Wrapper for WebSphere Application Server 5.1 on Linux

Document version 1.0

Har M Puri
Staff Software Engineer
IBM Integrated Runtime development
Rochester MN 55901

Contents

Figures

Introduction

The IBM Integrated Runtime (IR) product contains WebSphere Application Server Express Version 5.1. Some of the larger Independent Software Vendors (ISV), Solution Developers (SD), and Solution Integrators (SI) want a mechanism to use the IBM IR deployment framework to deliver WebSphere Application Server 5.1 (WAS) and take advantage of the increased capability of the base WAS product or its licensing terms.

This document is targeted at the ISV or other developer who wants to create a wrapper of the base WAS product for use in a solution. This application wrapper is developed on the Linux platform. It can be developed on the Windows platform with minor changes.

To keep this component as simple as possible, the IBM HTTP Server (IHS), which is included with the WebSphere Application Server 5.1 product, is used. The disadvantage of this approach is that the deployer cannot target separate machines for the IHS and WebSphere Application Server installations. If this approach presents a problem, the solution developer can chose to use the IHS Integrated Runtime component and disable installation of the IHS in the WebSphere Application Server response file.

All operations must be done from the `root` login id.

Methodology

This paper is separated into several sections to assist in completing the following tasks:

- Modifying a response file.
- Building the wrapper.
- Testing the wrapper.

Each section might include the following components:

- **Description** – A general description of the section or exercise.
- **Interactions** – The dependencies of the component in the solution, and the products that are depending on the component. Descriptions of interactions between other components within the solution and the subject component.
- **Control** – How to start and stop the component, and the methods that you use to determine if the component is running.

- **Configuration** – A description of the methods that you use to configure the component.
- **Problem Determination** – How to determine the causes of problems with the component.
- **Additional Information** – Where to find additional information about the component.
- **Exercises** – Hands-on exercises with the component. Exercises are separated into four types:
  - Installation Exercises – The components are installed in these exercises.
  - Configuration Exercises – The components are configured in these exercises.
  - Control Exercises – The components are controlled in some way in these exercises.
  - Optional Exercises – Some additional features or functions of the components are used in these exercises.

Conventions

This document uses the following conventions to illustrate various types of information.

- All exercises are numbered, and each step in the exercise is numbered. It is important to complete the steps and exercises in the order they are provided. If an exercise or step is not required, it is indicated as optional.
- Throughout the exercises, informational text is formatted like this current paragraph.
- Commands are shown in the following font and format and should be typed exactly as shown, including letter case:

```
Type this exactly - This command should be typed as though on a single command line. If
part of the line is already present, do not type it. For example the drive d:\> may already
be displayed in the command prompt and there is no need to type it again.

install.bat d:\
```

- Code samples are show in the following font and format.

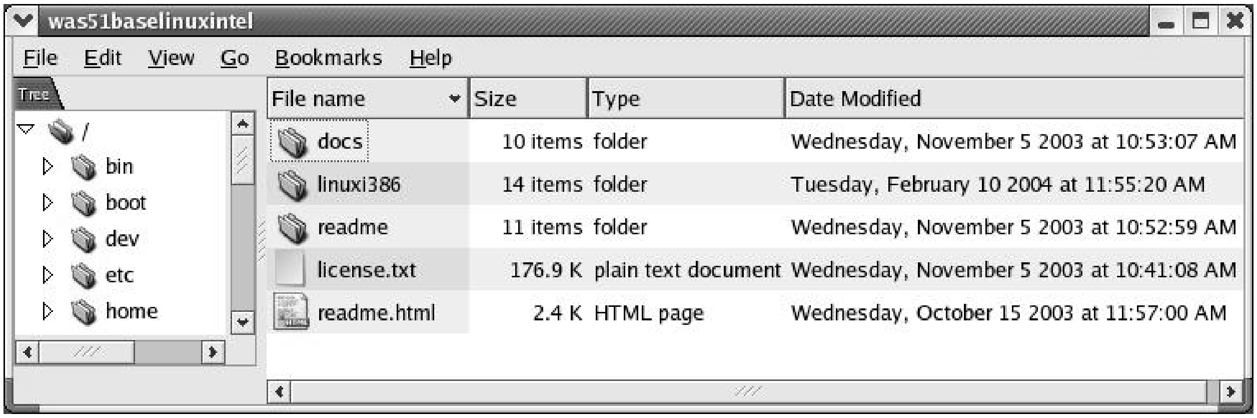
```
<html>
<body>
<p>This is sample code.</p>
</body>
</html>
```

WebSphere Application Server CD

The source for creating the WebSphere Application Server application is the WebSphere Application Server product CD that you can purchase from IBM through the Partner World O.E.M sales channel. You must purchase an appropriate license to bundle WebSphere Application Server with a solution. Please contact a channel sales representative for licensing issues.

CD Contents

The WebSphere Application Server CD could contain a TAR file named c53ipml.tar (318,648,320 bytes) or have contents and directory structure similar to the following illustration:



File name	Size	Type	Date Modified
docs	10 items	folder	Wednesday, November 5 2003 at 10:53:07 AM
linuxi386	14 items	folder	Tuesday, February 10 2004 at 11:55:20 AM
readme	11 items	folder	Wednesday, November 5 2003 at 10:52:59 AM
license.txt	176.9 K	plain text document	Wednesday, November 5 2003 at 10:41:08 AM
readme.html	2.4 K	HTML page	Wednesday, October 15 2003 at 11:57:00 AM

Figure 1 WebSphere Application Server CD root directory

If the CD has the directory structure as illustrated in figure 1, create a TAR file using the following command:

```
tar cvf WAS5.1BaseLinuxIntel_C53IPML.tar /mnt/cdrom
```

If you have the TAR file on CD, rename it to from c53ipml.tar to WAS5.1BaseLinuxIntel_C53IPML.tar. For the remainder of this document, this product image file is refrerred to as WAS5.1BaseLinuxIntel_C53IPML.tar.

A TAR file is used for this application because of the presence of symbolic links in the installation image or CD.

Response file

A WebSphere Application Server response file installation helps you install WebSphere Application Server without interactive user input. Response files contain installation choices and configuration data that would otherwise be provided by the user interactively. A response file automates the process and helps you to embed the WebSphere Application Server installation process transparently inside your own installation program. The response file installation is one of the ways WebSphere Application Server provides a complete solution for your integrated application server needs.

A sample response file responsefile.txt is available on the WebSphere Application Server installation CD in the linuxi386/ directory.

```
# *******************************************
#
# Response file for WebSphere Application Server 5.1 Install
#
# Please follow the comments to use the response file and
# understand the various options.  You must carefully complete
# or change the various values. If the values are not completed
# properly, the install may be unsuccessful.
#
# NOTE: This file is for silent install only.
#
#
# IMPORTANT: ALL VALUES MUST BE ENCLOSED IN DOUBLE QUOTES ( "" ).
#
# *******************************************

# *******************************************
# This value is required.
# PLEASE DO NOT REMOVE THIS VALUE.
# *******************************************

-W setupTypes.selectedSetupTypeId="custom"


# *******************************************
# Below is the beginning of the response file that needs to be
# filled in by the user.
# *******************************************

# *******************************************
# The below value specifies silent install. This value
# indicates that the install will be silent.
# *******************************************

-silent


# *******************************************
# WebSphere Application Server Install Location
#
# Please specify the destination directory for the WebSphere Application
# Server installation.   You will need to change this for UNIX
# platforms. As an example for AIX, the value may be
# "/usr/WebSphere/AppServer"
# *******************************************

-P wasBean.installLocation="C:\Program Files\WebSphere\AppServer"


# *******************************************
```

Figure 2 Partial Contents of a generated response file

The first step of the response file installation process is to modify the response file. When installing WebSphere Application Server using the interactive setup utility, you are prompted to make installation choices and provide necessary configuration data before installation takes place. Using the following instructions you can create a response file that contains the choices you would make when you use the installation GUI. Later, during the application development, you can encode a process to replace the response file variables (for example, *WebSphere Application Server node*) with text that you can use to prompt the user during the deployment process.

Modify the response file

1. Copy responsefile.txt from the linuxi386/ directory to your home directory. Rename it baseresponsefile.txt.
2. Modify baseresponsefile.txt to disable the installation of the embedded JMS server and change the paths to suit a Linux environment. The following table shows the changes that should be made. Please note that in the interest of making some strings fit in the table partial strings can be used in the original column.

