Overview

In Part 1, you learned about the IBM RFID middleware solution, which includes WebSphere RFID Device Infrastructure and WebSphere RFID Premises Server. You developed a GPS adapter with the RFID Device Development Kit and ran it on RFID middleware. In Part 2, you'll learn how to define the network topology using the WebSphere RFID Premises Server (hereafter called Premises Server) administrative console, and deploy the application you developed in Part 1 to the WebSphere RFID Device Infrastructure with a real GPS device on a PDA.

After completing this article, you'll be able to:

- Configure WebSphere RFID Premises Server and define the RFID solution's network topology using the Premises Server administrative console.
- Deploy code to WebSphere RFID Device Infrastructure on a PDA.
- Test the end-to-end flow of a data collector on embedded devices, such as PDAs or OBUs (On Board Units) using IBM RFID middleware.

Overview of the scenario

Before starting the second part, review the solution architecture shown in Figure 1. In Part 2, we'll deploy the application that we developed in Part I to the WebSphere RFID Device Infrastructure on a Windows ™ Mobile 2003 PDA, which has a GPS device attached to it. The GPS data will be sent to the Premises Server instead of a premises simulator.
Figure 1. Architecture of the sample scenario

Set up your environment

This article assumes that you have completed the steps in Part 1. In addition to the hardware and software requirements for Part I, you'll need the following hardware and software to complete the steps in this article:

Hardware

Windows Mobile 2003 or Windows Mobile 2003 Second Edition PDA with the following:

- Wireless or wired network adapter, which allows the PDA to connect to the network. (The PDA needs a real IP address when running WebSphere RFID Device Infrastructure.)
- Bluetooth, which allows the GPS device to connect to the PDA.
- A cradle, which allows you to copy/sync files between the PDA and a PC.

Software

For the Windows Mobile 2003 or Windows Mobile 2003 Second Edition PDA:

- A software tool, such as **FRANSON GpsGate**, to convert the Bluetooth/COM access to TCP/IP, because WebSphere RFID Device Infrastructure doesn't support Bluetooth and COM port access on Windows CE. The GPS adapter will use TCP/IP to retrieve the GPS data.
- Windows Mobile ActiveSync to copy files from a PC to the PDA.

Set up Premises Server and an Edge Controller on a PDA

The major steps to building a RFID solution, as described in this article, are as follows. You'll need two desktop machines: one running Premises Server and the other running WebSphere Studio Device Developer (hereafter called Device Developer).
1. Install the Premises Server and start the application server by starting the IBM WebSphere Application Server V5 - server1 service. IBM MQSeries and DB2-related services should already be running after the Premises Server is installed.

2. Define the network topology, as described in Define the RFID network topology.

3. Create a runtime for WebSphere RFID Device Infrastructure on the PDA as described in Create a runtime for WebSphere RFID Device Infrastructure.

4. Deploy the application to the PDA using WSDD, described in Deploy the applications to the PDA with Device Developer.

Define the RFID network topology

The RFID network topology plays an important role in RFID solutions because, like a map, it contains information about the devices in your network. In IBM RFID solutions, the RFID network topology is stored in a configuration database on the Premises Server. The information that's stored in the network topology includes:

- Store IDs for each store location, including dock door IDs. (Here the Store ID is the location ID of the GPS device. This information is important if the Premises Server wants to serve multiple GPS devices on multiple PDAs.)
- Reader IDs and configuration information. (Here the Reader ID is the GPS device ID.)
- Edge Controller IDs and configuration information.

The Edge Controller retrieves this information to set all of the Manager Bundle parameters, including Reader Manager, Controller Manager, and Digital I/O Manager. Before beginning this process, make sure to:

- Get the IP addresses and port numbers of the readers (in our example, readers mean GPS devices) in the network.
- Get the MAC addresses of the Edge Controllers and PDA in the network. (For the purposes of this example, you can ignore this.)

In the following sections, you'll learn how to use the Premises Server administrative console to create and edit the topology definitions. Notice that we treat GPS devices as RFID readers here, so we may use RFID readers and GPS devices interchangeably in this article.

