

IBM WEBSHERE BUSINESS MONITOR V6.0.2 – LAB EXERCISE

**Monitoring using the sample DB2<sup>®</sup> Emitter**

What this exercise is about ..... 2

Lab requirements ..... 2

What you should be able to do ..... 2

Introduction ..... 3

Exercise instructions ..... 4

Part 1: Modify code to handle CustomerOrder event using the CEI API SDK and export the DBEmitter application (EAR) ..... 5

Part 2: Create a DB2 emitter database ..... 20

Part 3: Configure data source for DB2 emitter database ..... 25

Part 4: Configure a scheduler service and deploy the DB2Emitter EAR ..... 28

Part 5: Testing DB2 emitter ..... 36

Part 6: Solution ..... 40

What you did in this exercise ..... 42

## What this exercise is about

This exercise demonstrates using IBM WebSphere® Integration Developer to update the DB2® Emitter source code to customize it to emit specific user defined events using the CEI SDK that will be monitored using WebSphere Business Monitor. You will customize the supplied DB2 emitter source code, export the EAR, and deploy the EAR to WebSphere Process Server V6.0.2. You will configure WebSphere Process Server V6.0.2 with a Data Source for the new JDBC Provider, and a scheduler. Finally you will conduct a simple test to see if the events are being emitted.

## Lab requirements

List of system and software required for the student to complete the lab.

- WebSphere Integration Developer V6.0.2
- WebSphere Process Server V6.0.2 or WebSphere Application Server V6.1 with CEI API SDK

## What you should be able to do

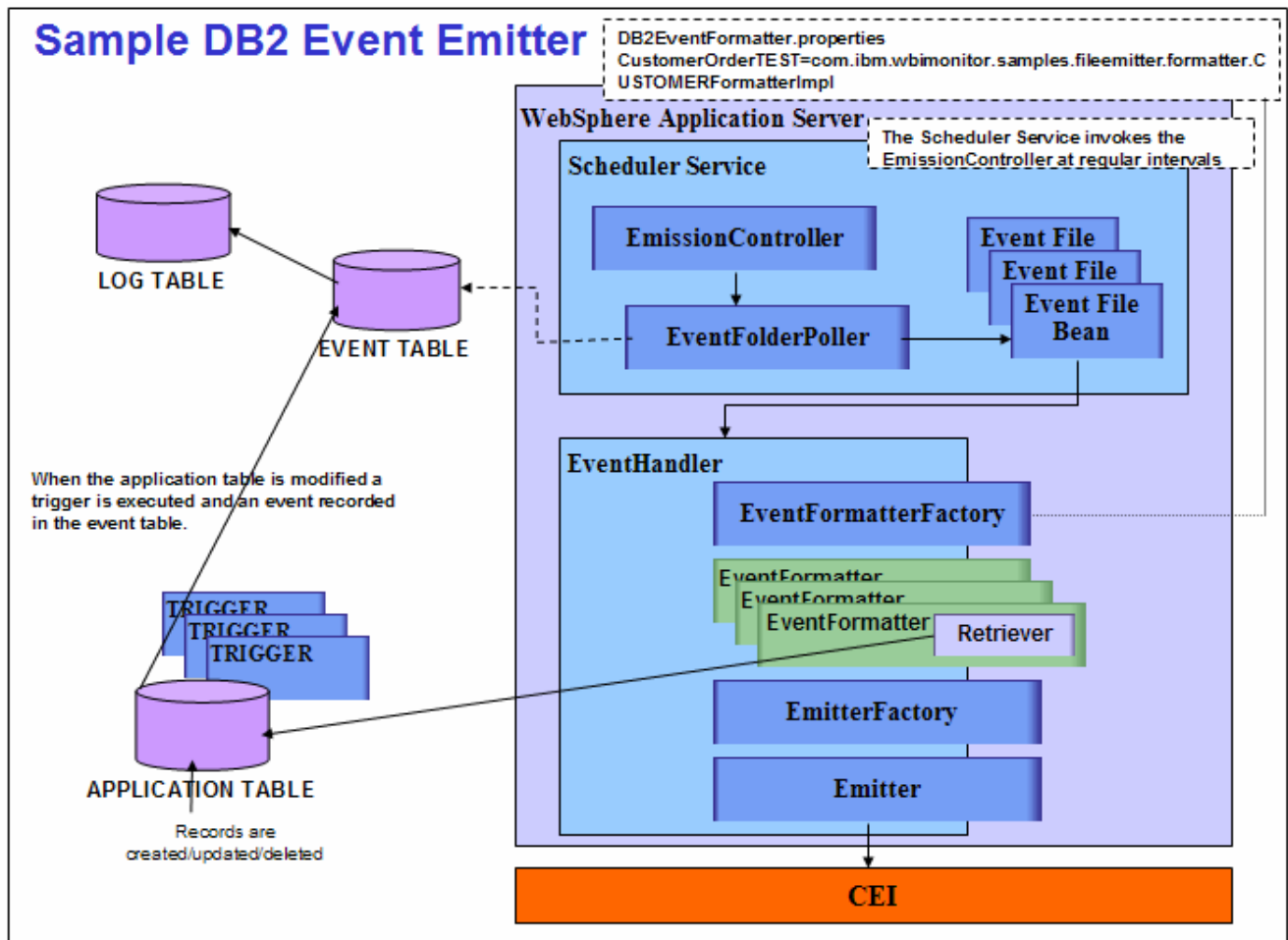
At the end of this lab you should be able to:

- Use WebSphere Integration Developer to import the DB2 Emitter source with CEI SDK emitter transport and perform necessary modifications to it, build the DB2 Emitter project and finally export an EAR file for deployment
  - Use the DB2 Command Line Window to create a new database which will be accessed by the DB2 Emitter Application, run the DDL scripts to create the necessary tables and triggers.
  - Use the WebSphere Process Server administrative console to configure the JDBC Provider, a Data source, a cloudscape database for the scheduler and finally the Scheduler itself
  - Use the WebSphere Process Server administrative console to install the EAR
  - Run a simple test to view successful events emitted
  - View the events emitted using the Common Event Browser (CBE Browser)
-

## Introduction

The Sample DB2 Event Emitter is a sample program written in Java™ that demonstrates how an enterprise information system (EIS) resource (an IBM DB2 database in this sample) which stores data pertaining to the state of a business can be instrumented to contribute to the overall monitoring of the activities of a business.

The main goal of the Sample DB2 Event Emitter is to introduce the use of the libraries and APIs provided by the Common Event Infrastructure (CEI) to generate and emit business events in the form of Common Base Events (CBEs). Common Base Events are the data packaging and format used by the WebSphere Business Monitor (WBM) Server to propagate business events.



## Exercise instructions

Some instructions in this lab may be Windows® operating-system specific. If you plan on running the lab on an operating-system other than Windows, you will need to run the appropriate commands, and use appropriate files (.sh vs. .bat) for your operating system. The directory locations are specified in the lab instructions using symbolic references, as follows:

Reference variable	Windows location	AIX®/UNIX® location
<WID_HOME>	C:\WID602	
<WPS_HOME>	C:\IBM\WebSphere\ProcServer	
<WPS_PROFILE_DIR>	<WPS_HOME>\profiles\<profile_name>	
<LABFILES>	C:\Labfiles602	/tmp/Labfiles602

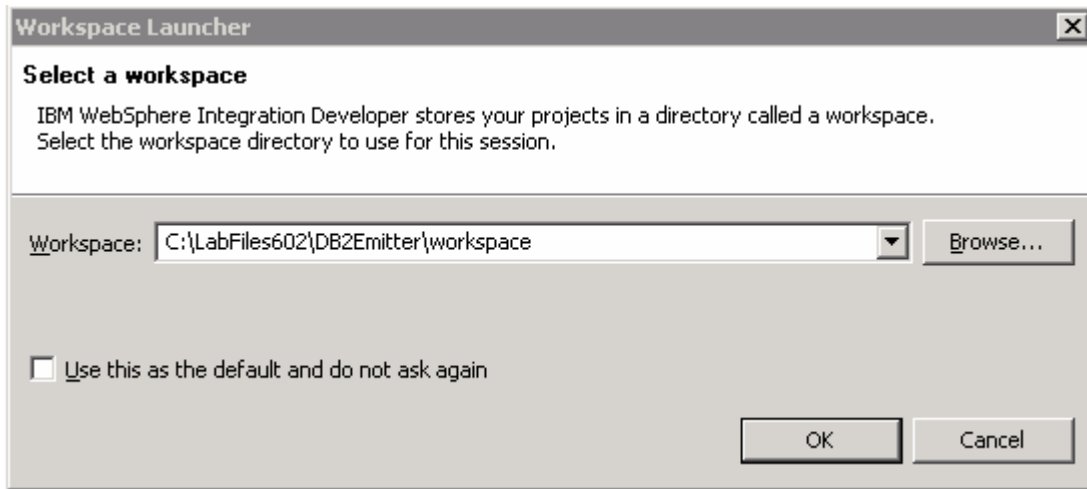
**Important:** This LAB does not cover the **Customer Order** Monitor Model deployment to the WebSphere Business Monitor Server. You should have first hand experience deploying and running a Monitor Model on the Monitor Server.

**Important:** You can either use WebSphere Process Server V6.0.2 (at times designated as Process Server) or a WebSphere Application Server V6.1 with CEI API SDK installed. This LAB uses WebSphere Process Server V6.0.2.


**Important:** The source code modification is a demonstration on how the events emitted will be correlated with the Monitor Model. Each event to be consumed by WebSphere Business Monitor needs to be matched to the Monitor Model.

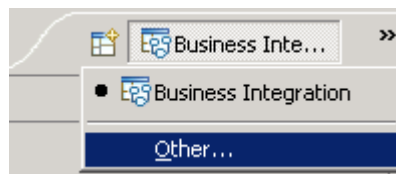
## Part 1: Modify code to handle CustomerOrder event using the CEI API SDK and export the DBEmitter application (EAR)

- \_\_\_ 1. Start WebSphere Integration Developer, creating a new workspace in the folder **C:\LabFiles602\DB2Emitter\workspace**, and turn off the auto-build feature
  - \_\_\_ a. The workspace Launcher window will be displayed. Click the **Browse...** button and select your workspace directory.

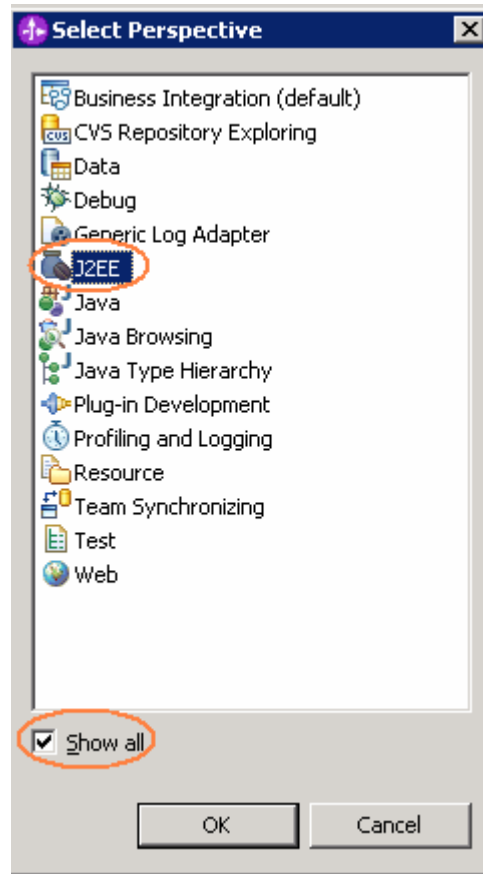


- \_\_\_ b. Click **OK**
  - \_\_\_ c. Close the welcome window by clicking the arrow in the top right corner of the welcome window
- \_\_\_ 2. Switch to the J2EE perspective and turn off automatic builds

- \_\_\_ a. By default WebSphere Integration Developer opens in **Business Integration** perspective. You need to change it to **J2EE** perspective. To do this click  on the top right corner of the WebSphere Integration Developer and choose **Other** if **J2EE** is not listed here.



- \_\_\_ b. From the **Select Perspective** dialog select **J2EE** and click **OK**

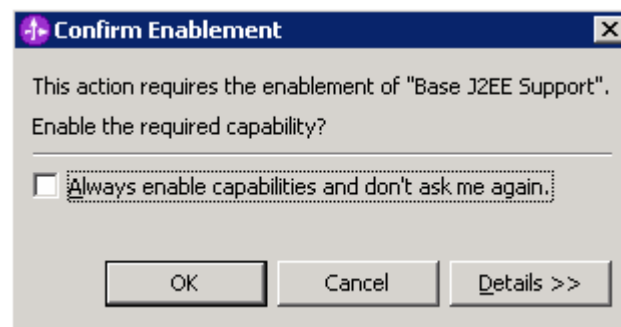


---

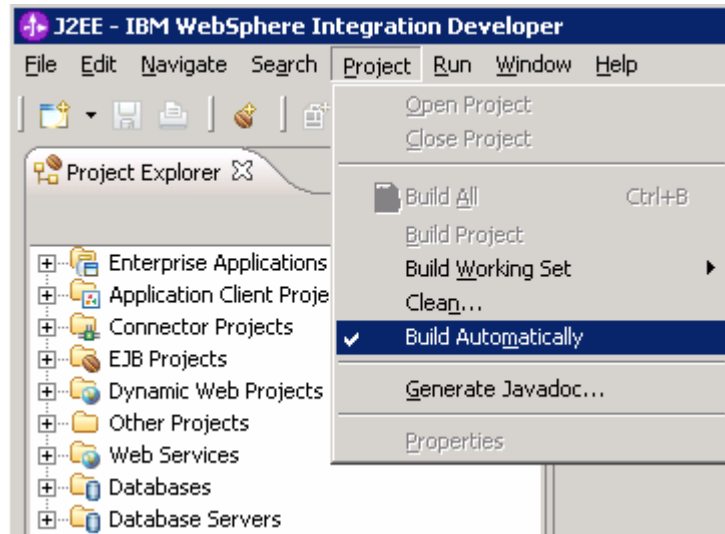
**Note:** Select the **Show all** check box if **J2EE** perspective does not show up.

---

\_\_ c. Click **OK** on the confirm enablement warning.



\_\_ d. Select **Project > Build Automatically**, to turn off automatic builds.