Original	Change to
-P wasBean.installLocation="C:\Program Files\WebSphere\AppSer	-P wasBean.installLocation="/opt/WebSphere/AppServer"
-P ihsFeatureBean.installLocation="C:\Program Files\IBMHTTPSe	-P ihsFeatureBean.installLocation="/opt/IBMHTTPServer"
-P mqSeriesBean.active="true"	-P mqSeriesBean.active="false"
-P mqSeriesServerBean.active="true"	-P mqSeriesServerBean.active="false"
-P mqSeriesServerBean.installLocation="C:\Program Files\IBM\W …	#-P mqSeriesServerBean.installLocation="C:\Program Files\IBM\...
-P mqSeriesClientBean.installLocation="C:\Program Files\IBM\W …	#-P mqSeriesClientBean.installLocation="C:\Program Files\IBM\...
-P mqSeriesSamplesBean.active="true"	-P mqSeriesSamplesBean.active="false"
-P samplesBean.active="true"	-P samplesBean.active="false"
-P performanceAndAnalysisToolsBean.active="true"	-P performanceAndAnalysisToolsBean.active="false"
-P tivoliPerfBean.active="true"	-P tivoliPerfBean.active="false"
-P DCMBean.active="true"	-P DCMBean.active="false"
-P performanceServletBean.active="true"	-P performanceServletBean.active="false"
-P logAnalyzerBean.active="true"	-P logAnalyzerBean.active="false"
-P javadocBean.active="true"	-P javadocBean.active="false"
-W serviceSettingsWizardBean.active="true"	#-W serviceSettingsWizardBean.active="true"
-W serviceSettingsWizardBean.ihsChoice="true"	#-W serviceSettingsWizardBean.ihsChoice="true"
-W serviceSettingsWizardBean.wasChoice="true"	#-W serviceSettingsWizardBean.wasChoice="true"
-W serviceSettingsWizardBean.userName="YOUR_USER_NAME"	#-W serviceSettingsWizardBean.userName="YOUR_USER_NAME"
-W serviceSettingsWizardBean.password="YOUR_PASSWORD"	#-W serviceSettingsWizardBean.password="YOUR_PASSWORD"
-P StartServerIconBean.active="true"	-P StartServerIconBean.active="false"
-P StopServerIconBean.active="true"	-P StopServerIconBean.active="false"
-P AdminConsolIconBean.active="true"	-P AdminConsolIconBean.active="false"
-P SamplesGalleryIconBean.active="true"	-P SamplesGalleryIconBean.active="false"
-P TivoliPerfIconBean.active="true"	-P TivoliPerfIconBean.active="false"
-P infoCenterIconBean.active="true"	-P infoCenterIconBean.active="false"
-P firstStepsIconBean.active="true"	-P firstStepsIconBean.active="false"
-P logAnalyzerIconBean.active="true"	-P logAnalyzerIconBean.active="false"
-W defaultIHSConfigFileLocationBean.value="C:\Program Files\I …	-W defaultIHSConfigFileLocationBean.value= "/opt/IBMHTTPServer/conf/httpd.conf"
#-W coexistenceOptionsBean.doCoexistence="true"	-W coexistenceOptionsBean.doCoexistence="true"
#-W coexistencePanelBean.useIhs="true"	-W coexistencePanelBean.useIhs="true"
#-W coexistencePanelBean.bootstrapPort="2810"	-W coexistencePanelBean.bootstrapPort="2809"
#-W coexistencePanelBean.ihsPort="81"	-W coexistencePanelBean.ihsPort="80"
#-W coexistencePanelBean.httpTransportPort="9086"	-W coexistencePanelBean.httpTransportPort="9080"
#-W coexistencePanelBean.httpsTransportPort="9044"	-W coexistencePanelBean.httpsTransportPort="9043"
#-W coexistencePanelBean.adminConsolePort="9091"	-W coexistencePanelBean.adminConsolePort="9090"
#-W coexistencePanelBean.secureAdminConsolePort="9444"	-W coexistencePanelBean.secureAdminConsolePort="9443"
#-W coexistencePanelBean.soapConnectorAddress="8881"	-W coexistencePanelBean.soapConnectorAddress="8880"

Figure 3 Changes to the default response file

3. If you want to expose additional parameters, they can be uncommented too, but you have to provide defaults or modify additional files to accept user input during deployment.
4. This process leaves a default response file (baseresponsefile.txt) in the /root directory.

Creating the WebSphere Application Server application

In IBM Integrated Runtime terminology, an *application* is one of the components of a *solution*. Examples of applications include WebSphere Application Server or the actual application program package written by the developer. Applications are contained in *application wrappers*. One or more application wrappers are combined into a complete solution by creating a *solution wrapper*.

You must create an application wrapper for WebSphere Application Server and then package it into a test solution. Since the wrapper for the WebSphere Application Server Express Edition is very similar to the WebSphere Application Server Workgroup Server Edition it makes a good starting point. You can modify a copy to make a wrapper for WebSphere Application Server.

Copy the WebSphere Application Server Express project

1. Start the Solution Developer by selecting **Start >IBM Integrated Runtime > Solution Developer**.

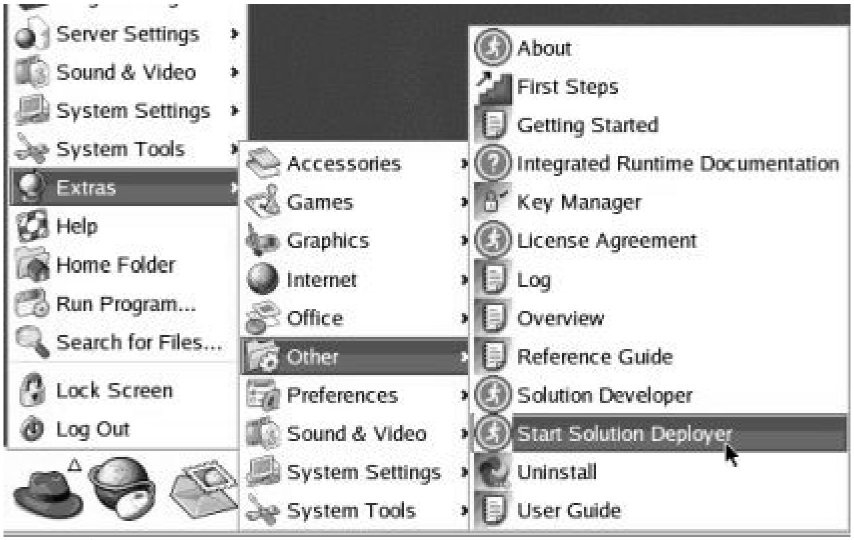
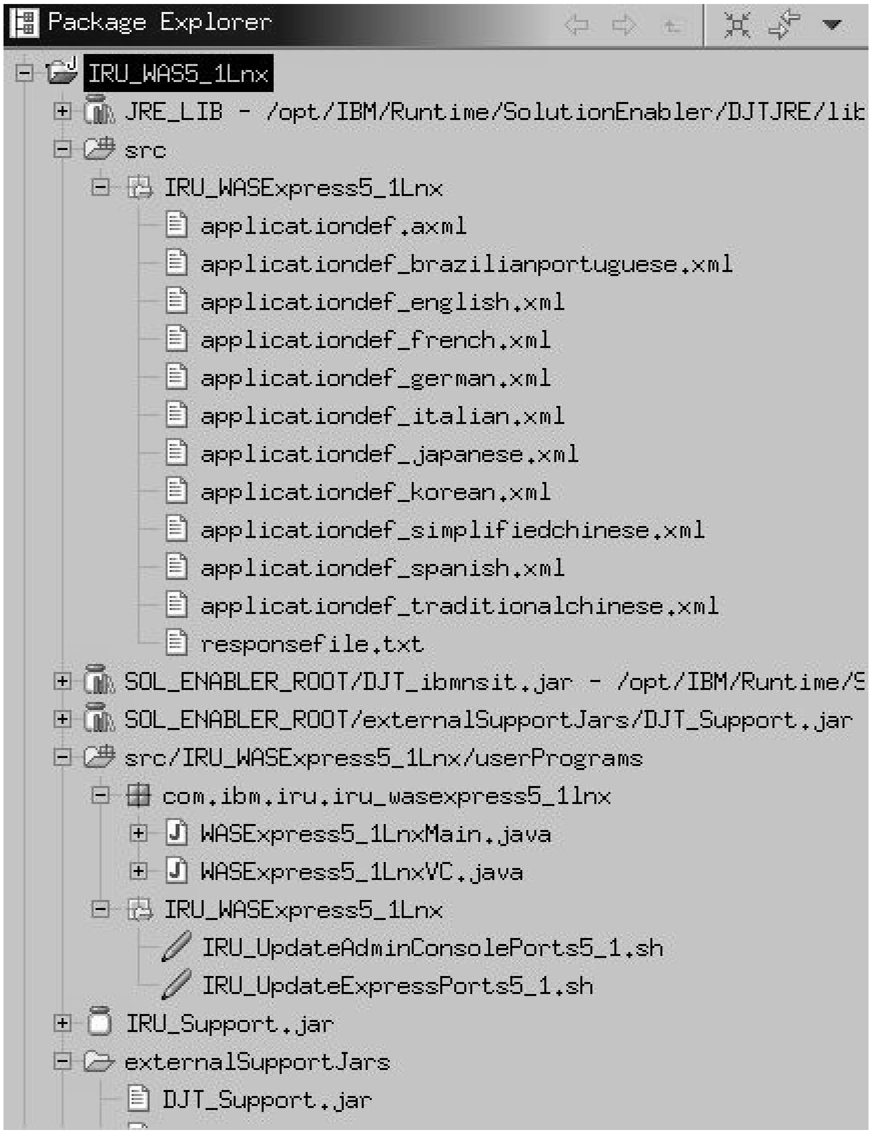