Create the GPS adapter as a new RFID reader type

1. To add GpsNmea as a new RFID reader type in the Premises Server, import the SQL commands shown in Listing 1 into the RFID database. You need to do this because the Premises Server administrative console doesn't currently provide a user interface for adding new RFID reader types. In the following steps, you'll add RFID reader types using the SQL commands.

Listing 1. Sample SQL commands to create a new reader type, gps_db2.sql file

```
INSERT INTO SAGE.READERTYPE VALUES
('GpsNmeaType', '', 'GpsNmea', 'GpsNmea Reader');
INSERT INTO SAGE.RDR_AGENTS VALUES
('GpsNmeaReaderAgent', 'agent for GpsNmea reader');
INSERT INTO SAGE.RDR_AGENT_LOCS VALUES
```
2. Open a DB2 command window by clicking **Start Programs** => **IBM DB2** => **Command Line Tools** => **Command Window** as shown in Figure 2:

**Figure 2. Open DB2 command window**

3. Connect to the RFID database by entering: `db2 connect to IBMRFID` as shown in Figure 3:

**Figure 3. Connect to database**
4. Import the SQL statements by entering `db2 -tvf gps_db2.sql`. Place the file in the `c:\dWork` directory, as shown in Figure 4:

**Figure 4. Import the SQL command**

![Image of SQL command execution](image)

The result of importing the SQL statements is shown in Figure 5:

**Figure 5. Result of the SQL import command**

![Image of SQL command output](image)

5. Verify the newly added reader type on the Premises Server administrative console. The default URL for the Premises Server administrative console is: `http://premises_ip_address:9080/event_admin_web`. From the console, select **Readers** in the left navigation pane. You should see `GpsNmeaType` listed as one of the available reader types as shown in Figure 6:
Now, you're ready to define the RFID network topology. Since you'll deploy the GPS adapter on the PDA, you need to create a location, R2, associated with the PDA, the GPS adapter, also called R2. We'll use the same name for the location and reader (R2) for simplicity's sake, so that we know that location R2 is for reader R2. Each physical reader in the RFID network needs to be assigned to an Edge Controller, so you need to create an Edge Controller and assign the location, R2, and the adapter, R2, to it.

Define the Edge Controller

To create new Edge Controller IDs for the Edge devices in the network, do the following:

1. Log in to the Premises Server administrative console.
2. In the console, select Controller from the left navigation pane. The Controllers dialog displays as shown in Figure 7:
3. Click **New**.

4. In the Create New Controller dialog, do the following, as shown in Figure 8:
   - In the **Controller id** field, specify a unique logical identifier for ensuring information is routed to and from the correct Edge Controller.
   - In the **MAC address** field, enter the Edge Controller's MAC address, which is the MAC of the PDA. In this example, we don't care about the real MAC address of the Edge Controller.
   - The **Alert Threshold** field determines the level of information to be included in the Alert log file on the Premises Server. The default setting is **error**. Setting the alert threshold to **info** of **debug** can generate a large amount of traffic, and overload the network.
   - Click **Create**. The new Edge Controller is created and displayed in the Controllers panel. You'll associate the location and reader with this Edge Controller in a later step.
Define the location information

To define the store location information for the PDA in this network, do the following:

1. Select **Location** from the left navigation pane.
2. In the Locations dialog, select **Root Location** as shown in Figure 9, to create a store location:
3. In the Edit Locations window, accept the default values and click **Create Contained Location** as shown in Figure 10:
4. In the Create New Locations dialog, do the following, as shown in Figure 11:
   - In the Location Id field, enter R2 as the unique Location ID to identify this store. The Location Id can be an easily recognized name, even if the back-end system requires a more cryptic identifier for the location.
   - In the Location Alias field, enter the unique alias name of the location, Alias_R2.

   These fields are logical identifiers that ensure that the GPS data from a particular location is properly routed from the Edge Controller through the Premises Server to the enterprise system. Note that aliases are typically used if the enterprise system to which the Premises Server is passing data requires an identifier other than the one used in the Location Id field.