### 3. Import the **DB2Emitter** Projects

In this step, import all DB2 Emitter projects; make necessary build configurations based on the intended emitter transport (CEI API). Also import the necessary jars to support CEI API. Following is a description of the projects that will be imported (part of the Project Interchange **DB2Emitter.zip**):

- **DBEmitter**: Project to package artifacts into EAR
- **EmitterFW**: Common emitter framework source
- **CEIEmitter** : CEI emitter source
- **DBEmitterEJB**: DB2 Emitter specific source
- **DBEmitterImpl**: DB2 Emitter specific source. Implementation classes specific to retrieving and formatting events

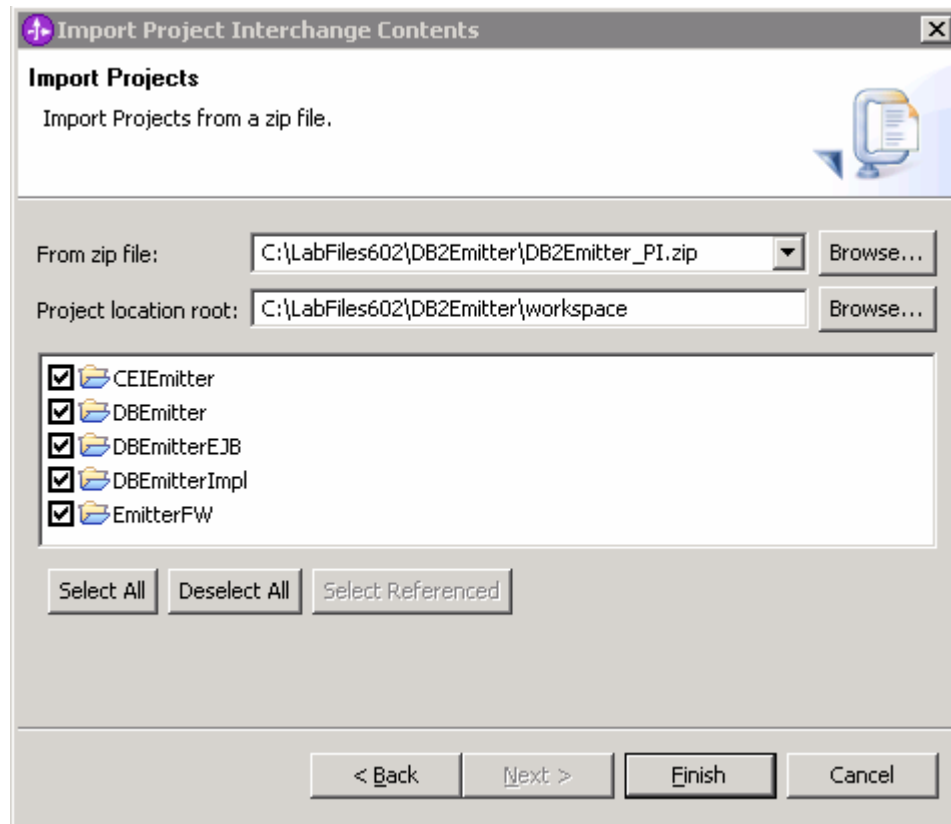
\_\_\_ a. Select **File > Import** from the main menu.

\_\_\_ b. From the **Import** window, select **Project Interchange** as the source and click **Next**

\_\_\_ c. The **Import Project Interchange Contents** window will be opened.

\_\_\_ d. Now click the first **Browse...** button and select **<LABFILES>\DB2Emitter\DB2Emitter\_PI.zip** as the source .zip file

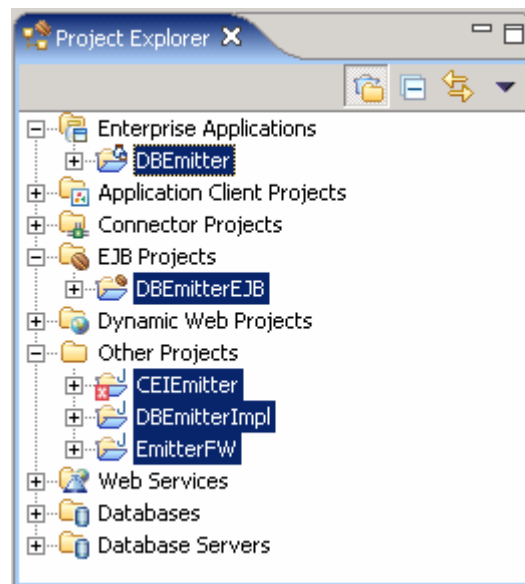
**For example:** C:\LabFiles602\DB2Emitter\DB2Emitter.zip



\_\_\_ e. Select all the projects and then click **Finish**

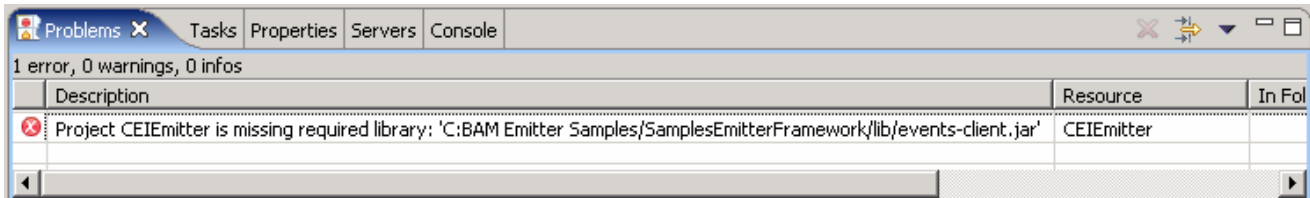
\_\_\_ 4. Review the imported artifacts

\_\_\_ a. Expand **Enterprise Applications, EJB Projects** and **Other Projects** to view the projects as shown below



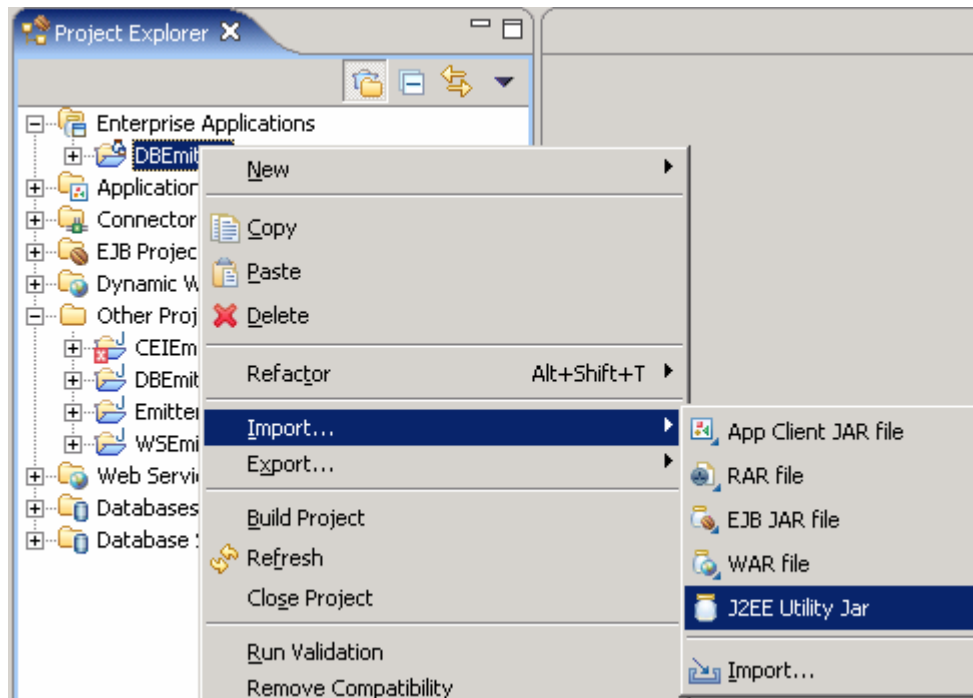
\_\_\_ b. At this time there will be some errors noticed. Click on the **Problems** tab to view the problems as shown below



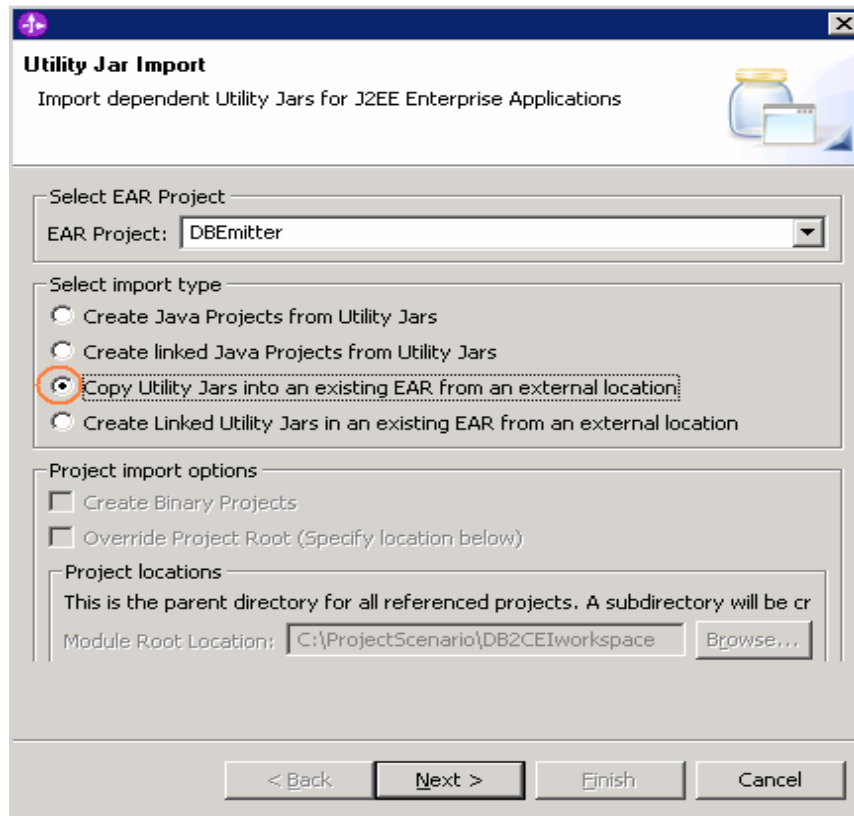


5. Working towards avoiding the build path errors

- a. It clearly states that the CEIEmitter project is missing a required library **events-client.jar**. Import it as J2EE Utility JAR by right-clicking on the **DBEmitter** project from the project explorer as shown below

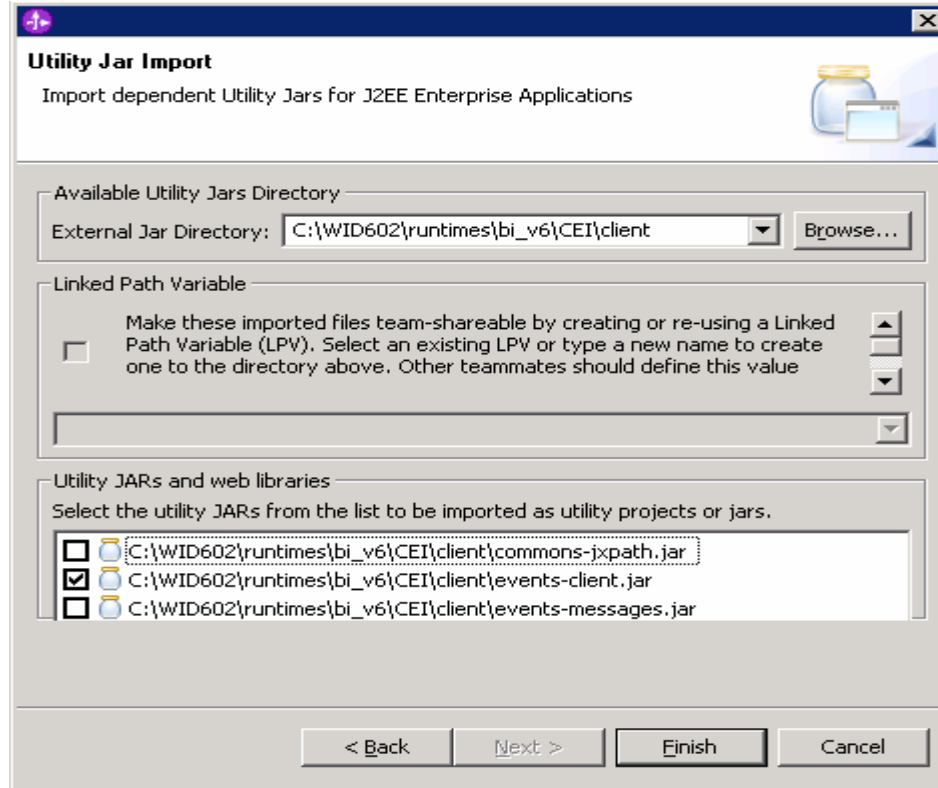


- b. In the **Utility Jar Import** dialog select the type of Import as **Copy Utility Jars into an existing EAR from an external location** as shown below



\_\_\_ c. Click **Next**

\_\_\_ d. In the next dialog click **Browse** to locate the **events-client.jar** and make sure you select the jar as shown below