Figure 4 Solution developer

2. Right-click IRU_WASExpress5_1Lnx and select **Copy.**
3. Right-click again and select **Paste**.
4. Enter the new name for the application in the dialog box. For the remainder of this document the name of the application is IRU_WAS5_1Lnx.
5. You have the following tree structure in Solution Developer:



```
Package Explorer
IRU_WAS5_1Lnx
  JRE_LIB - /opt/IBM/Runtime/SolutionEnabler/DJTJRE/lit
  src
    IRU_WASExpress5_1Lnx
      applicationdef.axml
      applicationdef_brazilianportuguese.xml
      applicationdef_english.xml
      applicationdef_french.xml
      applicationdef_german.xml
      applicationdef_italian.xml
      applicationdef_japanese.xml
      applicationdef_korean.xml
      applicationdef_simplifiedchinese.xml
      applicationdef_spanish.xml
      applicationdef_traditionalchinese.xml
      responsefile.txt
  SOL_ENABLER_ROOT/DJT_ibmnsit.jar - /opt/IBM/Runtime/S
  SOL_ENABLER_ROOT/externalSupportJars/DJT_Support.jar
  src/IRU_WASExpress5_1Lnx/userPrograms
    com.ibm.iru.iru_wasexpress5_1lnx
      WASExpress5_1LnxMain.java
      WASExpress5_1LnxVC.java
    IRU_WASExpress5_1Lnx
      IRU_UpdateAdminConsolePorts5_1.sh
      IRU_UpdateExpressPorts5_1.sh
  IRU_Support.jar
  externalSupportJars
    DJT_Support.jar
```

Figure 5 Structure of the new project after the copy operation

Modify the IRU_WAS5_1Lnx project

Modify the copied files to fit the new project. The following operations need to be done:

1. Delete the response file `responsefile.txt` and copy the baseresponsefile.txt file from the /root directory, created during the previous step, to the same location.
2. Edit the `applicationdef.axml` file and make the following changes:
    a. Search and delete all occurrences of the string `Express`.
    b. Change the name of the response file from responseFileTemplate="responsefile.txt"> to responseFileTemplate="baseresponsefile.txt">. There should be three occurrences of responsefile.txt.

c. Change `<file>51_LinuxPruned.tar</file>` to `<file>WAS5.1BaseLinuxIntel_C53IPML.tar</file>`

d. Delete the following lines of code:

```
<file>IRU_WAS5_1Lnx/IRU_UpdateAdminConsolePorts5_1.sh</file>
    <file>IRU_WAS5_1Lnx/IRU_UpdatePorts5_1.sh</file>
```

e. Delete the exit program definition by deleting the following lines of code:

```
<!-- ************************************************************ -->
<!-- Specify an exit program to be run *after* the install.       -->
<!-- ************************************************************ -->
<exitProgram type="java" programName="com.ibm.iru.common.LicenseHandler"
logFile="LicenseHandler.log">
        <arguments>
                <argument>iru_was5_1lnx</argument>
                <argument>installDir</argument>
                <argument>LicenseHandler.log</argument>
                <argument>license</argument>
        </arguments>
</exitProgram>
```