5. Click Create to create the new location.
Figure 11. The Create new Location dialog

The newly created store location, R2, is listed in the Locations dialog as shown in Figure 12:
Figure 12. Locations with new location R2

Define the RFID reader in the network

Since you'll be deploying the GPS adapter on the PDA later, you need to define the PDA as an RFID reader in this network. To define the reader, do the following:

1. Select **Readers** from the left pane, then click **New**, as shown in Figure 13:
Figure 13. Readers dialog

In the Create a New Reader dialog, do the following, as shown in Figure 14:

- In the **Reader Id** field, specify a unique ID to identify your PDA.
- In the **Reader Type** field, select **GpsNmeaType**.
- In the **Reader IP address** and **Reader IP port number** fields, specify the reader's IP address and port number that the GPS device outputs the location data on the PDA. Since the GPS device is running on the PDA, the IP address should be the IP address of the PDA. The GPS adapter you created will connect to the IP and port later.
- Click **Create**.
Figure 14. Create a new Reader dialog

The new reader, R2, is created and displayed in the Reader dialog, as shown in Figure 15:
Figure 15. The Reader dialog showing the new reader R2

Associate the location and the reader with the Edge Controller

Next, you need to associate the location and the reader with an Edge Controller, by doing the following:

1. Select **Controller** from the left pane.
2. In the Controllers dialog, select the controller ID **E2** that you created previously, as shown in Figure 16:
3. In the Edit Controller Details dialog, do the following, as shown in Figure 17:
   • Select **R2** from the Available Locations list and click the right arrow. R2 will display in the Selected Locations list.
   • Select **R2** from the Available Readers list and click the right arrow. R2 will display in the Selected Readers list.
   • Click **Update** to update the Edge Controller configuration database.
Update the filter agent setting

In order for the GPS adapter to work properly, you need to modify the filter agent setting, because you only load the CaseTags filter agent when deploying the applications. (Refer to RfidIbmTutorialReaderSimulatorLoader project => com.ibm.esc.rfid.ibm.tutorial.reader.simulation.loader.bundle package => Activator.java for more details.)

To access and modify the agent properties, do the following:

1. Select **Agent configuration** from the left pane.
2. In the Edit Agent Properties dialog, do the following, as shown in Figure 18:
   - In the **Reader Agent** field, select **FilterAgent**.
   - In the **Agent Property** field, select **filters**.
   - In the **Property Value**, specify **CaseTags**.
   - Click **Update** to store the changes to the configuration database.
Verify the network topology configuration data

At this point, you should verify the Edge Controller configurations. To do this, check whether you can obtain valid XML when browsing to the following URL:

http://<hostname>:9080/event_admin_web/premises.sl?action=getconfig&edge=<EdgeId>

where <hostname> is the Premises Server DNS name and <EdgeId> is the ID of the Edge Controller you created. In our case, the URL should look like: http://9.191.74.241:9080/event_admin_web/premises.sl?action=getconfig&edge=E2. The content of the XML file is similar to edge-configuration-E2.xml used in Part I. You can see that GpsNmeaReaderAgent is shown as one of the reader agents. The XML also contains the IP address and port to receive the GPS data as shown in Figure 19:
Run the Service Management Framework (SMF) on the Premises Server

To view the logs shown on the console of the SMF on the Premises Server, you need to stop the Windows IBM SMF Service and start the SMF. To do this:

1. Open a command prompt and execute smf.bat in the directory `<RFID_HOME>\edgecontroller\premises\smf` to start the SMF service. Press Enter twice. The SMF prompt is displayed as shown in Figure 20:

Figure 20. Run SMF

2. From the SMF command prompt in the window where you started SMF, type `ss` to list the installed bundles as shown in Figure 21. Note that the status of the KimonoPremisesTest bundle is RESOLVED. This bundle is a test program that listens to the MQ queue (CNTROL.OUT.QUEUE), which stores all the data sent from the Edge Controller. It retrieves the data from the queue and then prints it out to the SMF console. You need to start this bundle to verify whether the GPS data is being sent to MQ correctly. In a real solution, you could not start the bundle. Instead, you would need to write a program, such as a Message-Driven Bean (MDB) run on WebSphere Application Server, to retrieve the data from CONTROL.OUT.QUEUE. Before starting the KimonoPremisesTest bundle, you need to revise
its property file, `<RFID_HOME>\edgecontroller\premises\smf \premises-test.properties` to comment out the two lines shown in Listing 2:

**Listing 2. Modify premises-test.properties**

```
# Properties for Activator.java
#
# To disable a particular component in its entirety, comment out the
# appropriate line.
#
premises.test.factory.class.0=test.com.ibm.kimono.premises.
  event.handler.EventHandlerFactory
#
premises.test.factory.class.1=test.com.ibm.kimono.premises.command.CommandFactory
#
premises.test.factory.class.2=test.com.ibm.kimono.
  premises.event.simulator.EventSimulatorFactory
```

3. Start the KimonoPremisesTest bundle by typing `start 25`, as shown in Figure 21:

**Figure 21. Start the KimonoPremisesTest bundle service**

```
smf> start 25
```

4. From the SMF command prompt in the window where you started SMF, type `ss` to list the installed bundles. This time, you should see that The KimonoPremisesTest bundle service (bundle service 25) status is now ACTIVE as shown in Figure 22:
Create a runtime for WebSphere RFID Device Infrastructure

In this section, you'll create a WebSphere RFID Device Infrastructure runtime, then deploy and run it on a Windows Mobile 2003 PDA using WebSphere Studio Device Developer (hereafter called Device Developer).

To create a runtime by using the Device Developer platform builder, do the following:

1. From the workbench, select **File => New => Other**.
2. In the Select dialog, select **Extension Services** on the left, **Platform Builder** on the right, and then click **Next**, as shown in Figure 23:
Figure 23. Create a new platform builder project

3. In the New Platform Builder Project dialog, enter WM2003Edge as the Project name and click Next, as shown in Figure 24:
4. The Platform Options dialog defines the primary characteristics of the Platform Builder project. It contains sections dealing with the project's device, JVM and build output options. Specify the device type and set package output location as shown in Figure 25:
   - For **Device Type**, select **PocketPC**, which is suitable for Windows Mobile 2003 PDA.
   - Check **Default target root**.
   - Enter `c:\` as the **Output location**. (Or, choose another output location, if desired.)
   - Click **Next**.
5. In the J9 Options dialog, specify the characteristics of the J9 Java™ Virtual Machine (JVM) as shown in Figure 26:
   - Select jclFoundation for J9 class library.
   - Select JAR for Set Package format.
   - Click Next.
6. In the Startup Options dialog, do the following:
   • Check **Console support** to include the SMF console in the target platform.
   • Check **Bundle Developer support** to allow Device Developer to connect to the target platform runtime.
   • Specify a **Port**. For our example, you can use the default port of 1457.
   • Specify the **Bundle Server address** and **Port** to allow the target platform to communicate with an SMF bundle server. Here, the bundle server will run on the Device Developer machine, so enter its IP and bundle server default port (8080).
   • Click **Next**.
7. In the Languages dialog, select the languages that the target platform will support. For this example, select **English**, then click **Next**, as shown in Figure 28:
8. Since we aren't going to add any custom bundles in this exercise, click **Next** in the Custom Bundles dialog, and then in the System Bundles dialog.

9. In the Summary dialog, you'll see a summary of all the target platform options you selected. Make sure that **Build platform** is checked and click **Finish**, as shown in Figure 29:
10. When the build is complete, you’ll see BUILD SUCCESSFUL in the console window, as shown in Figure 30:

**Figure 30. The console log for generating the platform package**

11. Verify that the target platform package is placed in the output directory you specified. The file name of the target platform is platform.zip. In this case, set the output location to directory c:\ as shown in Figure 31:
Deploy the platform package to the PDA

Before copying the platform package file to the PDA, make the following changes:

1. Unzip the platform.zip file. You’ll see two directories: smf and jvm.
2. Modify the /smf/StartSMF.lnk file to add -noverify -Xint as shown in Listing 3. (Originally, we planned to add -noverify -Xint -Dpremises.ip=9.191.74.241:9080 -Dedge.id=E2, but the length of the Windows Mobile 2003 .lnk file can’t be over 256 characters. Therefore, we decided to move -Dpremises.ip=9.191.74.241:9080 -Dedge.id=E2 to a property file, premisesXML.property, as described in Update the code.