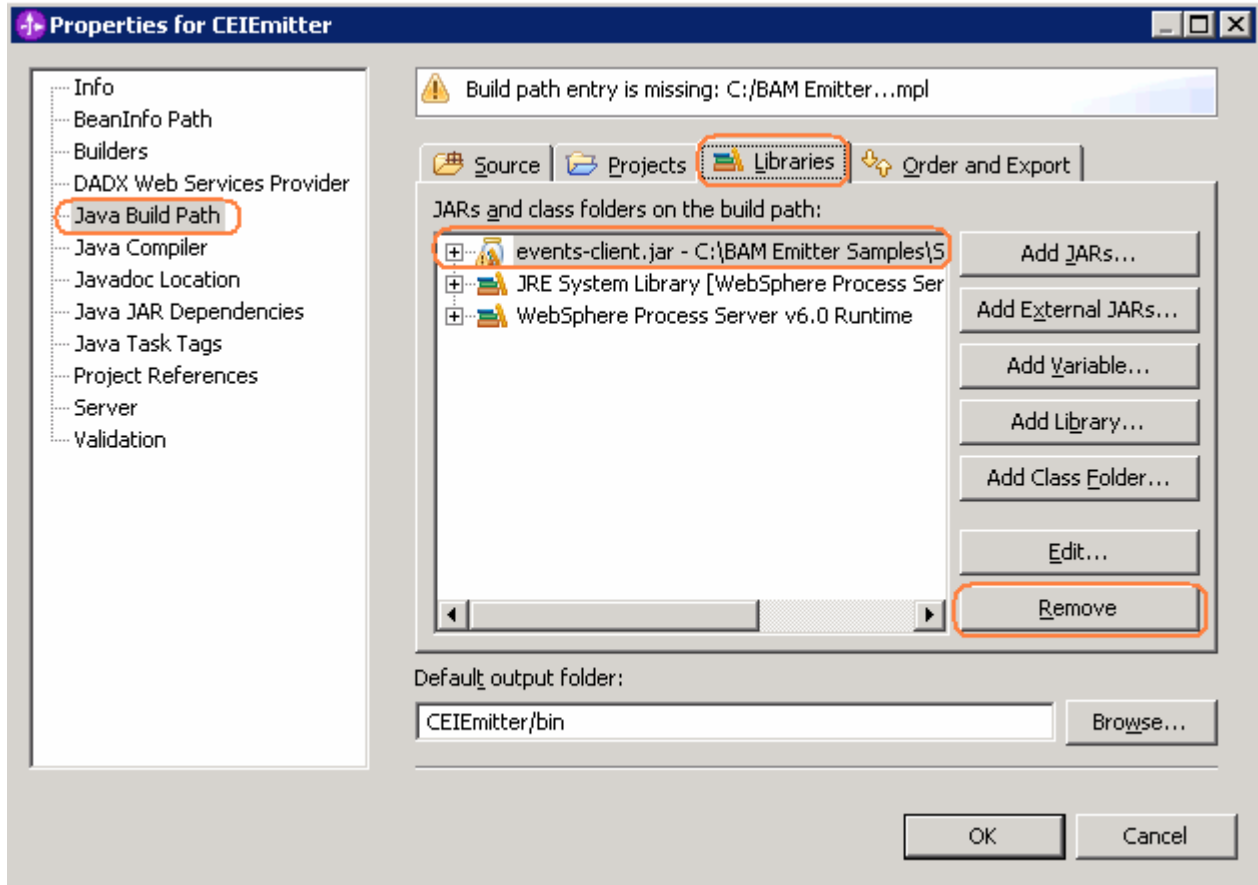
**Note:** You can find these libraries in WebSphere Process Server or WebSphere Integration Developer as follows:

<WID\_HOME>\runtimes\bi\_V6\CEI\client or

<WPS\_HOME>\CEI\client

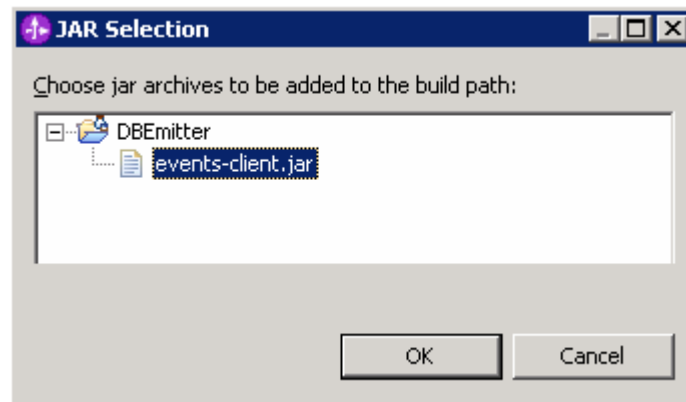
If CEI SDK is not installed on your machine local to the WebSphere Integration Developer, copy the necessary libraries from a remote WebSphere Process Server installation.

- \_\_\_ e. Click **Finish**
- \_\_\_ f. In the Project Explorer, right-click on the CEIEmitter project under **Other Projects** and select properties
- \_\_\_ g. In the **Properties for CEIEmitter** window, select Java Build Path in the left frame, select Libraries tab in the right frame



\_\_\_ h. Select the **events-client.jar** from the **JARs** listed and click the **Remove** button to remove build path.

\_\_\_ i. Now click the **Add JARs** button and select the **events-client.jar** (DBEmitter → events-client.jar)



\_\_\_ j. Click **OK** over the JAR Selection pop-up window

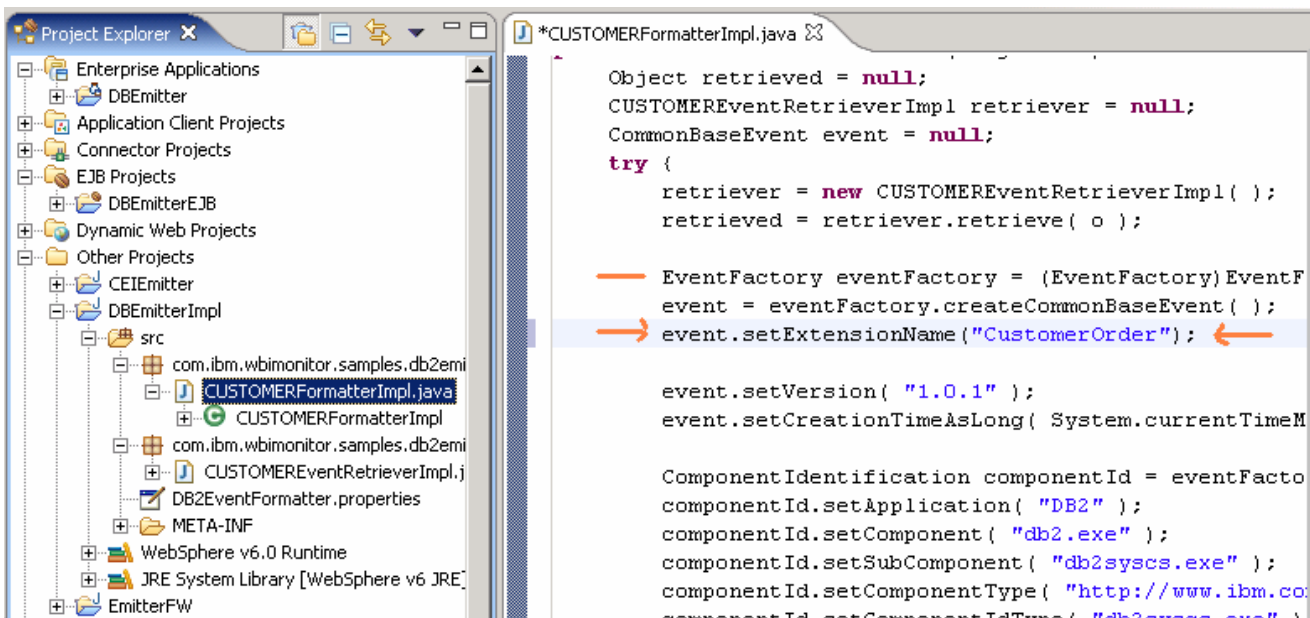
\_\_\_ k. Now click **OK** over the **Properties for CEIEmitter** window

\_\_\_ l. There must not be any errors reflected in the Problems tab at this time

6. Now modify the DB2 Emitter specific code. The sample code that is provided is based on a CUSTOMER event to show you how to modify it to handle a different event type, CustomerOrder. You will update the **CUSTOMERFormatterImpl.java** which implements the EventFormatter class and the **CUSTOMEREventRetrieverImpl.java** which extends **AbstractEventRetriever**.

**Note:** Two text files named **CUSTOMEREventRetrieverImpl.txt** and **CUSTOMERFormatterImpl.txt** with lines of code that are to be modified are under <LABFILES>\DB2Emitter\src. It is easy to copy these lines of code and paste at the appropriate location.

- a. In the Project Explorer expand **Other Projects > DBEmitterImpl > src > com.ibm.wbimonitor.samples.db2emitter.formatter** and double click on **CUSTOMERFormatterImpl.java**
- b. Set an extension name for the CommonBaseEvent that is being returned. Add the line **event.setExtensionName("CustomerOrder");** after the Event factory is created.



- c. Search for the following code segment

```

ExtendedDataElement id = event.addExtendedDataElement( "ID" );
id.setValuesAsInt( Integer.parseInt( (String)retrieved ) );
  
```

**And replace them with the following**

```

ExtendedDataElement ordernumber = event.addExtendedDataElement( "OrderNumber" );
ordernumber.setValuesAsInt( Integer.parseInt( (String)retrieved ) );
  
```

- d. Search for the following code segment

```

ExtendedDataElement id = event.addExtendedDataElement("ID");
id.setValuesAsInt(((ResultSet) retrieved).getInt("ID"));

ExtendedDataElement name = event.addExtendedDataElement("NAME");
  
```

```
name.setValuesAsString(((ResultSet) retrieved).getString("NAME"));
ExtendedDataElement address = event.addExtendedDataElement("ADDRESS");
address.setValuesAsString(((ResultSet) retrieved).getString("ADDRESS"));
ExtendedDataElement tel = event.addExtendedDataElement("TEL");
tel.setValuesAsString(((ResultSet) retrieved).getString("TEL"));
ExtendedDataElement email = event.addExtendedDataElement("EMAIL");
email.setValuesAsString(((ResultSet) retrieved).getString("EMAIL"));
```

**And replace them with the following**

```
ExtendedDataElement ordernumber = event.addExtendedDataElement("OrderNumber");
ordernumber.setValuesAsInt(((ResultSet) retrieved).getInt("OrderNumber"));
ExtendedDataElement customername = event.addExtendedDataElement("CustomerName");
customername.setValuesAsString(((ResultSet) retrieved).getString("CustomerName"));
ExtendedDataElement country = event.addExtendedDataElement("Country");
country.setValuesAsString(((ResultSet) retrieved).getString("Country"));
ExtendedDataElement city = event.addExtendedDataElement("City");
city.setValuesAsString(((ResultSet) retrieved).getString("City"));
ExtendedDataElement productnumber = event.addExtendedDataElement("ProductNumber");
productnumber.setValuesAsString(((ResultSet) retrieved).getString("ProductNumber"));
ExtendedDataElement quantity= event.addExtendedDataElement("Quantity");
quantity.setValuesAsInt(((ResultSet) retrieved).getInt("Quantity"));
ExtendedDataElement orderprice= event.addExtendedDataElement("OrderPrice");
orderprice.setValuesAsFloat(((ResultSet) retrieved).getFloat("OrderPrice"));
```

```

if( retrieved instanceof String ) {
    ExtendedDataElement id = event.addExtendedDataElement( "OrderNumber" );
    id.setValuesAsInt( Integer.parseInt( (String)retrieved ) );
}
else {
    ExtendedDataElement ordernumber = event.addExtendedDataElement( "OrderNumber" );
    ordernumber.setValuesAsInt( ((ResultSet) retrieved).getInt( "OrderNumber" ) );
    ExtendedDataElement customername = event.addExtendedDataElement( "CustomerName" );
    customername.setValuesAsString( ((ResultSet) retrieved).getString( "CustomerName" ) );
    ExtendedDataElement country = event.addExtendedDataElement( "Country" );
    country.setValuesAsString( ((ResultSet) retrieved).getString( "Country" ) );
    ExtendedDataElement city = event.addExtendedDataElement( "City" );
    city.setValuesAsString( ((ResultSet) retrieved).getString( "City" ) );
    ExtendedDataElement productnumber = event.addExtendedDataElement( "ProductNumber" );
    productnumber.setValuesAsString( ((ResultSet) retrieved).getString( "ProductNumber" ) );
    ExtendedDataElement quantity= event.addExtendedDataElement( "Quantity" );
    quantity.setValuesAsInt( ((ResultSet) retrieved).getInt( "Quantity" ) );
    ExtendedDataElement orderprice= event.addExtendedDataElement( "OrderPrice" );
    orderprice.setValuesAsFloat( ((ResultSet) retrieved).getFloat( "OrderPrice" ) );
}

```

\_\_ e. Save (**Ctrl+S**) and close the editor

\_\_ f. In the Project Explorer expand **Other Projects > DBEmitterImpl > src > com.ibm.wbimonitor.samples.db2emitter.retriever** and double click on **CUSTOMEREventRetrieverImpl.java**

\_\_ g. Search for the following code segment

```
private static String CREATE_EVENT_SQL = "SELECT * FROM CUSTOMER WHERE ID = ?";
```

```
private static String UPDATE_EVENT_SQL = "SELECT * FROM CUSTOMER WHERE ID = ?";
```

**And replace them with the following**

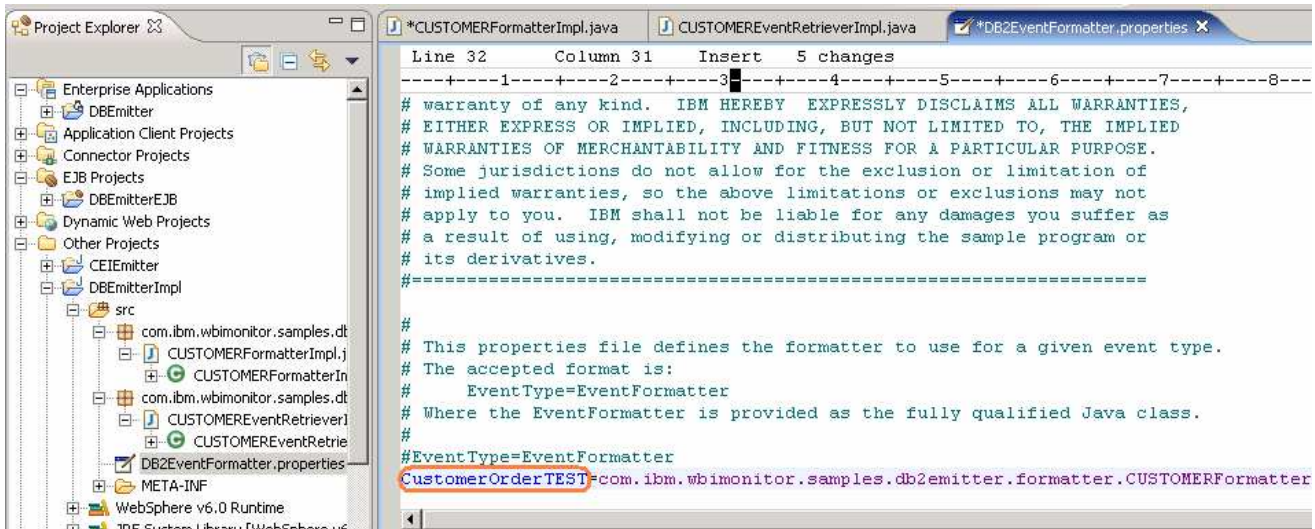
```
private static String CREATE_EVENT_SQL = "SELECT * FROM CUSTOMERORDER WHERE ORDERNUMBER = ?";
```

```
private static String UPDATE_EVENT_SQL = "SELECT * FROM CUSTOMERORDER WHERE ORDERNUMBER = ?";
```