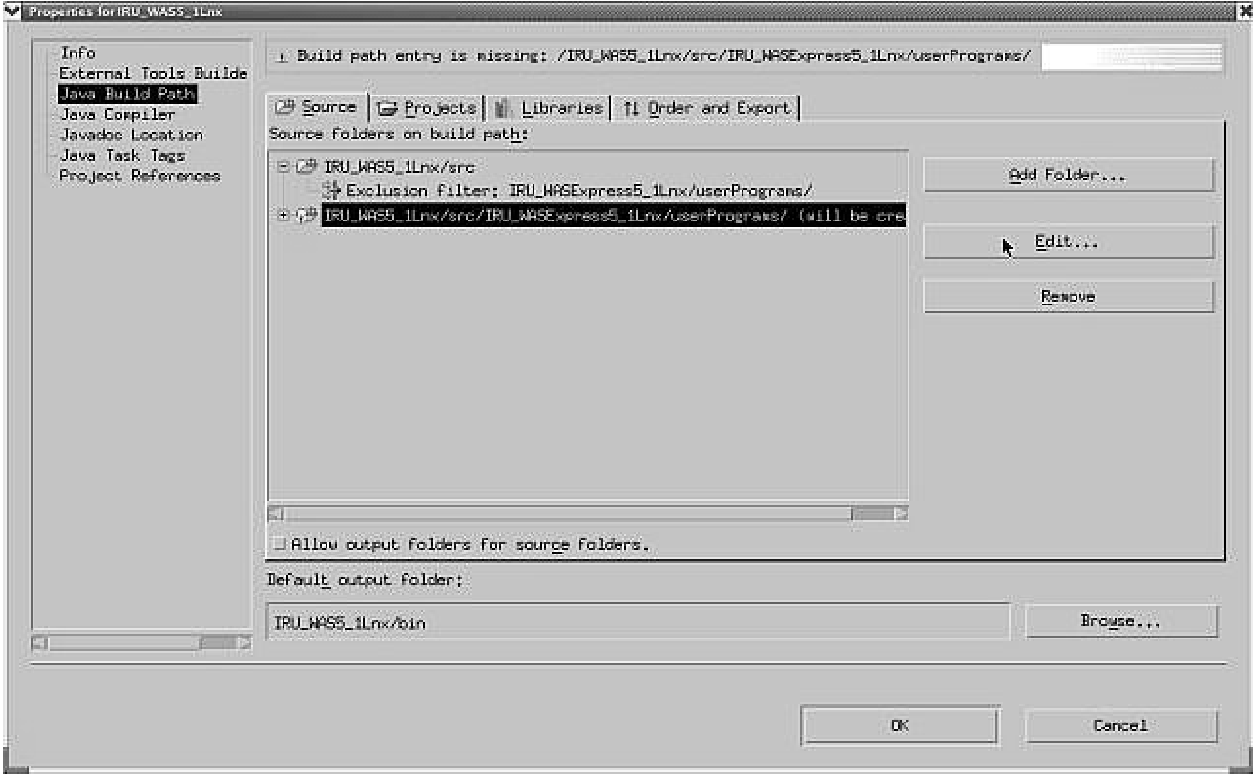
f. Add or modify a number of variables. These variables are displayed on the task details pane of the Solution Deployer. You have some discretion here on which variables you want the person who is deploying the solution to be able to change. For example, if you do not want the deployer to change any port assignments, do not define the variables here. Assign default values in the baseresponsefile.txt file. The entire variables section of the response file is shown in figure 6:

```
<!-- ********************************************************* -->
<!-- Specify a variable to expose the node name.       -->
<!-- ********************************************************* -->
<stringVariable name="nodeNameBean.nodeName" minimumLength="4">
        <defaultData>DefaultNode</defaultData>
        <labelText translatedKey="nodeLabel" />
        <helpText translatedKey="nodeHelp" />
        <propertiesAssociations>
                <propertiesAssociation keyword="nodeNameBean.nodeName" />
        </propertiesAssociations>
        <ismpFileAssociations>
                <ismpFileAssociation responseFileName="baseresponsefile.txt"
propertyKeyType="wizard" propertyKey="nodeNameBean.nodeName" />
        </ismpFileAssociations>
        <!-- ******************************************************* -->
        <!-- Specify validation rules for the node name.     -->
        <!-- ******************************************************* -->
        <inputValidation>
                <valid>
                        <characters
ignoreCase="true">abcdefghijklmnopqrstuvwxyz0123456789</characters>
                </valid>
        </inputValidation>
</stringVariable>

<!-- ********************************************************* -->
<!-- Specify a variable to expose the host name.       -->
<!-- ********************************************************* -->
<stringVariable name="nodeNameBean.hostName" minimumLength="4">
        <defaultData>localhost</defaultData>
        <labelText translatedKey="hostLabel" />
        <helpText translatedKey="hostHelp" />
        <propertiesAssociations>
                <propertiesAssociation keyword="nodeNameBean.hostName" />
        </propertiesAssociations>
        <ismpFileAssociations>
                <ismpFileAssociation responseFileName="baseresponsefile.txt"
propertyKeyType="wizard" propertyKey="nodeNameBean.hostName" />
        </ismpFileAssociations>
        <!-- ******************************************************* -->
        <!-- Specify validation rules for the host name.     -->
        <!-- ******************************************************* -->
        <inputValidation>
                <valid>
                        <characters
ignoreCase="true">.abcdefghijklmnopqrstuvwxyz0123456789</characters>
                </valid>
        </inputValidation>
</stringVariable>

<!-- ********************************************************* -->
<!-- Specify a variable to expose the install directory.       -->
<!-- ********************************************************* -->
<stringVariable name="wasBean.installLocation" minimumLength="4">
        <defaultData>/opt/WebSphere/AppServer</defaultData>
        <labelText translatedKey="dirLabel" />
        <helpText translatedKey="dirHelp" />
        <propertiesAssociations>
                <propertiesAssociation keyword="wasBean.installLocation" />
        </propertiesAssociations>
        <ismpFileAssociations>
                <ismpFileAssociation responseFileName="baseresponsefile.txt"
propertyKeyType="product" propertyKey="wasBean.installLocation" />
        </ismpFileAssociations>
        <!-- ******************************************************* -->
        <!-- Specify validation rules for the install directory.     -->
        <!-- ******************************************************* -->
        <inputValidation>
                <valid>
                        <prefixes>
                                <prefix ignoreCase="true">/</prefix>
                        </prefixes>
                        <characters ignoreCase="true">/-
_.abcdefghijklmnopqrstuvwxyz0123456789</characters>
                </valid>
        </inputValidation>
</stringVariable>

<!-- ********************************************************* -->
<!-- Specify a variable to expose the bootstrap port.          -->
<!-- ********************************************************* -->
<stringVariable name="coexistencePanelBean.bootstrapPort" minimumLength="1" maximumLength="5">
        <defaultData>2809</defaultData>
        <labelText translatedKey="rmiPortLabel" />
        <helpText translatedKey="rmiPortHelp" />
        <propertiesAssociations>
                <propertiesAssociation keyword="coexistencePanelBean.bootstrapPort" />
        </propertiesAssociations>
        <ismpFileAssociations>
                <ismpFileAssociation responseFileName="baseresponsefile.txt"
propertyKeyType="wizard" propertyKey="coexistencePanelBean.bootstrapPort" />
        </ismpFileAssociations>
        <!-- ******************************************************* -->
        <!-- Specify validation rules for the bootstrap port.        -->
        <!-- ******************************************************* -->
        <inputValidation>
                <valid>
                        <ranges>
                                <range>1 to 65535</range>
                        </ranges>
                </valid>
        </inputValidation>
</stringVariable>

<!-- ********************************************************* -->
<!-- Specify a variable to expose the soap port.               -->
<!-- ********************************************************* -->
<stringVariable name="coexistencePanelBean.soapConnectorAddress" minimumLength="1"
maximumLength="5">
        <defaultData>8880</defaultData>
        <labelText translatedKey="soapPortLabel" />
        <helpText translatedKey="soapPortHelp" />
        <propertiesAssociations>
                <propertiesAssociation keyword="coexistencePanelBean.soapConnectorAddress" />
        </propertiesAssociations>
        <ismpFileAssociations>
                <ismpFileAssociation responseFileName="baseresponsefile.txt"
propertyKeyType="wizard" propertyKey="coexistencePanelBean.soapConnectorAddress" />
        </ismpFileAssociations>
        <!-- ******************************************************** -->
        <!-- Specify validation rules for the port.                   -->
        <!-- ******************************************************** -->
        <inputValidation>
                <valid>
                        <ranges>
                                <range>1 to 65535</range>
                        </ranges>
                </valid>
        </inputValidation>
</stringVariable>

<!-- ********************************************************** -->
<!-- Specify a variable to expose the http port.                -->
<!-- ********************************************************** -->
<stringVariable name="coexistencePanelBean.httpTransportPort" minimumLength="1"
maximumLength="5">
        <defaultData>9080</defaultData>
        <labelText translatedKey="httpPortLabel" />
        <helpText translatedKey="httpPortHelp" />
        <propertiesAssociations>
                <propertiesAssociation keyword="coexistencePanelBean.httpTransportPort" />
        </propertiesAssociations>
        <ismpFileAssociations>
                <ismpFileAssociation responseFileName="baseresponsefile.txt"
propertyKeyType="wizard" propertyKey="coexistencePanelBean.httpTransportPort" />
        </ismpFileAssociations>
        <!-- ******************************************************** -->
        <!-- Specify validation rules for the port.                   -->
        <!-- ******************************************************** -->
        <inputValidation>
                <valid>
                        <ranges>
                                <range>1 to 65535</range>
                        </ranges>
                </valid>
        </inputValidation>
</stringVariable>

<!-- ********************************************************** -->
<!-- Specify a variable to expose the https port.               -->
<!-- ********************************************************** -->
<stringVariable name="coexistencePanelBean.httpsTransportPort" minimumLength="1"
maximumLength="5">
        <defaultData>9443</defaultData>
        <labelText translatedKey="httpsPortLabel" />
        <helpText translatedKey="httpsPortHelp" />
        <propertiesAssociations>
                <propertiesAssociation keyword="coexistencePanelBean.httpsTransportPort" />
        </propertiesAssociations>
        <ismpFileAssociations>
                <ismpFileAssociation responseFileName="baseresponsefile.txt"
propertyKeyType="wizard" propertyKey="coexistencePanelBean.httpsTransportPort" />
        </ismpFileAssociations>
        <!-- ******************************************************** -->
        <!-- Specify validation rules for the port.                   -->
        <!-- ******************************************************** -->
        <inputValidation>
                <valid>
                        <ranges>
                                <range>1 to 65535</range>
                        </ranges>
                </valid>
        </inputValidation>
</stringVariable>


<!-- ********************************************************** -->
<!-- Specify a variable to expose the admin port.               -->
<!-- ********************************************************** -->
<stringVariable name="coexistencePanelBean.adminConsolePort" minimumLength="1" maximumLength="5">
        <defaultData>9090</defaultData>
        <labelText translatedKey="adminPortLabel" />
        <helpText translatedKey="adminPortHelp" />
        <propertiesAssociations>
                <propertiesAssociation keyword="coexistencePanelBean.adminConsolePort" />
        </propertiesAssociations>
        <ismpFileAssociations>
                <ismpFileAssociation responseFileName="baseresponsefile.txt"
```

```
propertyKeyType="wizard" propertyKey="coexistencePanelBean.adminConsolePort" />
        </ismpFileAssociations>
        <!-- ******************************************************** -->
        <!-- Specify validation rules for the port.                   -->
        <!-- ******************************************************** -->
        <inputValidation>
                <valid>
                        <ranges>
                                <range>1 to 65535</range>
                        </ranges>
                </valid>
        </inputValidation>
</stringVariable>

<!-- ********************************************************** -->
<!-- Specify a variable to expose the admins port.              -->
<!-- ********************************************************** -->
<stringVariable name="coexistencePanelBean.secureAdminConsolePort" minimumLength="1"
maximumLength="5">
        <defaultData>9043</defaultData>
        <labelText translatedKey="adminsPortLabel" />
        <helpText translatedKey="adminsPortHelp" />
        <propertiesAssociations>
                <propertiesAssociation keyword="coexistencePanelBean.secureAdminConsolePort" />
        </propertiesAssociations>
        <ismpFileAssociations>
                <ismpFileAssociation responseFileName="baseresponsefile.txt"
propertyKeyType="wizard" propertyKey="coexistencePanelBean.secureAdminConsolePort" />
        </ismpFileAssociations>
        <!-- ******************************************************** -->
        <!-- Specify validation rules for the port.                   -->
        <!-- ******************************************************** -->
        <inputValidation>
                <valid>
                        <ranges>
                                <range>1 to 65535</range>
                        </ranges>
                </valid>
        </inputValidation>
</stringVariable>

<!-- ********************************************************** -->
<!-- Specify a variable to allow IHS to install             -->
<!-- ********************************************************** -->
<booleanVariable name="ihsFeatureBean.active" minimumLength="1" maximumLength="5">
        <defaultData>true</defaultData>
        <labelText translatedKey="ihsLabel" />
        <helpText translatedKey="ihsHelp" />
        <propertiesAssociations>
                <propertiesAssociation keyword="ihsFeatureBean.active" />
        </propertiesAssociations>
        <ismpFileAssociations>
                <ismpFileAssociation responseFileName="baseresponsefile.txt"
propertyKeyType="product" propertyKey="ihsFeatureBean.active" />
        </ismpFileAssociations>
</booleanVariable>

<!-- ********************************************************** -->
<!-- Specify a variable to expose the IHS install directory.        -->
<!-- ********************************************************** -->
<stringVariable name="ihsFeatureBean.installLocation" minimumLength="4">
        <defaultData>/opt/IBMHTTPServer</defaultData>
        <labelText translatedKey="IHSdirLabel" />
        <helpText translatedKey="IHSdirHelp" />
        <propertiesAssociations>
                <propertiesAssociation keyword="ihsFeatureBean.installLocation" />
        </propertiesAssociations>
        <ismpFileAssociations>
                <ismpFileAssociation responseFileName="baseresponsefile.txt"
propertyKeyType="product" propertyKey="ihsFeatureBean.installLocation" />
        </ismpFileAssociations>
        <!-- ******************************************************** -->
        <!-- Specify validation rules for the install directory.      -->
        <!-- ******************************************************** -->
        <inputValidation>
                <valid>
                        <prefixes>
                                <prefix ignoreCase="true">/</prefix>
                        </prefixes>
                        <characters ignoreCase="true">/-
_.abcdefghijklmnopqrstuvwxyz0123456789</characters>
                </valid>
        </inputValidation>
</stringVariable>

<!-- ********************************************************* -->
<!-- Specify a variable to expose the IHS httpd.conf directory.       -->
<!-- ********************************************************* -->
<stringVariable name="defaultIHSConfigFileLocationBean.value" minimumLength="4">
        <defaultData>/opt/IBMHTTPServer/conf/httpd.conf</defaultData>
        <labelText translatedKey="IHSconfLabel" />
        <helpText translatedKey="IHSconfHelp" />
        <propertiesAssociations>
                <propertiesAssociation keyword="defaultIHSConfigFileLocationBean.value" />
        </propertiesAssociations>
        <ismpFileAssociations>
                <ismpFileAssociation responseFileName="baseresponsefile.txt"
propertyKeyType="wizard" propertyKey="defaultIHSConfigFileLocationBean.value" />
        </ismpFileAssociations>
        <!-- ******************************************************* -->
        <!-- Specify validation rules for the install directory.     -->
        <!-- ******************************************************* -->
        <inputValidation>
                <valid>
                        <prefixes>
                                <prefix ignoreCase="true">/</prefix>
                        </prefixes>
                        <characters ignoreCase="true">/-
_.abcdefghijklmnopqrstuvwxyz0123456789</characters>
                </valid>
        </inputValidation>
</stringVariable>

<!-- ********************************************************* -->
<!-- Specify a variable to expose the ihs port.               -->
<!-- ********************************************************* -->
<stringVariable name="coexistencePanelBean.ihsPort" minimumLength="1" maximumLength="5">
        <defaultData>80</defaultData>
        <labelText translatedKey="ihsPortLabel" />
        <helpText translatedKey="ihsPortHelp" />
        <propertiesAssociations>
                <propertiesAssociation keyword="coexistencePanelBean.ihsPort" />
        </propertiesAssociations>
        <ismpFileAssociations>
                <ismpFileAssociation responseFileName="baseresponsefile.txt"
propertyKeyType="wizard" propertyKey="coexistencePanelBean.ihsPort" />
        </ismpFileAssociations>
        <!-- ******************************************************* -->
        <!-- Specify validation rules for the port.                  -->
        <!-- ******************************************************* -->
        <inputValidation>
                <valid>
                        <ranges>
                                <range>1 to 65535</range>
                        </ranges>
                </valid>
        </inputValidation>
</stringVariable>

<!-- ********************************************************* -->
<!-- Specify a variable to allow samples to install            -->
<!-- ********************************************************* -->
<booleanVariable name="samplesBean.active" minimumLength="1" maximumLength="5">
        <defaultData>false</defaultData>
        <labelText translatedKey="samplesLabel" />
        <helpText translatedKey="samplesHelp" />
        <propertiesAssociations>
                <propertiesAssociation keyword="samplesBean.active" />
        </propertiesAssociations>
        <ismpFileAssociations>
                <ismpFileAssociation responseFileName="baseresponsefile.txt"
propertyKeyType="product" propertyKey="samplesBean.active" />
        </ismpFileAssociations>
</booleanVariable>
```