Listing 3. The /smf/StartSMF.lnk file

```
17""jvm\bin\j9.exe" -jcl:foun10 -Xalwaysclassgc -Xgc:stringconstantGC
-noverify -Xint -cp
"\smf;\smf;\smfconsole.jar;\smf;\smfbd.jar"
-Dcom.ibm.osgi.smf.properties=\smf;\smf.properties
com.ibm.osgi.smf.SMFLauncher -con -ide:1457 "launch"
```

3. Use ActiveSync to copy both the jvm and smf directories to the PDA root directory. Since we checked Default target root previously, make sure to copy to the root directory of the PDA. If you copied to other directories, the SMF won’t be started.

Run the Service Management Framework on the PDA

Before you start the SMF on the PDA, do the following:

1. Make sure the PDA WLAN/LAN is working properly.
2. Make sure that the Bluetooth GPS is connected to the PDA.
3. Because we expect the client to access GPS via TCP/IP, make sure you have a tool to, such as Franson GpsGate to convert the Bluetooth interface to a TCP/IP service. Make sure that you can access the GPS location data via TCP/IP.
4. From Windows Explorer, start SMF by clicking StartSMF.lnk in the /smf directory. The runtime should be launched successfully as shown in Figure 32:
Develop and deploy the application to the PDA with Device Developer

In this section, you'll make necessary code changes and then deploy the application to the PDA.

Update the code

1. Start Device Developer and select the workspace used in Part I (C:\IBM\wsdd571\workspace\gps), as seen in Figure 33:

   Figure 33. Select a workspace

2. Select RfidIbmTutorialReaderSimulatorLoader project => com.ibm.esc.rfid.ibm.tutorial.reader.simulation.loader.bundle package => Activator.java. Comment out KimonoSwitchAgent and KimonoMotionSensorAgent as shown in Listing 4, so that you can start to retrieve GPS data once the GpsNmeaReaderAgent is started, rather than relying on the switch and motion sensor to trigger the GpsNmeaReaderAgent to start retrieving the GPS data:

   Listing 4. Comment out switch and motion sensor

   //developerWorks Part II Don't need Switch "KimonoSwitchAgent", //$NON-NLS-1$
   //developerWorks Part II Don't need Motion Sensor "KimonoMotionSensorAgent", //$NON-NLS-1$

3. Select GpsNmeaReaderAgent project => com.ibm.gps.nmea.gpsnmea.readeragent package => GpsNmeaReaderAgent.java. In this file, you need to make some changes to the original code.
tutorialTransport.setConnection(tutorialTransport.getDefaultConnection());
creates the connection to the GPS device through the IP and port that are defined
in the CML file (RfidIbmTutorialReaderDevice.cml). However, in this case, we want
to use the IP and port specified in the configuration XML downloaded from the
Premises Server, so that we can easily use the Premises Server administrative
console to change the IP and port of the GPS device. Therefore, we'll use
tutorialTransport.setConnection(buildDefaultTcpipConnection((EscObject)
readerDevice)); instead.

Listing 5. Change connection method

protected DeviceService createDevice() {
    LogUtility.logInfo(this,"createDevice()";
    ConnectionService connection;
    DeviceService readerDevice = new RfidIbmTutorialReaderDevice(); // Here
    RfidIbmTutorialReaderTransport tutorialTransport = new RfidIbmTutorialReader
    Transport(); // Here
    //Get the connection type (TCP/IP or COM) from the
    //edge-configuration-E1.xml downloaded from the
    //premises server or premises simulator. We will use TCP/IP.
    String connectionType = getProperty(GpsNmeaReaderAgentConstants.
    GSPNMEA_CONNECTION_TYPE);
    //Although we define the host and remoteport in the transport CML file,
    //we will use the host and port defined in edge-configuration-E1.xml.
    //Part II fixed part I bug
    tutorialTransport.setConnection(tutorialTransport.getDefaultConnection());
    // dWork
    tutorialTransport.setConnection(buildDefaultTcpipConnection((EscObject)
    readerDevice));
    readerDevice.setTransport(tutorialTransport);
    tutorialTransport.start();
    return readerDevice;
}