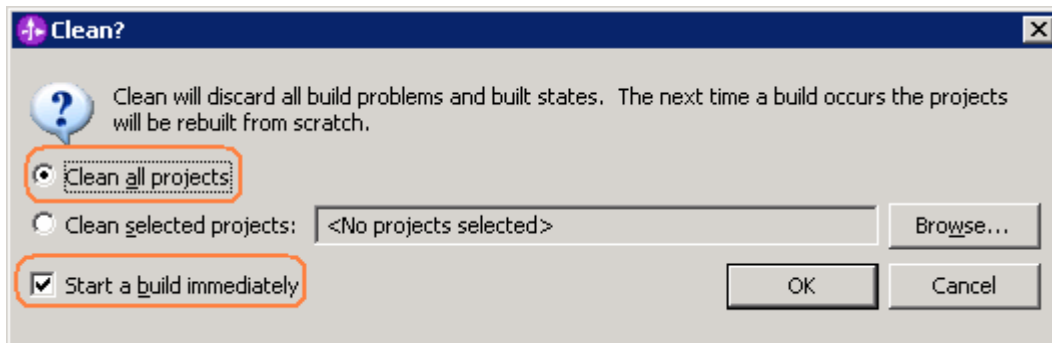
\_\_ h. Save (**Ctrl+S**) and close the editor

\_\_ i. Modify **DB2EventFormatter.properties**. In the Project Explorer expand **Other Projects > DBEmitterImpl > src** and double click to open **DB2EventFormatter.properties**

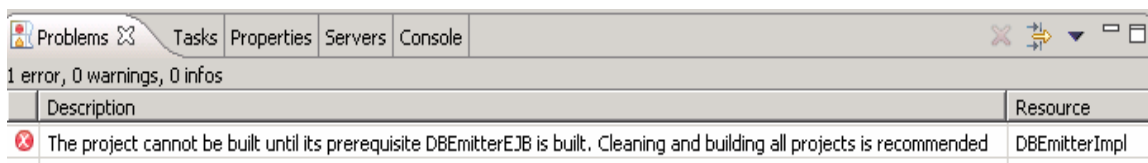
\_\_ j. The relationship between an event type and the EventFormatter class for the type is specified in this property file so that the Event Formatter registers to the Event Type **CustomerOrderTEST**. For example specify the event type as **CustomerOrderTEST** to be more specific to this scenario



- \_\_\_ k. Modify the event type from **CUSTOMER** to **CustomerOrderTEST** and leave the EventFormatter class  
(com.ibm.wbimonitor.samples.db2emitter.formatter.CUSTOMERFormatterImpl) as it is.
- \_\_\_ l. Save (**Ctrl+S**) and close the file
- \_\_\_ m. If workspace errors are not removed, it may be necessary to Refresh the projects and clean by clicking on the **Project > Clean** and then choose “**Clean All Projects**” in the dialog



- \_\_\_ n. At this time the following error might show up in the Problems tab



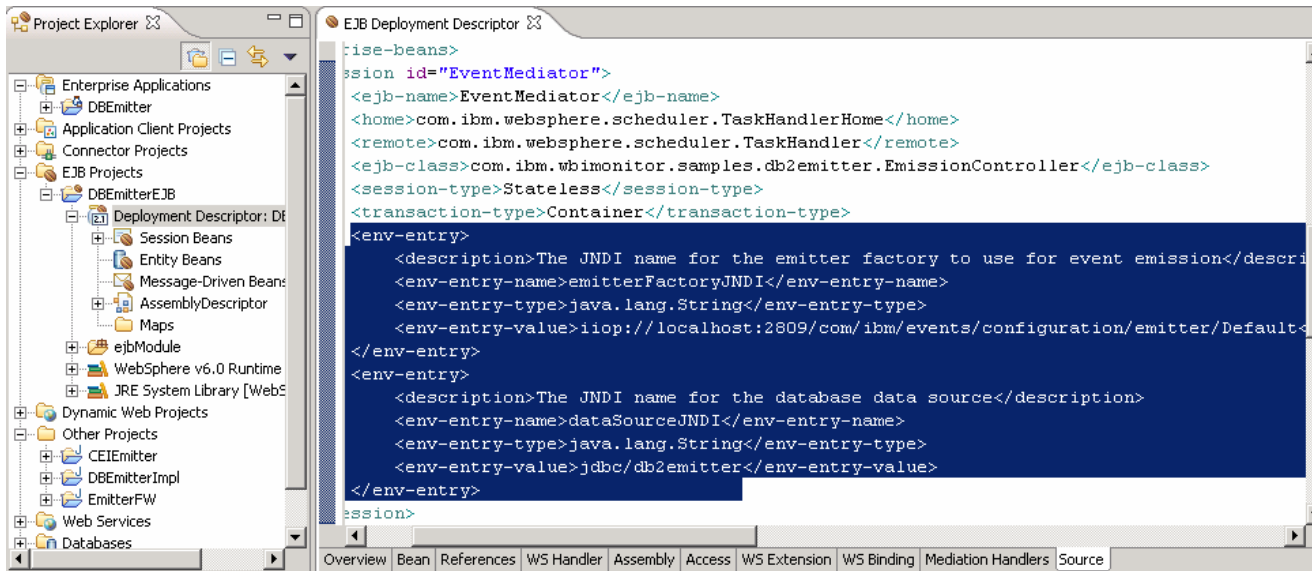
- \_\_\_ o. As the DBEmitterImpl project is dependent on DBEmitterEJB, build the DBEmitterEJB project by right clicking on it then selecting **Build Project**
- \_\_\_ p. Select **Project > Build All**

\_\_\_ 7. Review the EJB Deployment Descriptor

- \_\_\_ a. In the Project Explorer, expand **EJB Projects > DBEmitterEJB** and double click on the **Deployment Descriptor : DBEmitterEJB** to open it in an editor



- \_\_\_ b. In the EJB Deployment Descriptor Editor, ensure that the **Source** tab is selected



- \_\_\_ c. The following are the default values specified in the EJB deployment descriptor. These can be customized by editing the source projects and exporting a new EAR archive.

```
<env-entry-name>emitterFactoryJNDI</env-entry-name>
<env-entry-value>iiop://localhost:2809/com/ibm/events/configuration/emitter/Default</env-entry-value>
```

---

**Note:** The default emitter factory specified for the “**emitterFactoryJNDI**” environment entry uses port 2809. Check your server’s `BOOTSTRAP_ADDRESS` to ensure that is the correct port to use. Otherwise, import the source projects into WebSphere Integration Developer and make the necessary modifications to the EJB deployment descriptor.

```
<env-entry-name>dataSourceJNDI</env-entry-name>
<env-entry-value>jdbc/db2emitter</env-entry-value>
```

---

**Note:** The JNDI name for the data source “**dataSourceJNDI**” environment entry is `jdbc/db2emitter`. Configure the DB2 Emitter data source with this JNDI name.

```
<env-entry-name>schedulerJNDI</env-entry-name>
<env-entry-value>sched/DB2Poller</env-entry-value>
```

---

**Note:** The JNDI name for the scheduler “**schedulerJNDI**” environment entry is `sched/DB2Poller`. Configure the Scheduler with this JNDI name.

```

<env-entry>
  <description>The JNDI name for the emitter factory to use for event emission</description>
  <env-entry-name>emitterFactoryJNDI</env-entry-name>
  <env-entry-type>java.lang.String</env-entry-type>
  <env-entry-value>iiop://localhost:2809/com/ibm/events/configuration/emitter/Default</env-entry-value>
</env-entry>
<env-entry>
  <description>The JNDI name for the database data source</description>
  <env-entry-name>dataSourceJNDI</env-entry-name>
  <env-entry-type>java.lang.String</env-entry-type>
  <env-entry-value>jdbc/db2emitter</env-entry-value>
</env-entry>
</session>
<session id="TaskScheduler">
  <ejb-name>TaskScheduler</ejb-name>
  <home>com.ibm.websphere.startupservice.AppStartupHome</home>
  <remote>com.ibm.websphere.startupservice.AppStartup</remote>
  <ejb-class>com.ibm.wbimonitor.samples.db2emitter.TaskSchedulerBean</ejb-class>
  <session-type>Stateless</session-type>
  <transaction-type>Container</transaction-type>
  <env-entry>
    <description>The JNDI name for the scheduler</description>
    <env-entry-name>schedulerJNDI</env-entry-name>
    <env-entry-type>java.lang.String</env-entry-type>
    <env-entry-value>sched/DB2Poller</env-entry-value>
  </env-entry>

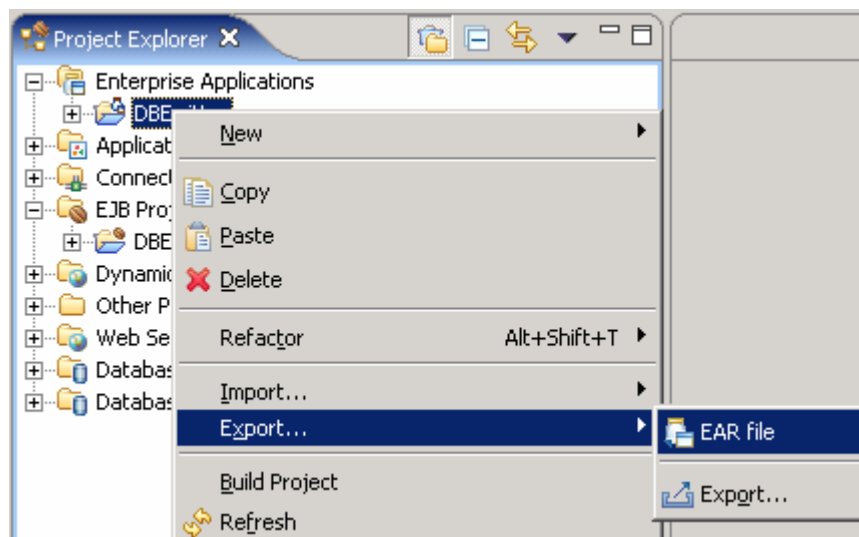
```

\_\_\_ d. Close the EJB Deployment Descriptor Editor.

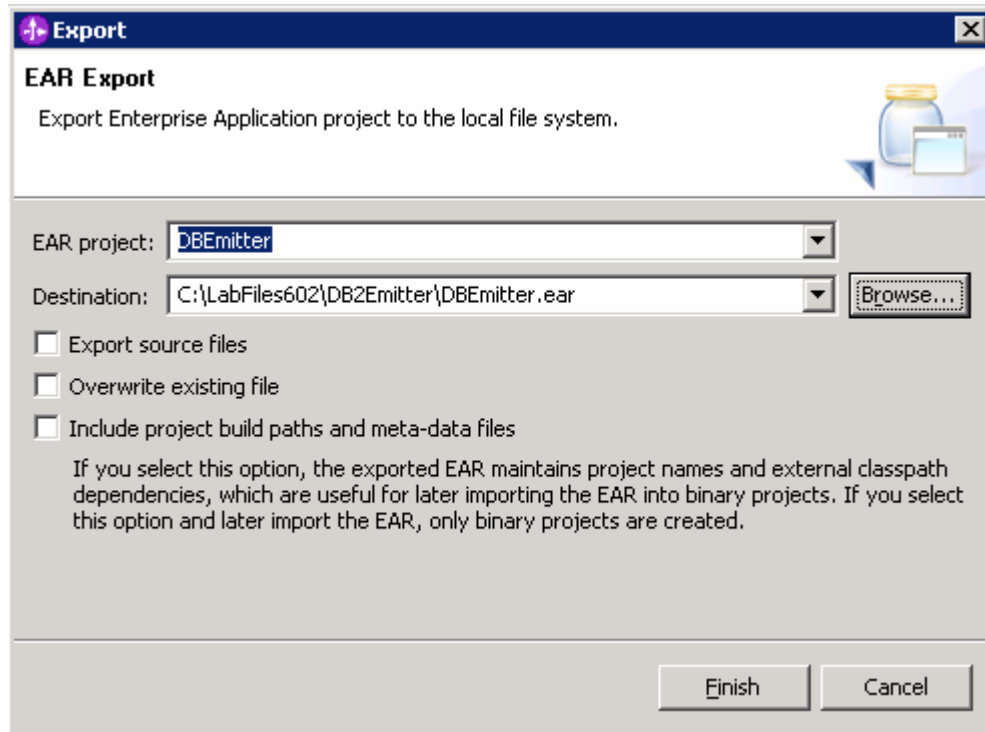
\_\_\_ 8. Export the **DBEmitter** ear file.

\_\_\_ a. Select **DBEmitter** project in the Enterprise Applications.

\_\_\_ b. Right-click on the **DBEmitter** project to open the context menu. Select **Export > EAR file**



\_\_\_ c. In the **Export** dialog, select **DBEmitter** for EAR Project and Browse to choose a Destination for the EAR.



\_\_ d. Press **Finish** button

## Part 2: Create a DB2 emitter database

In this part, the DB2 Emitter database (**For example:** DBEMITT) is created, and three DDL scripts are ran to create an Application Table (**For example:** CustomerOrder), the Event and Log Tables, a procedure and three TRIGGER statements for CREATE, DELETE & UPDATE for the Application table (**For example:** CustomerOrder).

### \_\_\_ 1. Creating Application, Event and Log tables

**Table 1:** Application Table (**CustomerOrder**)

ORDERNUMBER	INT	PRIMARY KEY GENERATED ALWAYS AS IDENTITY (START WITH 1 INCREMENT BY 1)
CUSTOMERNAME	VARCHAR	
COUNTRY	VARCHAR	
CITY	VARCHAR	
PRODUCTNUMBER	VARCHAR	
QUANTITY	INT	
ORDERPRICE	FLOAT	

Events happen here in the Application Table (**CustomerOrder**) with a record created, updated or deleted (**C/U/D**). Based on the state (**C/U/D**) of this record a trigger is invoked

**Table 2:** EVENTTABLE

SID	BIGINT	PRIMARY KEY GENERATED ALWAYS AS IDENTITY (START WITH 1 INCREMENT BY 1)
TYPE	VARCHAR	The type of event which the <b>EventFormatterFactory</b> knows.
TRIGGER	CHAR	The kind of trigger. ( <b>Create / Update / Delete</b> )
KEYTYPE	VARCHAR	The type of key which the <b>EventFormatter</b> knows.
KEYVALUE	VARCHAR	The value of key which the <b>EventFormatter</b> knows
CREATETIMESTAMP	TIMESTAMP	Time when an event posted.

Events on the Application Table (**CustomerOrder**) are stored as records in the EVENTTABLE by triggers. The trigger inserts a record with the primary key and additional information to the EVENTTABLE. The additional information includes the trigger type (**C/U/D**), and event type, and the creation timestamp.