Figure 6 Variables for applicationdef.axml

   g. Verify that the application id is successfully changed from IRU_WASExpress5_1Lnx to IRU_WAS5_1Lnx.
   h. Save the applicationdef.axml file
3. Modify the contents of the applicationdef_english.xml file to match the listing in Appendix F.
4. Rename the packages
   a. Expand the src/IRU_WASExpress5_1Lnx/userPrograms folder, right-click the com.ibm.iru.iru_wasexpress5_1lnx package, and select **Refactor, rename**. Enter `com.ibm.iru.iru_was5_1lnx` for the new package name and click **OK**.
   b. Right-click each Java program in the folder and rename them using the process from the preceding step. Change `WASExpress5_1LnxMain.java` to `WAS5_1LnxMain.java` and `WASExpress5_1LnxVC.java` to `WAS5_1LnxVC.java`.
   c. Click the Navigator tab, navigate to the src/IRU_WASExpress5_1Lnx directory, right-click it and rename it to `src/IRU_WAS5_1Lnx`.
   d. Right-click the IRU_WASExpress5_1Lnx folder containing the `IRU_UpdateAdminConsolePorts5_1.sh` and `IRU_UpdateExpressPorts5_1.sh` files and delete the folder.
   e. Click the Package Explorer tab.
5. Modify the project build path
   a. Right-click the IRU_WAS5_1Lnx project, select **Properties >Java Build Path**. The Properties Editor opens. Click the **Source** tab. Click the folder with a caution sign next to it and click **Edit** to edit the path.



Properties for IRU_WAS5_1Lnx

- Info
- External Tools Builde
- Java Build Path
- Java Compiler
- Javadoc Location
- Java Task Tags
- Project References

Build path entry is missing: /IRU_WAS5_1Lnx/src/IRU_WASExpress5_1Lnx/userPrograms/

Source | Projects | Libraries | Order and Export

Source folders on build path:

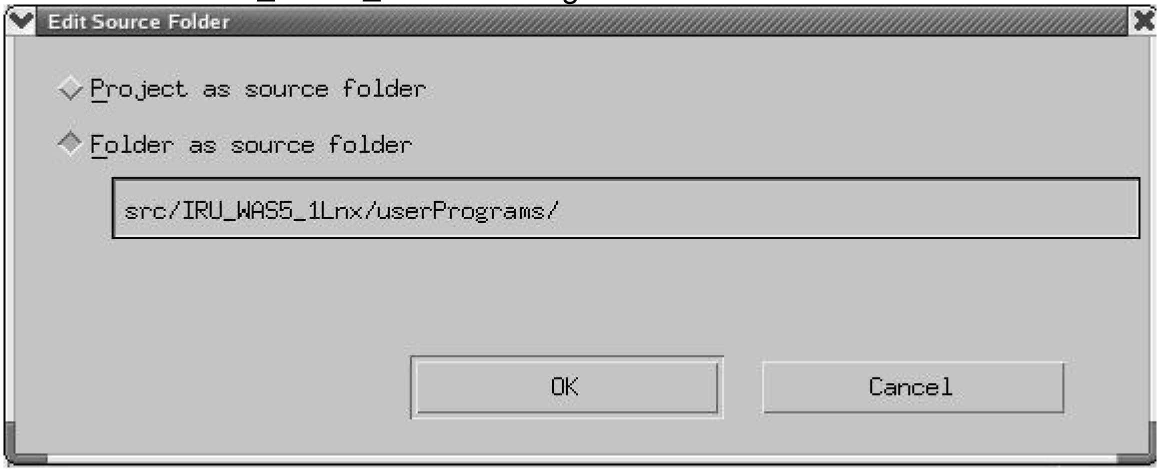
```
IRU_WAS5_1Lnx/src
    Exclusion filter: IRU_WASExpress5_1Lnx/userPrograms/
IRU_WAS5_1Lnx/src/IRU_WASExpress5_1Lnx/userPrograms/ (will be cre
```

□ Allow output folders for source folders.