4. Select KimonoEdgeConfiguration project => com.ibm.kimono.edge.configuration
package > EdgeConfiguration.java. Move the Premises IP and Edge ID configurations
from VM arguments to a property file, premisesXML.properties, because of the 256-character
command line length limitation of Windows Mobile 2003.

protected boolean doConfigurationWork() throws InterruptedException {
    try {
        Configuration config = configAdmin.getConfiguration
        (SystemConstants.EDGE_CONFIG_ADMIN_PID);
        Dictionary properties = config.getProperties();
        //developerWorks part II added start...
        //Read the Premises Server IP and edge ID from premisesXML.properties
        //because startSMF.lnk file in Windows Mobile 2003 restricts the max
        //number of characters to 256.
        //If we add -Dpremises.ip=172.16.136.193 -Dedge.id=E1 as the j9 arguments,
        //it will exceed 256 characters, so we move the premises server IP and
        //edge server id to premisesXML.properties.
        try{
            ResourceBundle rs= ResourceBundle.getBundle
            ("com.ibm.kimono.edge.configuration.premisesXML");
            String ip = rs.getString("premises.ip");
            String edge = rs.getString("edge.id");
            System.out.println("---------------------Get Premises Server IP
            & Edge ID from premisesXML.properties---------------------");
            System.out.println("---------------------Get Premises Server IP=
            "+ip+"---------------------");
            System.out.println("---------------------Get Edge ID=
            "+edge+"---------------------");
        } catch (IOException e) {
            e.printStackTrace();
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
}
5. Select **KimonoEdgeConfiguration project** => **com.ibm.kimono.edge.configuration** package => premisesXML.properties. If your Premises Server does not run on HTTP port 80, add the port number to premises.ip. In our example, the Premises Server is running on port 9080.

**Listing 7. Add port number to premises.ip**

```java
premises.ip=9.191.74.241:9080
edge.id=E2
```

6. Select **KimonoControllerAgent project** => **com.ibm.kimono.controlleragent** package => **ControllerAgent.java**. Since we don't run the light tree in our application, comment out the code used to publish light tree controlling commands. Otherwise, many light tree controlling commands might be in the MicroBroker, which could cause memory leak problems.

**Listing 8. Comment out light tree controlling commands**

```java
private void handleTopicPalletFeedback(String topic, Object data) {
    String value = (String) data;
    if (ControllerAgentConstants.PALLET_FEEDBACK_ACCEPTED.equals(value)) {
        //Part II: We don't have light tree so don't need to publish the data to
        //control light tree.
        //doPublishWithLogging(lightAcceptedTopic, SystemConstants.ON);
        LogUtility.logInfo(this, "Tag Accepted.");  //$NON-NLS-1$
    } else if (ControllerAgentConstants.PALLET_FEEDBACK_REJECTED.equals(value)) {
        //Part II: We don't have light tree so don't need to publish the data to
        //control light tree.
        //doPublishWithLogging(lightRejectedTopic, SystemConstants.ON);
        LogUtility.logInfo(this, "Tag Rejected.");  //$NON-NLS-1$
    } else {
        throw new RuntimeException("unhandled value"); //$NON-NLS-1$
    }
}
```

7. Select **RfidIbmTutorialReaderDevice project** => **RfidIbmTutorialReaderDeviceDevelopment folder** => **RfidIbmTutorialReaderDevice.cml**. Remove the `type="numericstring"` attribute of the parameter tag in the CML. The reason for this is that the GPS device sometimes returns empty data fields for each NMEA ASCII response. If you don't remove these attributes, you may receive some Java exceptions (NumberFormatException).
8. Regenerate the Java code using the revised \texttt{RfidIbmTutorialReaderDevice\.cml}, by right-clicking \texttt{RfidIbmTutorialReaderDevice\.cml}, then selecting \texttt{Device Kit => Generate}, as shown in Figure 34:

\textbf{Figure 34. Generate the Java code for RfidIbmTutorialReaderDevice\.cml}

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{figure34.png}
\caption{Generate the Java code for RfidIbmTutorialReaderDevice\.cml}
\end{figure}