**Table 3: LOGTABLE**

SID	BIGINT	The primary key.
TYPE	VARCHAR	The kind of event which the EventFormatterFactory knows.
TRIGGER	CHAR	The kind of trigger. (Create / Update / Delete)
KEYTYPE	VARCHAR	The type of key which the EventFormatter knows.
KEYVALUE	VARCHAR	The value of key which the EventFormatter knows
CREATETIMESTAMP	TIMESTAMP	Time when an event posted.
RESULT	CHAR	The result of emission (Success / Fail)
LOGTIMESTAMP	TIMESTAMP	Time when an event logged.

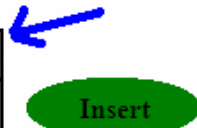
The results of the event emission are logged into the Log Table.

2. Creating the TRIGGER Statement

In the TRIGGER Statement, a statement for copying necessary data from the **Application Table** (CUSTOMERORDER) and inserting a record populated by using the data into the EVENTTABLE is needed. The Type of the key column of the Application Table should be specified.

**CUSTOMERORDER (Application Table)**

ORDERNUMBER	CUSTOMERNAME	COUNTRY	CITY	PRODUCTNUMBER	QUANTITY	ORDERPRICE
50	Jane Doe	USA	Austin	P123	2	2999.50



Trigger ↓

SID	TYPE	TRIGGER	KEYTYPE	KEYVALUE	CREATETIMESTAMP
250	CustomerOrderTEST	C	INT	50	2006-3-17 13:22:00

**EVENTTABLE**

In this case, the TRIGGER statements for **Create/Update/Delete** would be as follows:

Example1. TRIGGER statement for **CUSTOMERORDER** table

```
-- CREATE TRIGGER statement
CREATE TRIGGER CUSTOMERCREATE AFTER INSERT ON CUSTOMERORDER REFERENCING
NEW AS NEWROW FOR EACH ROW MODE DB2SQL BEGIN ATOMIC INSERT INTO
EVENTTABLE(TYPE, TRIGGER, KEYTYPE, KEYVALUE, CREATETIMESTAMP)
VALUES('CustomerOrderTEST', 'C', 'INT', RTRIM(CHAR(NEWROW. ORDERNUMBER)), CURRENT
TIMESTAMP); END;

-- UPDATE TRIGGER statement
CREATE TRIGGER CUSTOMERUPDATE AFTER UPDATE ON CUSTOMERORDER REFERENCING
OLD AS OLDROW FOR EACH ROW MODE DB2SQL BEGIN ATOMIC INSERT INTO
```

```
EVENTTABLE(TYPE, TRIGGER, KEYTYPE, KEYVALUE, CREATETIMESTAMP)
VALUES('CustomerOrderTEST', 'U', 'INT', RTRIM(CHAR(OLDROW. ORDERNUMBER)), CURRENT
TIMESTAMP); END;
```

-- DELETE TRIGGER statement

```
CREATE TRIGGER CUSTOMERDELETE AFTER DELETE ON CUSTOMERORDER REFERENCING
OLD AS OLDROW FOR EACH ROW MODE DB2SQL BEGIN ATOMIC INSERT INTO
EVENTTABLE(TYPE, TRIGGER, KEYTYPE, KEYVALUE, CREATETIMESTAMP)
VALUES('CustomerOrderTEST', 'D', 'INT', RTRIM(CHAR(OLDROW. ORDERNUMBER)), CURRENT
TIMESTAMP); END;
```

The **bold** description above represents customization points for you to implement your own triggers.

**CUSTOMERORDER** : The Application table Name

**CustomerOrderTEST** : The event type which is registered to the EmitterFactory or known by EventFormatterFactory

**C/U/D** : The Trigger type (**For example:** For Creation event set 'C')

**INT** : The key type. In this case, ORDERNUMBER column is INT

**RTRIM (CHAR (NEWROW. ORDERNUMBER)):** The key value. If the type of key column is not 'CHAR', you need to cast the key value to CHAR.

---

**Note:** The DDL scripts for creating the CUSTOMERORDER tables and the triggers are provided in <LABFILES>DB2Emitter\SetUpFiles

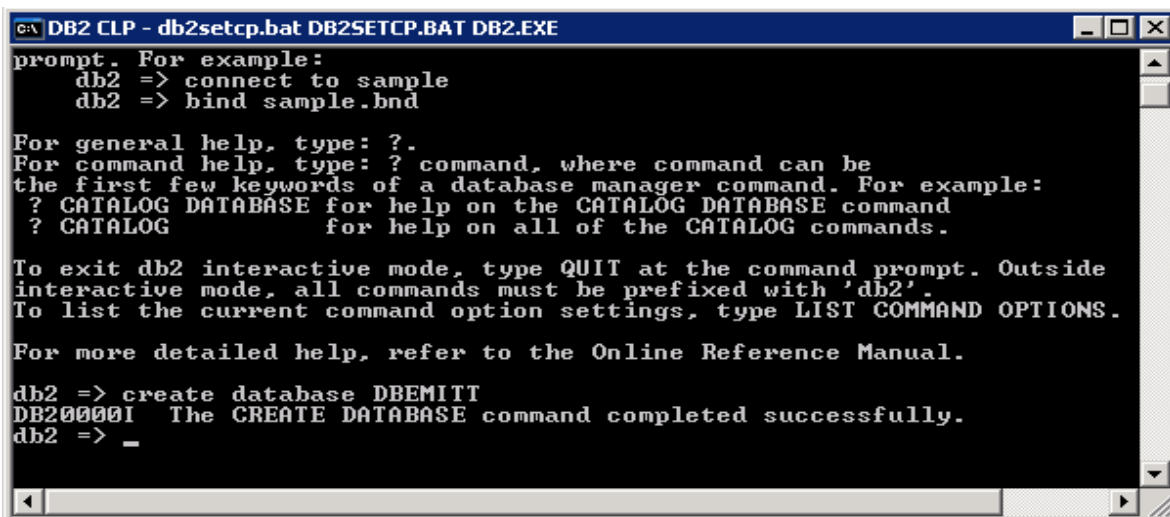
---

### 3. Creating the DB2Emitter Database

a. Open DB2 CommandLine Processor (**Open start > Programs > IBM DB2 > Command Line Tools > CommandLine Processor**)

b. Create new database using the CommandLine Processor

**For example:** create database DBEMITT



```

C:\> DB2 CLP - db2setcp.bat DB2SETCP.BAT DB2.EXE
prompt. For example:
db2 => connect to sample
db2 => bind sample.bnd

For general help, type: ?.
For command help, type: ? command, where command can be
the first few keywords of a database manager command. For example:
? CATALOG DATABASE for help on the CATALOG DATABASE command
? CATALOG          for help on all of the CATALOG commands.

To exit db2 interactive mode, type QUIT at the command prompt. Outside
interactive mode, all commands must be prefixed with 'db2'.
To list the current command option settings, type LIST COMMAND OPTIONS.

For more detailed help, refer to the Online Reference Manual.

db2 => create database DBEMITT
DB20000I  The CREATE DATABASE command completed successfully.
db2 => _

```

---

**Note:** If the DB2Emitter database is created on a remote machine, then catalog the database:

```
catalog tcpip node ${DB_NODE} remote ${DB_HOST} server 50000
catalog db DBEMITT as DBEMITT at node ${DB_NODE}
```

---

- \_\_ c. Close the CommandLine Processor
- \_\_ d. Open DB2 Command Window (**Open start > Programs > IBM DB2 > Command Line Tools > Command Window**)
- \_\_ e. Connect to **DBEMITT** database using the db2 administrator username and password

**For example:** db2 connect to DBEMITT user db2admin using xxxxxx (password)

```
C:\IBM\DB2\SQLLIB\BIN>db2 connect to DBEMIT user db2admin using db2admin

Database Connection Information

Database server          = DB2/NT 8.2.6
SQL authorization ID    = DB2ADMIN
Local database alias    = DBEMIT

C:\IBM\DB2\SQLLIB\BIN>
```

- \_\_ f. Run createCustomerTable.ddl

**For example:** db2 -tf C:\Labfiles602\DB2Emitter\SetUpFiles\createCustomerTable.ddl

```
C:\IBM\DB2\SQLLIB\BIN>db2 -tf C:\Labfiles602\DB2Emitter\SetUpFiles\createCustomerTable.ddl
DB21034E The command was processed as an SQL statement because it was not a valid Command Line Processor command. During SQL processing it returned:
SQL0204N "DB2ADMIN.CUSTOMERORDER" is an undefined name.  SQLSTATE=42704

DB20000I The SQL command completed successfully.

C:\IBM\DB2\SQLLIB\BIN>
```

---

**Note:** If the DDL scripts are being run for the first time, an error preceding every create table command will be seen. This is due to the DROP statements being run before the create statements. You can safely ignore these errors.

---

- \_\_ g. Run createEventAndLogTable.ddl

**For example:** db2 -tf C:\Labfiles602\DB2Emitter\SetUpFiles\createEventAndLogTable.ddl

```

C:\DB2 CLP
C:\IBM\DB2\SQLLIB\BIN>db2 -tf C:\Labfiles602\DB2Emitter\SetUpFiles\createEventAn
dLogTable.ddl
DB21034E The command was processed as an SQL statement because it was not a
valid Command Line Processor command. During SQL processing it returned:
SQL0204N "DB2ADMIN.EVENTTABLE" is an undefined name.  SQLSTATE=42704

DB20000I The SQL command completed successfully.

DB21034E The command was processed as an SQL statement because it was not a
valid Command Line Processor command. During SQL processing it returned:
SQL0204N "DB2ADMIN.LOGTABLE" is an undefined name.  SQLSTATE=42704

DB20000I The SQL command completed successfully.

DB21034E The command was processed as an SQL statement because it was not a
valid Command Line Processor command. During SQL processing it returned:
SQL0204N "DB2ADMIN.LOGPROC" is an undefined name.  SQLSTATE=42704

DB20000I The SQL command completed successfully.

```

\_\_ h. Run createCustomerTrigger.ddl

**For example:** db2 -tf C:\Labfiles602\DB2Emitter\SetUpFiles\createCustomerTrigger.ddl

```

C:\DB2 CLP
C:\IBM\DB2\SQLLIB\BIN>db2 -tf C:\Labfiles602\DB2Emitter\SetUpFiles\createCustome
rTrigger.ddl
DB21034E The command was processed as an SQL statement because it was not a
valid Command Line Processor command. During SQL processing it returned:
SQL0204N "DB2ADMIN.CUSTOMERCREATE" is an undefined name.  SQLSTATE=42704

DB20000I The SQL command completed successfully.

DB21034E The command was processed as an SQL statement because it was not a
valid Command Line Processor command. During SQL processing it returned:
SQL0204N "DB2ADMIN.CUSTOMERUPDATE" is an undefined name.  SQLSTATE=42704

DB20000I The SQL command completed successfully.

DB21034E The command was processed as an SQL statement because it was not a
valid Command Line Processor command. During SQL processing it returned:
SQL0204N "DB2ADMIN.CUSTOMERDELETE" is an undefined name.  SQLSTATE=42704

DB20000I The SQL command completed successfully.

```

\_\_ i. Close the DB2 Command Window

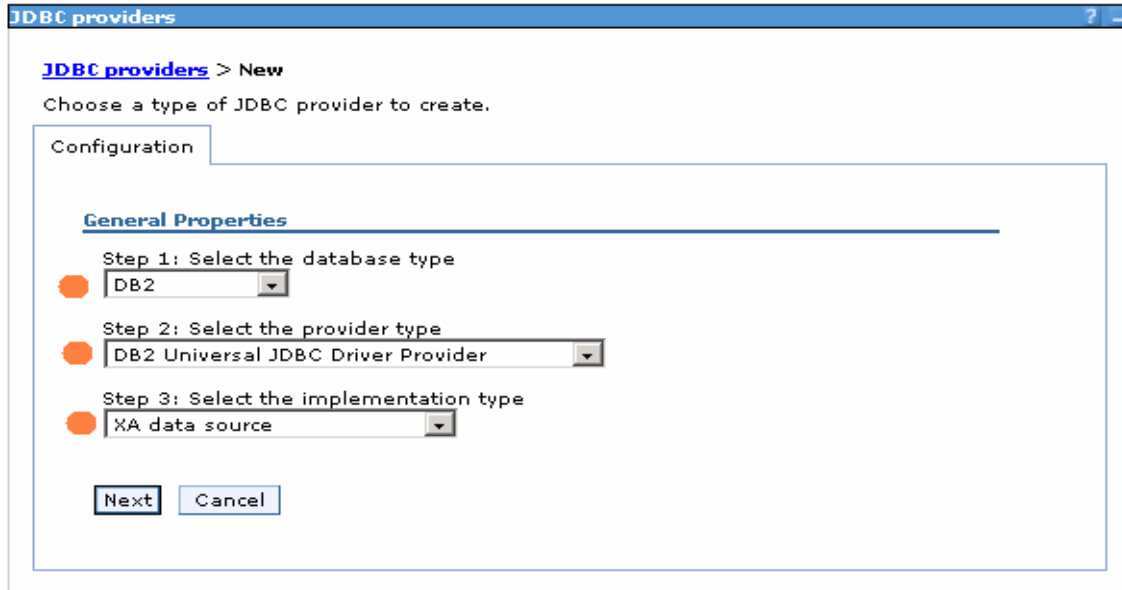


---

## Part 3: Configure data source for DB2 emitter database

In this part, the Process Server is configured with a data source for the DBEmitter application to do the database transactions with the DBEMITT database, configure J2C Authentication data for the data source to authenticate with the DBEMITT database.