Default output folder: IRU_WAS5_1Lnx/bin

Figure 7 Edit the properties for the project

   b. Delete the word “Express” from the path, so that the source folder is now named src/IRU_WAS5_1Lnx/userPrograms. Click **OK**.



○ Project as source folder

● Folder as source folder

src/IRU_WAS5_1Lnx/userPrograms/

Figure 8 Edit the source folder location

   c. Highlight the exclusion filter row for `IRU_WAS5_1Lnx/src` and click **Edit**. Click **Edit** again browse to and select the userPrograms folder and click **OK** on the next three dialog boxes so that the name of the IRU_WASExpress5_1Lnx/userPrograms folder changes to IRU_WAS5_1Lnx/userPrograms. Click **OK**.

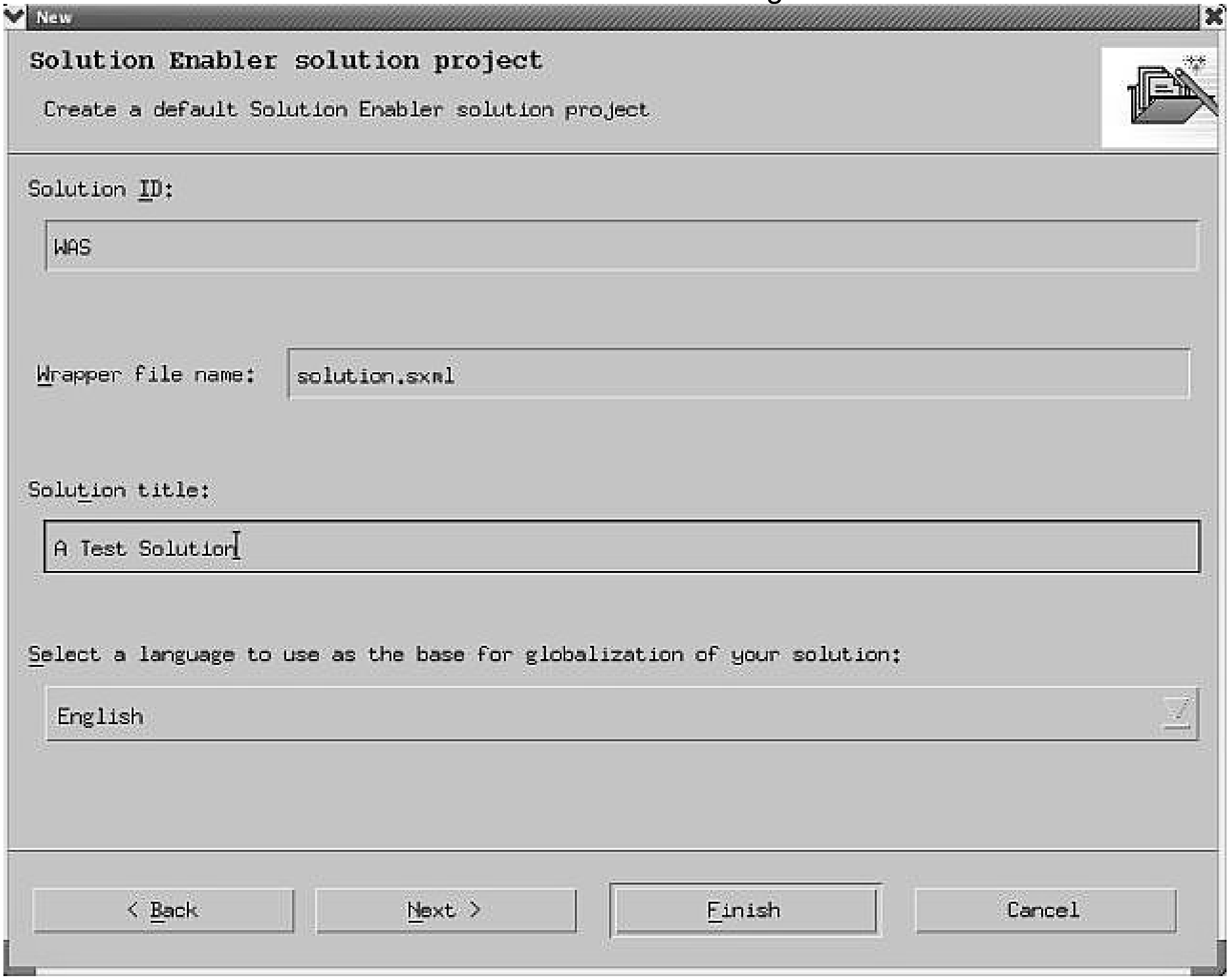
6. Modify the entry and main programs
   a. In the Solution Developer perspective, Package Explorer panel, expand the src/IRU_AS5_1Lnx/userPrograms folder and then expand the com.ibm.iru.iru_was5_1lnx package.
   b. Replace the contents of WAS5_1LnxVC.java with the listing in Appendix B.
   c. Replace the contents of WAS5_1LnxMain.java with the listing in Appendix C.
   d. Add program WASMessagesNLS.java and WASNLSKeys.java from the listing in the appendix.

7. Define the software image root
   a. From the Package Explorer panel of the Solution Developer, expand the src and IRU_WAS5_1Lnx folders.
   b. Right-click applicationdef.axml, and select **Properties > Application Software Image Root**.
   c. Browse to and select the directory containing the WAS5.1BaseLinuxIntel_C53IPML.tar file for WebSphere Application Server. Click **OK**.
8. Change the root element to match the project name in the applicationdef_english.xml file. You can change the name field in the applicationdef_english.xml file from WebSphere Application Server - Express Runtime to WebSphere Application Server. This string is displayed in the Deployer at multiple locations.
9. Build the application
   a. Right-click the application project and select **Solution Enabler Tools > Build Application Definition**.

10. Verify that there are no error messages that appear on the console during the build.

Create a test solution

1. Right-click in the Package Explorer panel of the Solution Developer perspective and select **New > Default Solution Project Template**.
2. Name the project (for example, TestWAS). Click **Next.**
3. Enter the rest of the attributes as shown in the following illustration:



New
Solution Enabler solution project
Create a default Solution Enabler solution project

