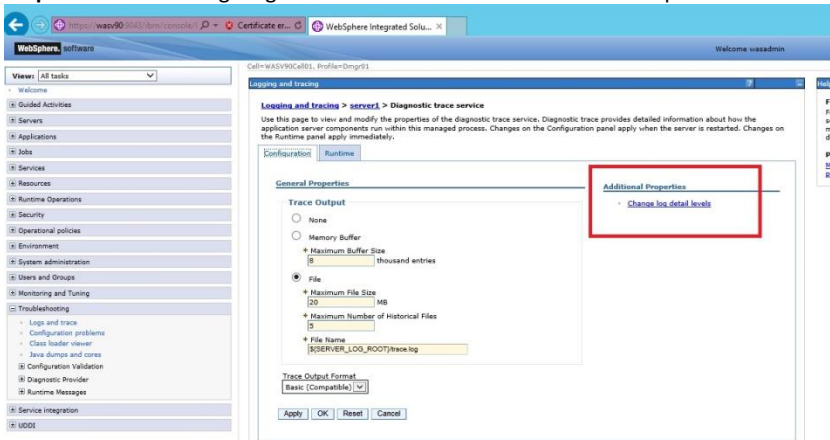


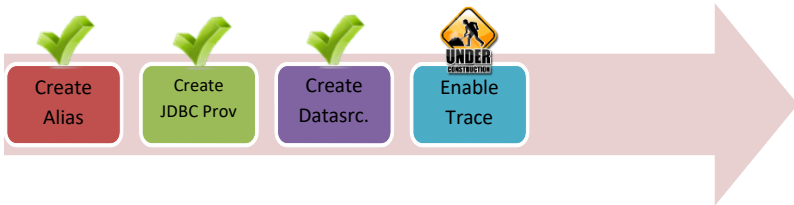


Step 2: Click on “Diagnostic Trace”.



Step 3: Click on “Change log detail levels” under the “Additional Properties”.





Step 4: In the “Change log detail levels”, put the following into the text box and then click “OK”.

**=info:*

WAS.j2c=all:

RRA=all:

Transaction=all

WebSphere Integrated Solutions console

Cell=ibmwas90, Profile=Dmgr01

Logging and tracing > server1 > Diagnostic trace service > Change log detail levels

Use log levels to control which events are processed by Java logging. Click Components to specify a log detail level for individual components, or click Groups to specify a log detail level for a predefined group of components. Click a component or group name to select a log detail level. Log detail levels are cumulative; a level near the top of the list includes all the subsequent levels.

Configuration Runtime

General Properties

☐ Save runtime changes to configuration as udl

Change log detail levels

☐ Disable logging and tracing of potentially sensitive data (WARNING: This might cause the log detail level setting to be modified when it is applied on the server.)

Select components and specify a log detail level. Log detail levels specified here will apply to the entire server. Expand log detail level for a predefined group of components. Click log detail level for individual components, or click Groups to specify a component or group name to select a log detail level. Log detail levels are cumulative.

RRA=all:
Transaction=all

☒ Components and Groups

Correlation

Enable log and trace correlation so entries that are serviced by more than one thread, process, or server will be identified as belonging to the same unit of work.

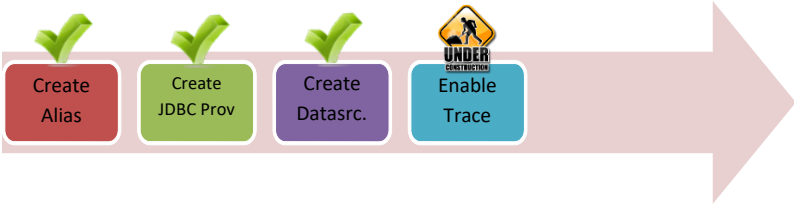
☐ Enable log and trace correlation

☒ Include request IDs in log and trace records

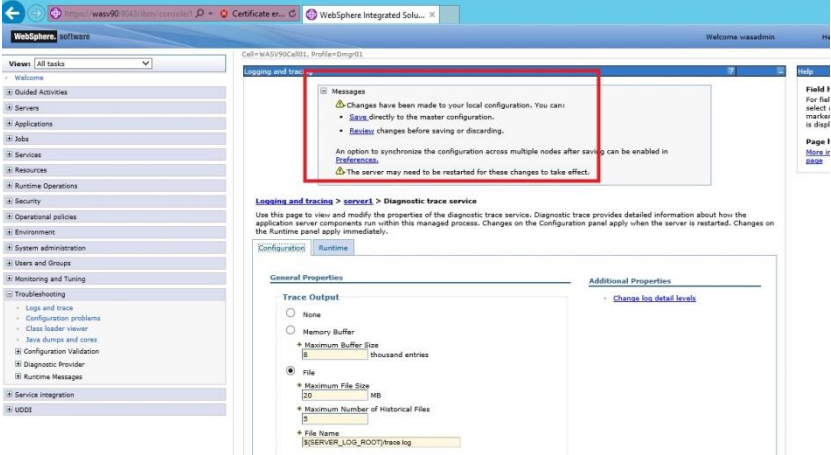
☐ Include request IDs in log and trace records and create correlation log records