- \_\_\_ 1. Configure J2C Authentication
  - \_\_\_ a. Log on to the Administrative Console
  - \_\_\_ b. In the left pane locate **Security > Global security** and in the right pane expand **JAAS Configuration** and click on **J2C Authentication data** under the **Authentication** category.
  - \_\_\_ c. In the following screen click **New** to create a new Authentication Alias
  - \_\_\_ d. Enter the following General properties:
    - 1) Alias : **DB2EmitAlias**
    - 2) User ID : **db2admin** (or the db2 user you are using)
    - 3) Password : **xxxxxx** (DB2 password you are using)
  - \_\_\_ e. Click **OK** and **Save** to the Master Configuration.
- \_\_\_ 2. Create a new JDBC Provider
  - \_\_\_ a. Locate **Resources > JDBC Providers** and select the radio button next to **Server** to ensure that the JDBC Provider is created at the server scope
  - \_\_\_ b. Click **New** to create new JDBC Provider
  - \_\_\_ c. Set the following for the General Properties:
    - 1) Database type: **DB2**
    - 2) Provider type: **DB2 Universal JDBC Driver Provider**
    - 3) Implementation type: **XA data source**



\_\_\_ d. Click **Next**

\_\_\_ e. In the following screen enter **DB2Emitter Provider (XA)** in Name field. Click **OK** and **Save** to the master configuration

Select	Name	Description
<input type="checkbox"/>	<a href="#">Cloudscape JDBC Provider</a>	Cloudscape 51 embedded JDBC2-compliant Provider
<input type="checkbox"/>	<a href="#">Cloudscape JDBC Provider (XA)</a>	Built-in Cloudscape JDBC Provider (XA)
<input type="checkbox"/>	<a href="#">DB2Emitter Provider (XA)</a>	XA DB2 Universal JDBC Driver-compliant Provider. Datasources created under this provider support the use of XA to perform 2-phase commit processing. Use of driver type 2 on WebSphere Application Server for Z/OS is not supported for datasources created under this provider.
<input type="checkbox"/>	<a href="#">Event DB2 JDBC Provider</a>	DB2 Universal JDBC Driver Provider (XA) for the Common Event Infrastructure
<input type="checkbox"/>	<a href="#">Scheduler Cloudscape JDBC Provider (XA)</a>	Cloudscape 51 embedded JDBC2-compliant Provider
Total 5		

\_\_\_ 3. Create a new Data source. Now that a JDBC provider had been created, a data source data will be created

\_\_\_ a. Locate **Resources > JDBC Providers > DB2Emitter Provider (XA)** and click on **Data sources** under Additional properties category

### Additional Properties

- [Data sources](#)
- [Data sources \(Version 4\)](#)

\_\_ b. Click on the **New** button to create a new data source

\_\_ c. Enter the following General Properties:

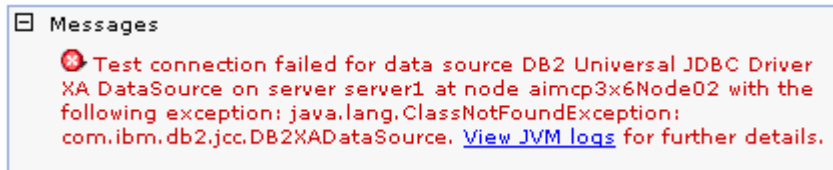
- 1) Name: **DB2 Emitter DataSource**
- 2) JNDI name: **jdbc/db2emitter** (Note, this is the default JNDI name specified as an environment entry in the EJB deployment descriptor)
- 3) Disable "Use this Data Source in container managed persistence (CMP)"
- 4) Description: **JDBC DataSource for DB2 Sample Event Emitter**
- 5) Component-managed authentication alias: **{nodeName}/DB2EmitterAlias** (the name of the authentication alias previously created in Step 1)
- 6) Database name: **DBEMITT** (the name of the database previously created in Step 5)

\_\_ d. Click **OK** and **Save** to the master configuration

\_\_ e. Test the Connection to the **DBEMITT** database

---

**Note:** If test connection fails as follows then you must set a WebSphere variable:



Locate WebSphere variables in the administrative console under the Environment section.

Set **DB2UNIVERSAL\_JDBC\_DRIVER\_PATH** in the name field and the DB2 library path in the value field (For example: C:\IBM\SQLLIB\java), then click Apply

***The scope of this variable depends on the scope of the JDBC Provider you configured to.***

---

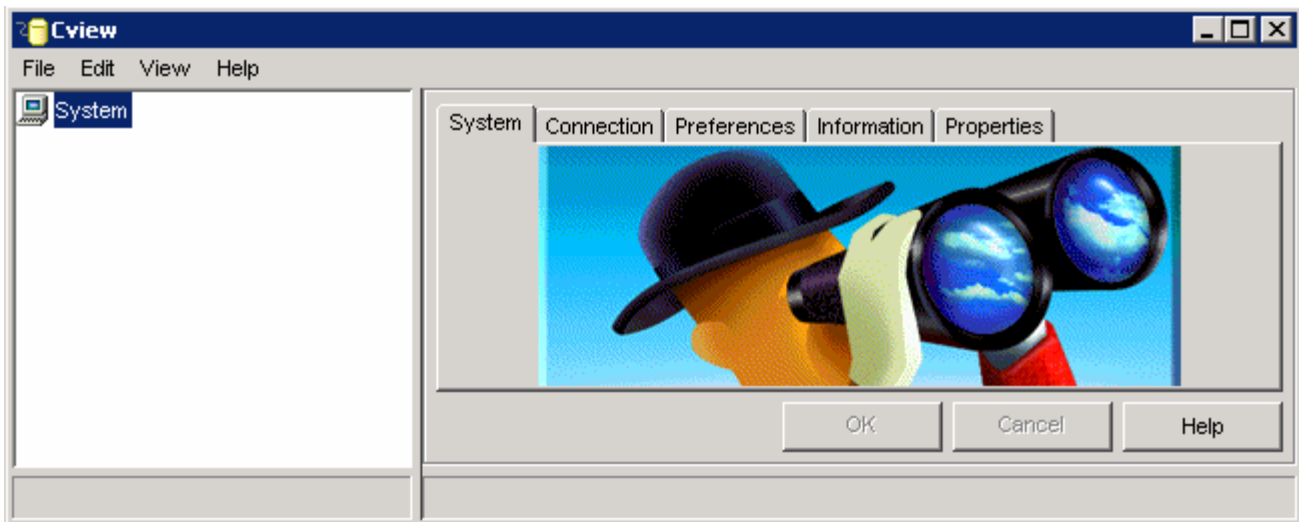
\_\_\_ 4. The DataSource configuration for the DBEMITT database is complete

## Part 4: Configure a scheduler service and deploy the DB2Emitter EAR

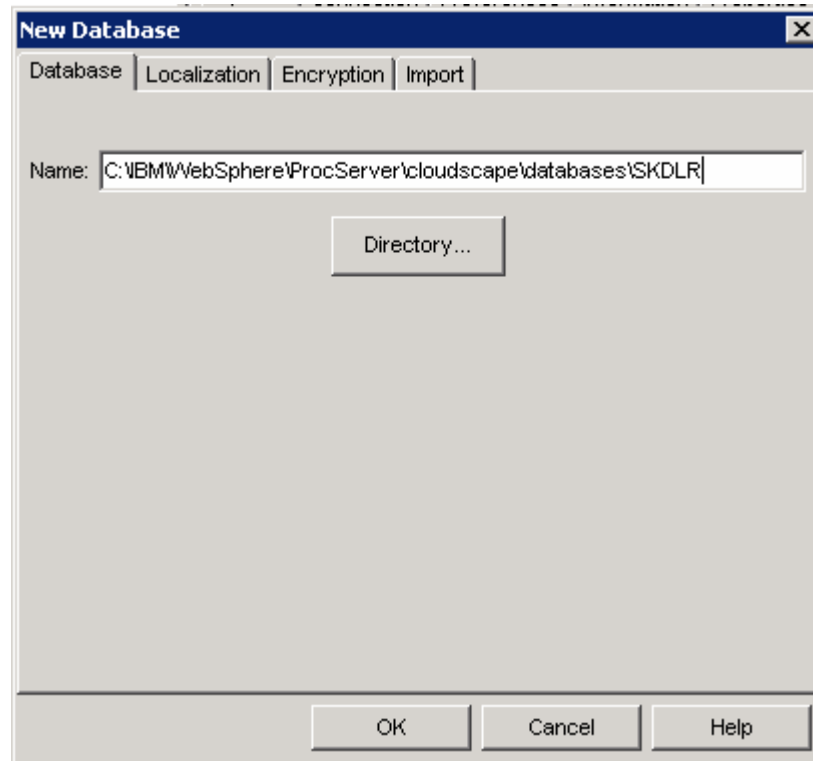
In this part, a Scheduler Service is created for the DB2Emitter to invoke the EmissionController at a specified interval.

Follow the steps below to create a cloudscape database for the scheduler service; configure a DataSource for the same and finally a scheduler for the DB2Emitter

- \_\_\_ 1. Create a Cloudscape database for the Scheduler Service
  - \_\_\_ a. Launch cview.bat
  - \_\_\_ b. Explore to <WPS\_HOME>/cloudscape/bin/embedded and double-click **cvview.bat** to launch the Cview window

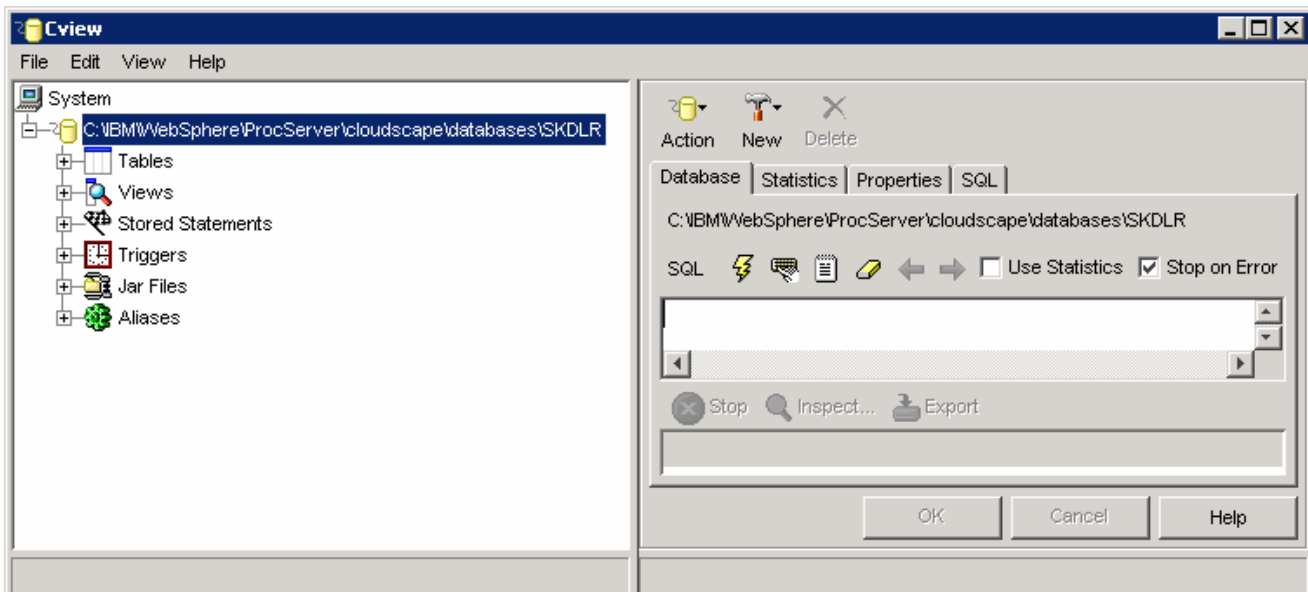


- \_\_\_ c. From the main menu, select **File → New → Database**
- \_\_\_ d. In the name field, enter <WPS\_HOME>/cloudscape/databases/SKDLR  
Where as <WPS\_HOME> is C:\IBM\WebSphere\ProcServer  
**For example: - C:\IBM\WebSphere\ProcServer\cloudscape\databases\SKDLR**



\_\_ e. Click **OK**

\_\_ f. Verify that the database is created and is listed on the left pane of the Cview window



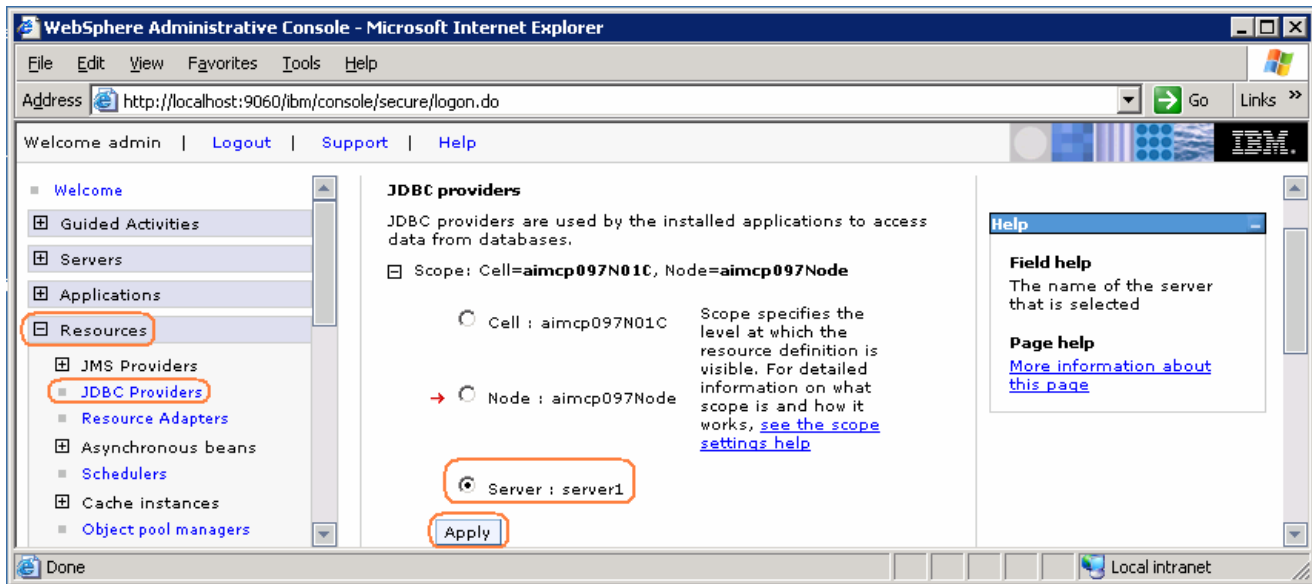
\_\_ g. Close the Cview window (**File → Exit**)

\_\_\_ 2. Create a DataSource for the Scheduler Service

\_\_ a. Launch the WebSphere Process Server Administrative Console and logon to it