\textbf{Start the SMF bundle server}

1. Select \texttt{Run => Run}.
2. In the Run dialog, expand \texttt{SMF Bundle Server} in the left navigation, select \texttt{SMF Bundle Server}.
3. Accept the defaults and click \texttt{Run}, as shown in Figure 35:
Submit the bundles to the SMF bundle server

Now you need to submit the bundles containing the revised files to the SMF bundle server, by doing the following:

1. Right-click **GpsNmeaReaderAgent** in the Package Explorer view of the SMF perspective, then select **SMF => Submit Bundle**, as shown in Figure 36:
2. In the Target Submission dialog, check **Submit Jar**, select **Admin@localhost:8080/smf**, check **Replace Bundles**, then click **Finish** as shown in Figure 37:
3. Repeat steps 1-3 for the KimonoControllerAgent, KimonoEdgeConfiguration, RfidIbmTutorialReaderDevice, and RfidIbmTutorialReaderSimulationLoader projects.

**Deploy the bundles to the Windows Mobile 2003 PDA**

Use Device Developer to deploy all the necessary bundles to the SMF runtime on the PDA. Before starting the deployment, make sure that:

- The PDA can connect to the network and get an IP address. Check this by pinging the PDA from Device Developer machine.
- The Bluetooth GPS device can connect to the PDA.
- The tool that converts the Bluetooth GPS to TCP/IP (such as GpsGate 2.0) can run on the PDA, so that GpsNmeaDeviceAgent can get the GPS data when connecting to the TCP port.
- The SMF runtime on the PDA is running.

Once you've verified all of the above, deploy the bundles to the PDA SMF runtime by doing the following:

1. In Device Developer, select Run => Run.
2. In the Run dialog, right-click **Remote SMF Runtime** and select **New**, as shown in Figure 38:
3. A remote SMF runtime is created under **Remote SMF Runtime** in the left pane. Enter the IP address for the PDA in the **Host name** field. Leave the port as 1457, then click **Run**, as shown in Figure 39:

**Figure 39. Run remote SMF runtime connecting to the PDA SMF runtime**

Device Developer connects to the PDA SMF runtime as shown in Figure 40:
4. Switch to the SMF Bundle Servers view by clicking the **SMF Bundle Servers** tab at the bottom of the left pane, as shown in Figure 41.

5. Expand the Bundle Server tree, right-click on **RFID_IBM_TUTORIAL_READER_SIMULATION_LOADER**, and select **Install**.
6. After all the bundles are deployed to the PDA SMF runtime, you'll see the GPS data on the PDA screen. You'll also see that the GPS data has already been sent to the Premises Server and idumped on the SMF console as shown in Figure 42:

**Figure 42. GPS data sent to the Premises Server and dumped on the SMF console**
Summary

In this series, we've shown you how to develop an end-to-end GPS data collector solution using IBM RFID middleware. First, you learned how to create a device adapter on WebSphere RFID Device Infrastructure using the RFID Device Development Kit and test it with simulators. Next, you learned how to define the network topology (the WebSphere RFID Device Infrastructure configurations) using the Premises Server administrative console. Finally, you learned how to build a runtime platform for a device adapter and deploy it to a PDA. As you could see in the GPS adapter example, the GPS adapter was able to continually retrieve location data from the GPS device on the PDA, and send the collected data to the Premises Server seamlessly with the WebSphere MicroBroker.

As mentioned in Part I, you could also deploy the sample code to On Board Units (OBUs) to interact with GPS devices in cars or trucks, and then send the location information back to a data center. You can also use the concepts and process we used here to develop device adapters and agents for any sensors that could be deployed to OBUs, such as temperature sensors, RFID readers, anti-theft sensors, and so on. IBM RFID middleware makes it easy to develop complex data collection solutions.
## Downloadable resources

<table>
<thead>
<tr>
<th>Description</th>
<th>Name</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sample code for this article</td>
<td></td>
<td>16KB</td>
</tr>
</tbody>
</table>

© Copyright IBM Corporation 2006

Trademarks
(www.ibm.com/developerworks/ibm/trademarks/)