Solution ID: WAS

Wrapper file name: solution.sxml

Solution title: A Test Solution

Select a language to use as the base for globalization of your solution: English

< Back
Next >
Finish
Cancel

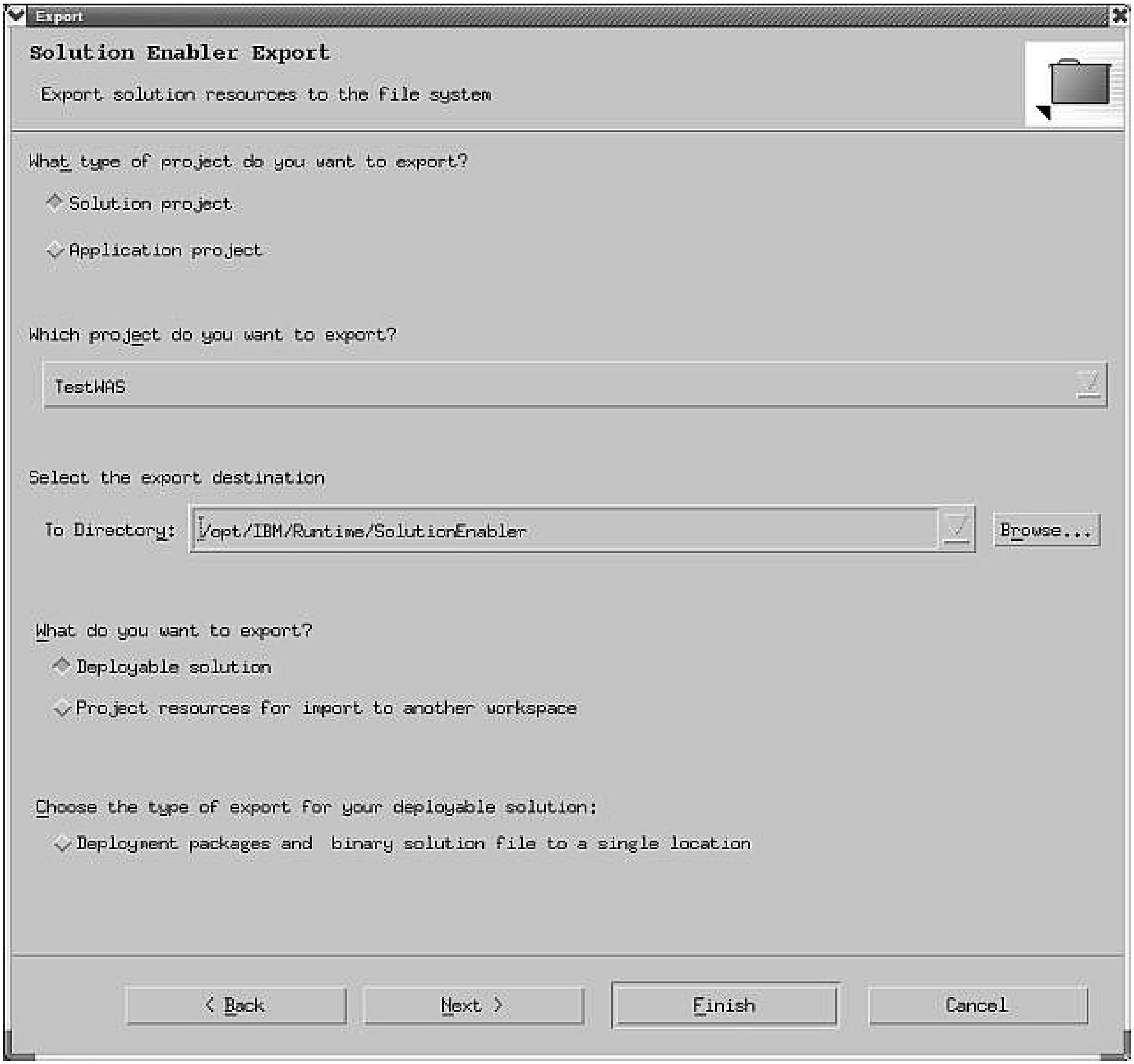
Figure 9 Creating a test solution

4. Click **Next**.
5. In the **Default solution export destination** field, click **Browse** and select the /opt/IBM/Runtime/SolutionEnabler directory. Click **Next**.
6. Select the IRU_WAS5_1Lnx application to include in the solution and click **Finish**.
7. The solution.sxml file opens in the editor. Find the `<description>` tag and replace "`Put your description here`" with "`A Test wrapper for WebSphere Application Server.`"
8. Replace operatingSystem="Windows" with operatingSystem="Linux"

9. Replace fileName="IRU_WAS5_1Lnx_win.ser" with fileName="IRU_WAS5_1Lnx_linux.ser"
10. Build the solution
    a. Right-click the solution project and select **Solution Enabler Tools > Build Solution**.
11. Verify that no error messages appear in the console during the build.

Deploy the solution

1. Export the solution
    a. Right-click the solution project and select **Export**.
    b. Select **Solution Developer** and click **Next**.
    c. Accept the default settings.



Export

Solution Enabler Export

Export solution resources to the file system

What type of project do you want to export?

● Solution project

○ Application project

Which project do you want to export?

TestWAS

Select the export destination

To Directory: /opt/IBM/Runtime/SolutionEnabler

What do you want to export?

● Deployable solution

○ Project resources for import to another workspace

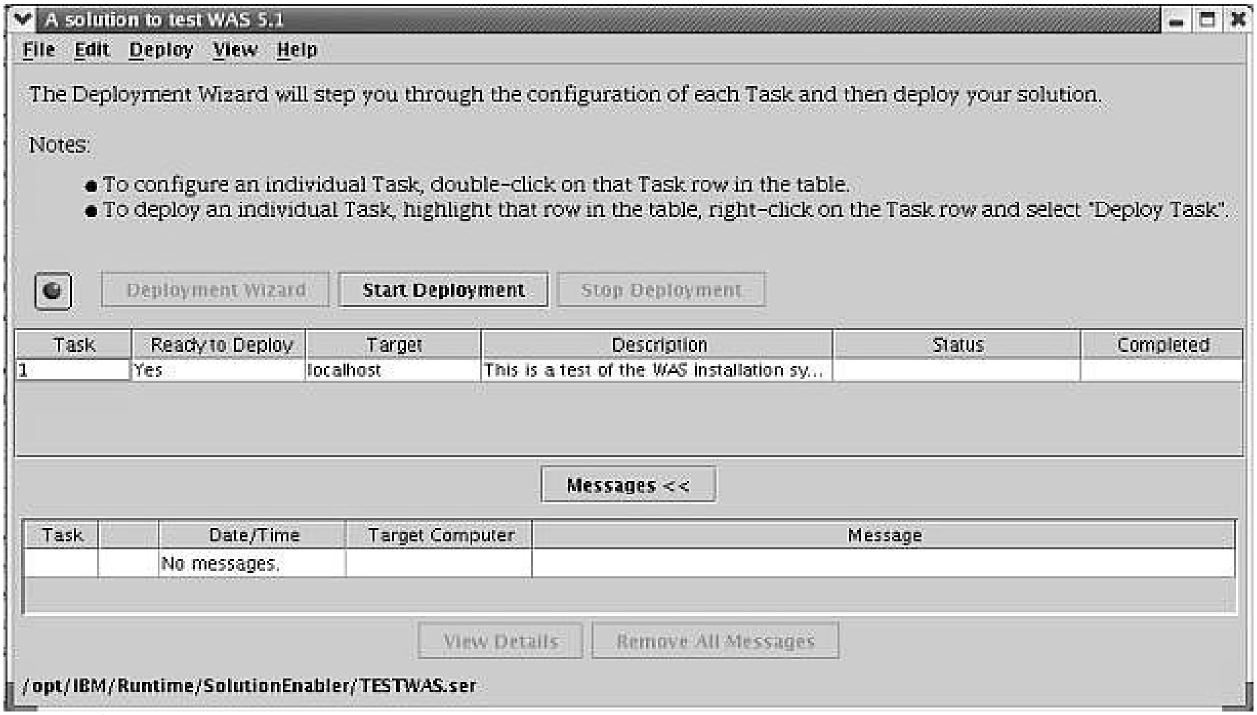
Choose the type of export for your deployable solution:

○ Deployment packages and binary solution file to a single location

Figure 10 Solution export dialog box

d. Click **Finish**.

2. The TestWAS.ser file is located in the /opt/IBM/Runtime/SolutionEnabler directory and the user JAR file is located in the installed location of the IBM Integrated Runtime. You can now deploy the solution.
3. Start the Solution Deployer from the Start menu.
4. Click **File > Open**.
5. Navigate to the TestWAS.ser file and open it.



The Deployment Wizard will step you through the configuration of each Task and then deploy your solution.

Notes:

- To configure an individual Task, double-click on that Task row in the table.
- To deploy an individual Task, highlight that row in the table, right-click on the Task row and select "Deploy Task".

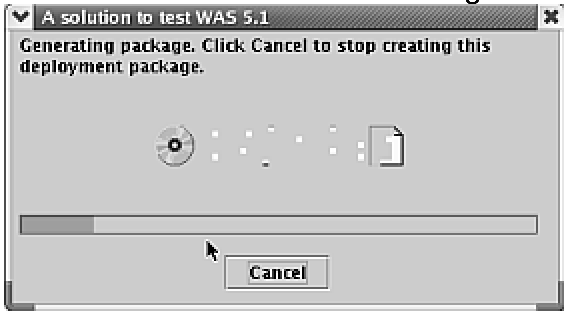
Task	Ready to Deploy	Target	Description	Status	Completed
1	Yes	localhost	This is a test of the WAS installation sy...		

Task		Date/Time	Target Computer	Message
		No messages.		