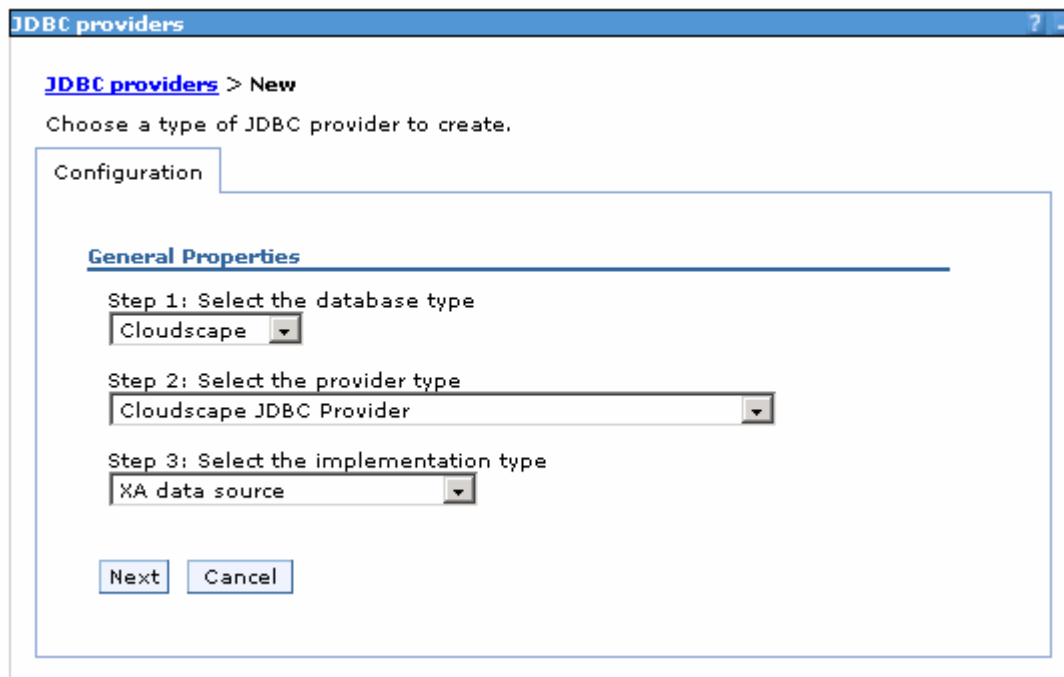
For example: - **http://localhost:9060/ibm/console/**

\_\_ b. In the left pane, locate Resources → JDBC Providers, set the scope to **Servers** and click **Apply**



\_\_ c. Click the **New** button to create a new JDBC Provider

- 1) Set **Cloudscape** in database type
- 2) Set **Cloudscape JDBC Provider** in provider type
- 3) Set **XA data source** in implementation type



\_\_ d. Click **Next**

- \_\_\_ e. In the following screen, enter the **Name** filed to **Scheduler Cloudscape JDBC Provider (XA)**, Click **OK** and **Save** to the master configuration
- \_\_\_ f. You must the new the Cloudscape JDBC Provider listed as shown below:

Select	Name	Description
<input type="checkbox"/>	<a href="#">Cloudscape JDBC Provider</a>	Cloudscape 51 embedded JDBC2-compliant Provider
	<a href="#">Cloudscape JDBC Provider (XA)</a>	Built-in Cloudscape JDBC Provider (XA)
<input type="checkbox"/>	<a href="#">Event DB2 JDBC Provider</a>	DB2 Universal JDBC Driver Provider (XA) for the Common Event Infrastructure
<input type="checkbox"/>	<a href="#">Scheduler Cloudscape JDBC Provider (XA)</a>	Cloudscape 51 embedded JDBC2-compliant Provider
Total 4		

- \_\_\_ g. Now that a JDBC provider had been created, create a data source
- \_\_\_ h. Locate **Resources > JDBC Providers > Scheduler Cloudscape JDBC Provider (XA)** and click on **Data sources** under **Additional properties**

Additional Properties

- [Data sources](#)
- [Data sources \(Version 4\)](#)

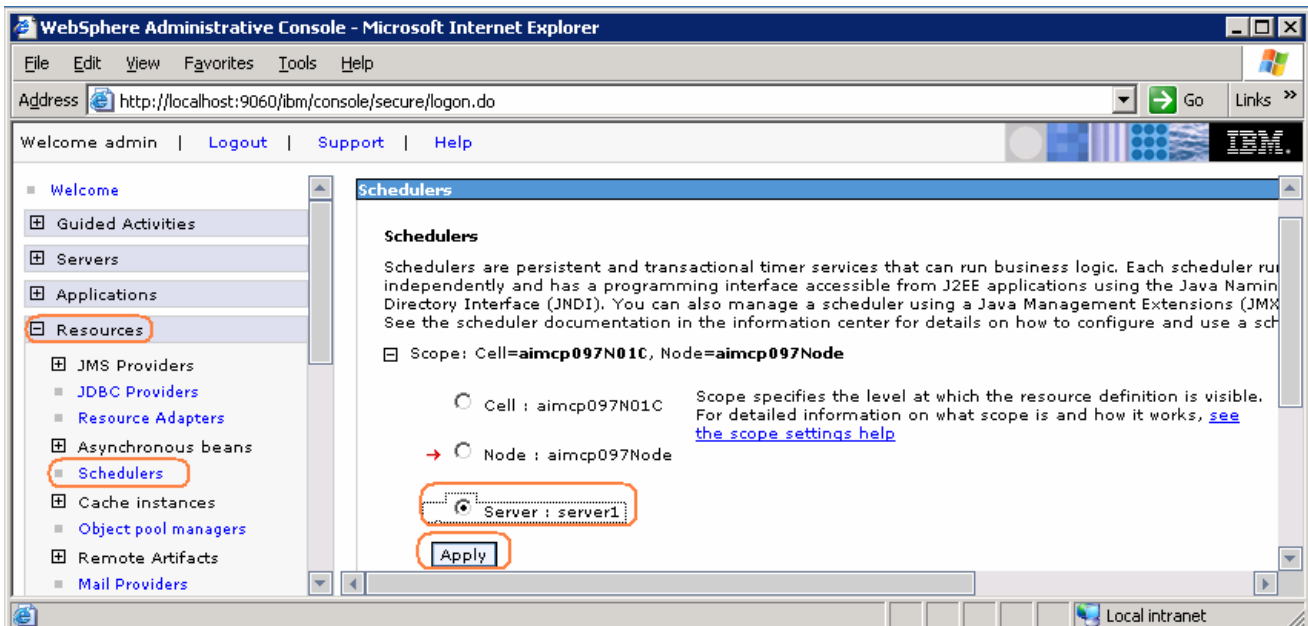
- \_\_\_ i. In the following screen, click the **New** button to create a new data source
- \_\_\_ j. Under General properties, enter the following:

- 1) Name : **SKDLR DataSource**
- 2) JNDI name: **jdbc/skdlr**
- 3) Disable "Use this Data Source in container managed persistence (CMP)"
- 4) Description: **JDBC DataSource for SKDLR Database**
- 5) Database name: **C:\IBM\WebSphere\ProcServer\cloudscape\databases\SKDLR (<WPS\_HOME>\cloudscape\databases\SKDLR)**

- \_\_\_ k. Click **OK** and **Save** to the master configuration
- \_\_\_ l. Test the connection to the scheduler database and ensure that a successful message is resulted

\_\_\_ 3. Create a scheduler for the DB2 Emitter:

- \_\_\_ a. Locate **Resources > Schedulers** and the set the scope to **Servers**



\_\_ b. Click **New** to create a new Scheduler configuration

\_\_ c. In the Scheduler > New screen, enter the following information:

- 1) Name: **DB2Poller**
- 2) JNDI name: **sched/DB2Poller** (This is the default specified as an environment entry in the EJB deployment descriptor)
- 3) Description: **Scheduler for DB2 Sample Emitter's Event Table Poller**
- 4) Data source JNDI name: **jdbc/skdlr** (select from the drop down list)
- 5) Table prefix: **DB2EMTR\_**
- 6) Poll interval: **30**
- 7) Work managers: **DefaultWorkManager**



**General Properties**

\* Scope

\* Name

\* JNDI name

Description

Category

\* Data source JNDI name

Data source alias

\* Table prefix

\* Poll interval  
 seconds

\* Work managers

Use administration roles

\_\_ d. Click **OK** and **Save** to the master configuration

\_\_ e. Select check box next to **DB2Poller** and click the **Create Tables** button

Select	Name	JNDI name	Data source JNDI name	Table prefix	Poll interval	Work managers
<input type="checkbox"/>	<a href="#">AppScheduler</a>	AppScheduler	jdbc/WPSDB	WSCH_	10	AppSchedulerWorkManager
<input checked="" type="checkbox"/>	<a href="#">DB2Poller</a>	sched/DB2Poller	jdbc/skdlr	DB2EMTR_	30	DefaultWorkManager
Total 2						

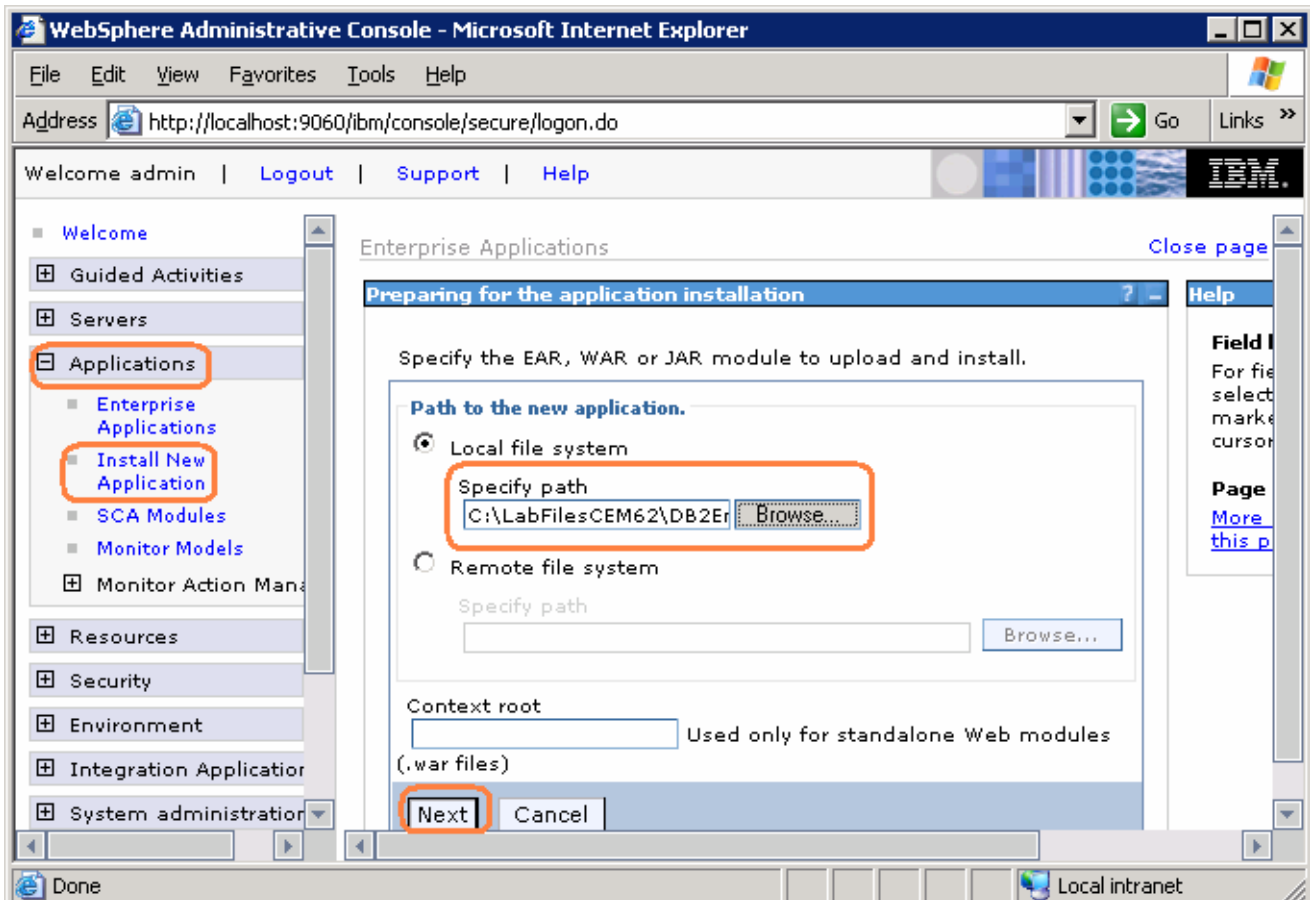
\_\_ f. Ensure that the tables are created successfully

\_\_\_ 4. The Scheduler configuration is complete

\_\_\_ 5. Deploy the DB2Emitter EAR file.

**Note:** Ensure that the EAR is copied to the file system local to the application server where you would like to deploy

- \_\_\_ a. Logon to the Administration console. In the left pane, select **Applications >Install New Application**
- \_\_\_ b. Click **Browse** to **Specify Path** on the Local file system in the right panel and click **Next**



- \_\_\_ c. Progress through all the 6 steps of "Install New Application" screens, accepting the defaults
- \_\_\_ d. In Step 6, the **Summary** screen, click the **Finish** button. The install will take several moments
- \_\_\_ e. After the installation is complete, ensure that the message "Application DBEmitter installed successfully" is displayed.

Application DBEmitter installed successfully.

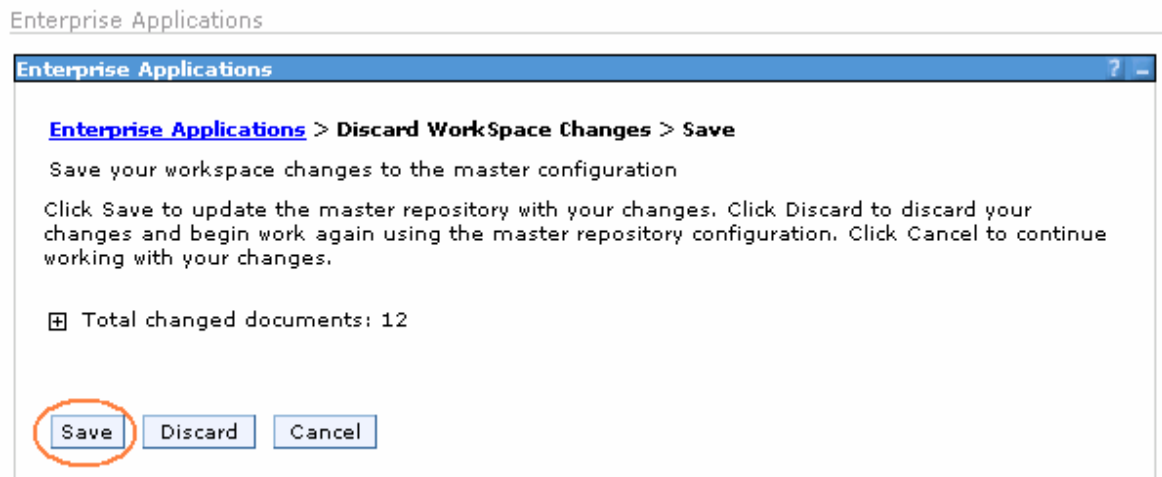
To start the application, first save changes to the master configuration.