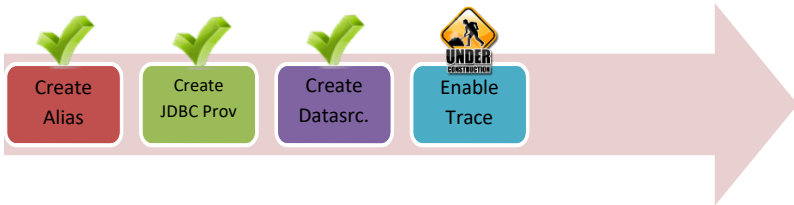
☐ Include request IDs in log and trace records; create correlation log records, and capture data snapshots

Apply OK Reset Cancel



Step 5: Click “Save” to write the changes to the master repository.

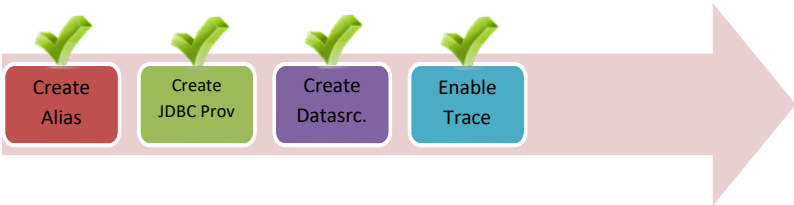




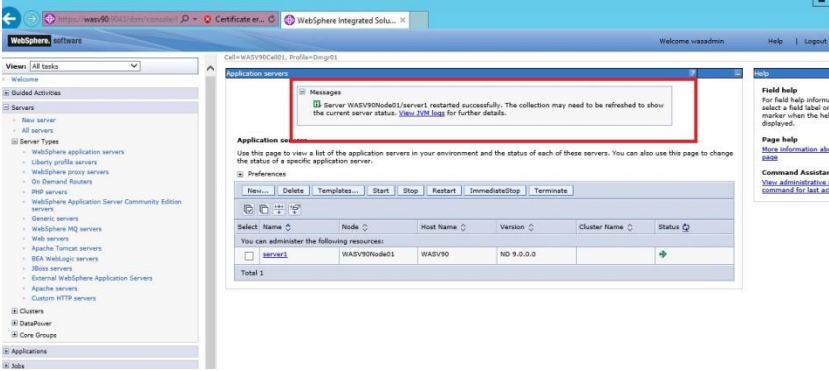
Step 6: Navigate to “Servers>Server Types>WebSphere application servers” and select the application server, then click “Restart”.

The screenshot shows the WebSphere Integrated Solutions console. The left sidebar displays the navigation tree with 'Servers' and 'Server Types' highlighted. The main area displays the 'Application servers' page, which includes a table of application servers and a 'Restart' button highlighted in red.

Select	Name	Node	Host Name	Version	Cluster Name	Status
<input checked="" type="checkbox"/>	WSV901	WASV90Node01	WASV90	ND 9.0.0.0		+
Total 1						



Step 9: When you see the success message, all the changes are effective.



Task 4 is complete!

SUMMARY

Business applications running on WebSphere Application Server requires access to database systems. In order to access to databases, we need to define a data source for each database. You need to create a JDBC provider that contains information of database drivers, type of access and location of the files needed for the implementation and to create a data source that defines which JDBC driver to use, database name and location, and other connection properties.

REFERENCES

- <http://publib.boulder.ibm.com/infocenter/dmndhelp/v6rxmx/index.jsp?topic=/com.ibm.wbit.sample.tech.2.doc/enterprisedisc/topics/tcrtalias.html>
- http://pic.dhe.ibm.com/infocenter/wasinfo/v7r0/index.jsp?topic=%2Fcom.ibm.websphere.nd.multiplatform.doc%2Finfo%2Fae%2Fae%2Ftdat_tccrtprovds.html
- http://publib.boulder.ibm.com/infocenter/wsd0c400/v6r0/index.jsp?topic=/com.ibm.websphere.iseries.doc/info/ae/ae/rtrb_jdbccomp.html

INDEX

Connection pool.....	207
Data source	207
J2C authentication alias	208
JDBC	207
JDBC provider	208
JDBC Provider.....	208
JNDI.....	208
version 4 driver	207