**Save to Master Configuration**

To work with installed applications, click the "Manage Applications" button.

**Manage Applications**

- \_\_\_ f. Click **“Save to Master Configuration”**, and finally click **Save** on the **“Save”** screen.



- \_\_\_ 6. **Restart the Process Server**

## Part 5: Testing DB2 emitter

1. Now that the Process Server configuration is complete and DB2Emitter application is deployed, it is time to do a simple test for CEI event emissions.
  - a. Ensure that the Process Server and the DB2Emitter application are running.
  - b. Open the DB2 Command Line Processor. (**Start > Programs > IBM DB2 > Command Line Tools > Command Line Processor**)
  - c. Run **connect to DBEMITT** using the DB2 administrator user and password
 

**For example:** connect to DBEMITT user db2admin using xxxxxx (or the user/password being used)
  - d. Run an insert statement for the CUSTOMERORDER table. **For example:**

**example:** insert into CUSTOMERORDER(CUSTOMERNAME, COUNTRY , CITY, PRODUCTNUMBER, QUANTITY, ORDERPRICE) values('Jane Doe', 'USA', 'Austin', 'P123', 2 , 2999.50)

```

C:\> DB2 CLP - db2setcp.bat DB2SETCP.BAT DB2.EXE

For general help, type: ?.
For command help, type: ? command, where command can be
the first few keywords of a database manager command. For example:
? CATALOG DATABASE for help on the CATALOG DATABASE command
? CATALOG          for help on all of the CATALOG commands.

To exit db2 interactive mode, type QUIT at the command prompt. Outside
interactive mode, all commands must be prefixed with 'db2'.
To list the current command option settings, type LIST COMMAND OPTIONS.

For more detailed help, refer to the Online Reference Manual.

db2 => connect to DBEMITT user db2admin using db2admin

Database Connection Information

Database server          = DB2/NT 8.2.6
SQL authorization ID    = DB2ADMIN
Local database alias    = DBEMITT

db2 => insert into CUSTOMERORDER(CUSTOMERNAME, COUNTRY , CITY, PRODUCTNUMBER, QU
ANTITY, ORDERPRICE) values( 'Jane Doe', 'USA', 'Austin', 'P123', 2 , 2999.50)
DB20000I The SQL command completed successfully.
db2 =>
  
```

**Note:** Immediately after the insert statement is run, the inserted data is moved to 'EVENTTABLE' by the trigger. DB2Emitter refers to the 'EVENTTABLE' at regular intervals using the Scheduler.

The following is the CUSTOMERORDER table with a record inserted (created)

ORDERNUMBER	CUSTOMERNAME	COUNTRY	CITY	PRODUCTNUMBER	QUANTITY	ORDERPRICE
1	Jane Doe	USA	Austin	P123	2	2,999.5

The result of the event emission is logged to the LOGTABLE as shown below:

SID	TYPE	TRIGGER	KEYTYPE	KEYVALUE	CREATETIMESTAMP	RESULT	LOGTIMESTAMP
1	CustomerOrderTEST	C	INT	1	Mar 24, 2006 10:20:5...	S	Mar 24, 2006 10:2...

**Note:** If emission is successful, a successful record (RESULT='S') is inserted to 'LOGTABLE'

If an Exception occurs, a failed record (RESULT='F') is inserted to 'LOGTABLE'

If a Runtime Exception or an Error occurred, the record is rolled back

- \_\_\_ e. Run an update statement for the CUSTOMERORDER table.  
**(For example:** update CUSTOMERORDER SET ORDERPRICE = 9999.90 where ORDERNUMBER = 1 )

```

C:\> DB2 CLP - db2setcp.bat DB2SETCP.BAT DB2.EXE
SQL authorization ID = DB2ADMIN
Local database alias = DBEMITT

db2 => insert into CUSTOMERORDER(CUSTOMERNAME, COUNTRY , CITY, PRODUCTNUMBER, QU
ANTITY, ORDERPRICE) values('Jane Doe', 'USA', 'Austin', 'P123', 2 , 2999.50)
DB20000I The SQL command completed successfully.
db2 => update CUSTOMERORDER SET ORDERPRICE = 9999.90 where ORDERNUMBER = 1
DB20000I The SQL command completed successfully.
db2 => _
    
```

The following is the representation of data in CUSTOMERORDER and LOGTABLE after an update statement is processed.

ORDERNUMBER	CUSTOMERNAME	COUNTRY	CITY	PRODUCTNUMBER	QUANTITY	ORDERPRICE
1	Jane Doe	USA	Austin	P123	2	9,999.9

SID	TYPE	TRIGGER	KEYTYPE	KEYVALUE	CREATETIMESTAMP	RESULT	LOGTIMESTAMP
1	CustomerOrderTEST	C	INT	1	Mar 24, 2006 10:20:5...	S	Mar 24, 2006 10:2...
2	CustomerOrderTEST	U	INT	1	Mar 24, 2006 11:00:2...	S	Mar 24, 2006 11:0...

- \_\_\_ f. Run a delete statement for the CUSTOMERORDER table. **(For example:** delete from CUSTOMERORDER where ORDERNUMBER = 1 )

```

CA: DB2 CLP - db2setcp.bat DB2SETCP.BAT DB2.EXE
db2 => insert into CUSTOMERORDER(CUSTOMERNAME, COUNTRY, CITY, PRODUCTNUMBER, QU
ANTITY, ORDERPRICE) values('Jane Doe', 'USA', 'Austin', 'P123', 2, 2999.50)
DB20000I The SQL command completed successfully.
db2 => update CUSTOMERORDER SET ORDERPRICE = 9999.90 where ORDERNUMBER = 1
DB20000I The SQL command completed successfully.
db2 => db2 delete from CUSTOMERORDER where ORDERNUMBER = 1
SQL0104N An unexpected token "db2" was found following "BEGIN-OF-STATEMENT".
Expected tokens may include: "SELECT". SQLSTATE=42601
db2 => delete from CUSTOMERORDER where ORDERNUMBER = 1
DB20000I The SQL command completed successfully.
db2 =>
    
```

The following is the representation of data updated in the CUSTOMERORDER and LOGTABLE after a delete statement is issued.

**Open Table - CUSTOMERORDER**

AIMCP3X6 - DB2 - DBEMITT - DB2ADMIN.CUSTOMERORDER

ORDERNUMBER	CUSTOMERNAME	COUNTRY	CITY	PRODUCTNUMBER

Add Row  
Delete Row

**Open Table - LOGTABLE**

AIMCP3X6 - DB2 - DBEMITT - DB2ADMIN.LOGTABLE

SID	TYPE	TRIGGER	KEYTYPE	KEYVALUE	CREATETIMESTAMP	RESULT	LOGTIMESTAMP
1	CustomerOrderTEST	C	INT	1	Mar 24, 2006 10:20:5...	S	Mar 24, 2006 10:21:05 ...
2	CustomerOrderTEST	U	INT	1	Mar 24, 2006 11:00:2...	S	Mar 24, 2006 11:00:26 ...
3	CustomerOrderTEST	D	INT	1	Mar 24, 2006 11:21:4...	S	Mar 24, 2006 11:21:50 ...

Add Row  
Delete Row

- g. To view the events emitted in a CBE browser, logon to the Process Server's Administrative console, Locate **Integration Applications > Common Base Event Browser** click on the **Get Events** and then **All Events**.

The screenshot shows the WebSphere Administrative Console in Microsoft Internet Explorer. The browser address bar shows `http://localhost:9061/bm/console/secure/logon.do`. The page title is "WebSphere software CBE Event Browser".

On the left, there is a navigation menu with the following items: Welcome, Guided Activities, Servers, Applications, Resources, Security, Environment, Integration Applications (highlighted), System administration, Monitoring and Tuning, Troubleshooting, Service integration, and UDDI. Under "Integration Applications", "Common Base Event Browser" is selected.

The main content area is titled "CBE Event Browser" and contains a table of events. The table has columns for Selection, Creation Time, Name, Priority, Severity, Server, Sub-component, and Situation. Three events are listed, all of type "CustomerOrder" from "db2syscs.exe".

Below the table, a summary shows "Page 1 of 1", "Total: 3", "Filtered: 3", "Displayed: 3", and "Selected: 1".

The detailed view of the selected event shows the following data:

extendedDataElement / TRIGGER	CREATE
extendedDataElement / OrderNumber	1
extendedDataElement / CustomerName	Jane Doe
extendedDataElement / Country	USA
extendedDataElement / City	Austin
extendedDataElement / ProductNumber	P123
extendedDataElement / Quantity	2
extendedDataElement / OrderPrice	2999.5
reporterComponentId	
sourceComponentId / component	db2.exe
sourceComponentId / subComponent	db2syscs.exe
sourceComponentId / componentIdType	db2syscs.exe
sourceComponentId / instanceId	
sourceComponentId / application	DB2
sourceComponentId / executionEnvironment	
sourceComponentId / location	lycos

---

## Part 6: Solution

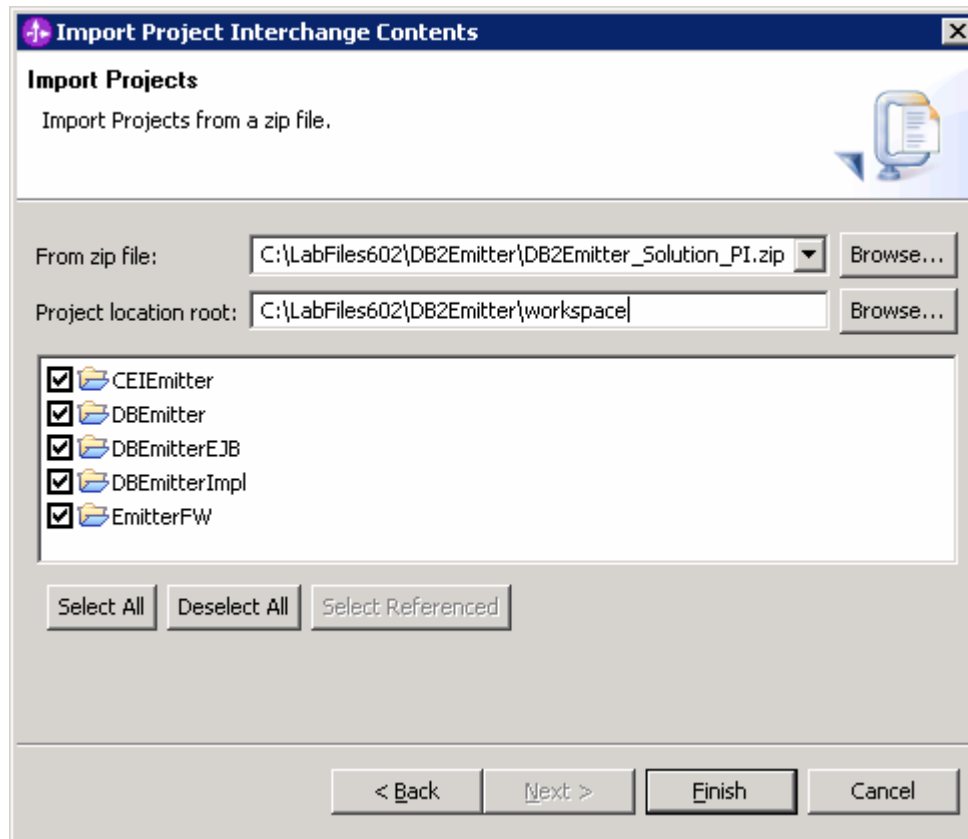
If you would prefer not to create the emitter from scratch, then you can follow this section to import a project interchange file which has the updated source. Using the project interchange that is provided, you can import it to WebSphere Integration Developer, then Build All projects and export the EAR.

If using the solution, skip Part 1, complete this section, and then continue with Part 2.

- \_\_\_ 1. Open WebSphere Integration Developer
  - \_\_\_ a. Select **Start > Programs > IBM WebSphere > Integration Developer V6.0.2 > WebSphere Integration Developer V6.0.2**
  - \_\_\_ b. Workspace Launcher window will be displayed. Click the **Browse...** button and select your workspace directory. (**For example:** C:\LabFiles602\DB2Emitter\workspace)
  - \_\_\_ c. Click **OK**.
  - \_\_\_ d. Close the welcome window by clicking the arrow in the top right corner of the welcome window
  - \_\_\_ e. By default WebSphere Integration Developer opens in **Business Integration** perspective. You need to change it to **J2EE** perspective. To do this click on the top right corner of the WebSphere Integration Developer and choose **Other** if **J2EE** is not listed here.
  - \_\_\_ f. Select **Project > Build Automatically**, to turn off automatic builds
- \_\_\_ 2. Import the solution DB2Emitter project interchange into WebSphere Integration Developer
  - \_\_\_ a. Select **File > Import** from the main menu.
  - \_\_\_ b. From the **Import** window select **Project Interchange** and click **Next**
  - \_\_\_ c. The **Import Project Interchange Contents** window will be opened.
  - \_\_\_ d. Click **Browse...** and select <LABFILES>\DB2Emitter\DB2Emitter\_Solution\_PI.zip as the source .zip file.

**For example:** C:\LabFilesCEM61\DB2Emitter\DB2Emitter\_Solution\_PI.zip
  - \_\_\_ e. Click **Select All** then click **Finish**





- \_\_\_ f. Refresh the projects and clean by clicking on **Project > Clean** and then choose “**Clean All Projects**” in the dialog
- \_\_\_ g. Click **Project > Build All** to build all the projects
- \_\_\_ h. At this point you might see an error in the Problems tab
- \_\_\_ i. As the DBEmitterImpl project is dependent on DBEmitterEJB, build the DBEmitterEJB project by right clicking on it.
- \_\_\_ j. Finally in the Project Explorer, right click on **DBEmitter** under Enterprise Applications to choose **Export > EAR file**.
- \_\_\_ k. Provide the Destination you would like to save the EAR.
- \_\_\_ l. Close WebSphere Integration Developer.
- \_\_\_ m. Continue the lab with **Part 2** of this document.

## What you did in this exercise

You imported the sample DB2 Emitter source into WebSphere Integration Developer and modified it to match the DB2 Emitter's CUSTOMERORDER data. Then you exported the DBEmitter EAR to WebSphere Process Server V6.0.2 that has a CEI SDK installed.

You Deployed the Application to WebSphere Process Server V6.0.2

You created a new DataSource for the Db2 Emitter database.

You created a cloudscape database for the Scheduler, created a DataSource for it and then created a Scheduler service.

You updated records in the Application Table (CUSTOMERORDER) to test for the Event Emissions to the CEI server

Finally you used the Process Server's Common Event Browser to view the emitted events.