/opt/IBM/Runtime/SolutionEnabler/TESTWAS.ser

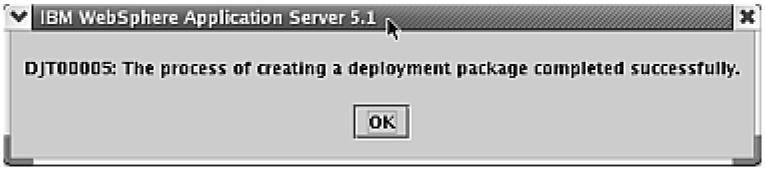
Figure 11 Solution deployer interface

6. Configure the deployment task
   a. Right-click task 1 and select **Configure Task 1**.
   b. In the Task 1 configuration window, click **Target Hostnames**.
   c. Enter `localhost` in the New Target Hostname field and click **Add**.
   d. In the Deployment Parameters panel, click WebSphere Application Server. Enter the DNS hostname.
   e. Click WebSphere Application Server under the deployment packages and click **Create Package**.
   f. Browse to the location of the WAS5.1BaseLinuxIntel_C53IPML.tar file. Click **OK**.
   g. This operation takes some time. The tar file is archived into a JAR file and placed in the installed location of the IBM Integrated Runtime.



Generating package. Click Cancel to stop creating this deployment package.

Figure 12 Deployment package creation in progress



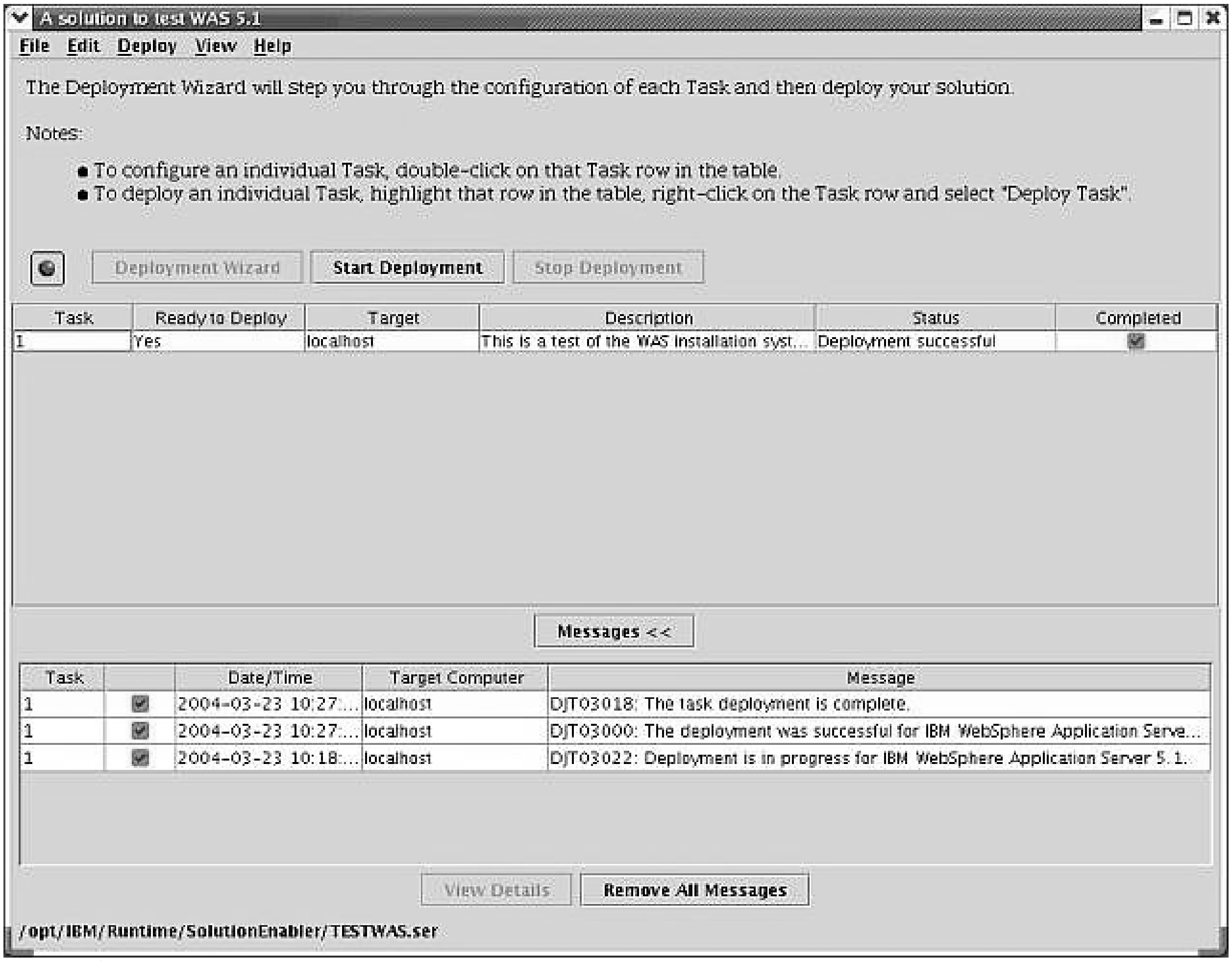
DJT00005: The process of creating a deployment package completed successfully.

Figure 13 Deployment package completion

7. When the process is complete, click **OK** twice. The Start Deployment button becomes active and the “Ready to deploy” entry for the task changes to Yes. Click the **Start Deployment** button.
8. Deployment is complete when the animation of the icon stops and the following message indicating completion of the deployment appears:

```
DJT03000: The deployment was successful for IBM WebSphere Application Server Universal
Database Workgroup Server Edition - Runtime 8.1.
```

The Status field for the task changes to Deployment successful.



The Deployment Wizard will step you through the configuration of each Task and then deploy your solution.

Notes:

- To configure an individual Task, double-click on that Task row in the table.
- To deploy an individual Task, highlight that row in the table, right-click on the Task row and select "Deploy Task".

Task	Ready to Deploy	Target	Description	Status	Completed
1	Yes	localhost	This is a test of the WAS installation syst...	Deployment successful	☑

Task		Date/Time	Target Computer	Message
1	☑	2004-03-23 10:27:...	localhost	DJT03018: The task deployment is complete.
1	☑	2004-03-23 10:27:...	localhost	DJT03000: The deployment was successful for IBM WebSphere Application Serve...
1	☑	2004-03-23 10:18:...	localhost	DJT03022: Deployment is in progress for IBM WebSphere Application Server 5.1...

/opt/IBM/Runtime/SolutionEnabler/TESTWAS.ser

Figure 14 Successful deployment

You now have a complete WebSphere Application Server installation on the system.

Troubleshooting

Here are some activities that you could perform to debug any problems.

- Test the version checker entry program by running it in the Solution Developer. You might have to configure the JRE to get it to work. Do the following steps:
    1. Click **Window > Preferences** to open the preferences editor.
    2. Expand **Java > Installed JRE > Search**.
    3. Browse to the installation location of IBM Integrated Runtime.
    4. Select **DJTJRE** and click **OK**. Wait for the workspace to recompile.
    5. Run the Version Checker as a regular Java program.
- Use the command `/opt/IBM/Runtime/SolutionEnabler/DJT_Deployer -leavefiles` to do the deployment. The log files and intermediate files are left on the machine. You can then examine them for clues for the failure.
- Important files are /opt/IBM/Runtime/SolutionEnabler/DJT_IIAtrace.log and /opt/IBM/Runtime/SolutionEnabler/DJT_SolutionEnabler.log
- If you used the -leavefiles option, additional logs are located in the /opt/IBM/Runtime/SolutionEnabler/logs directory. Notice the edited response file in that directory. It is created from the options entered during the Deployer task configuration. Check the file and ensure that the responses are correct.
- If you make any changes to the application wrapper or any of the XML files, remember to rebuild and export the solution.
- To verify the installation of WebSphere Application Server, start the server and try to access http://localhost:<HTTP transport port usually 9080>/snoop.. This runs the snoop servlet. If you see a page from the snoop servlet, your installation is working.

  If you do not see the servlet, go to the WebSphere Software InfoCenter (http://publib.boulder.ibm.com/infocenter/ws51help/index.jsp) and select **WebSphere Application Server > TroubleShooting > Getting started > Troubleshooting the installation.**

Summary

You have seen how easy it is to package a new application into the delivery structure of the IBM Integrated Runtime product. IBM Integrated Runtime is a middleware solution that can pay for itself through productivity gains alone. It takes care of deployment issues and helps you concentrate on your application. Buy it now by visiting http://www-306.ibm.com/software/smb and selecting your geography.

Appendix

A: Testing for a successful installation

To test for a successful installation of WebSphere Application Server, perform the following steps:

1. Start the application server.
2. Access http://localhost:9090/admin. You see the administrative console. Note that this port number can be changed during installation. You should use the same port that is specified in the response file.
3. Go to http://localhost:9080/WSSamples. You see the predefined samples if you installed them. Note that this document uses the default installation, which does not install the samples.

B: Program listing for version checker

```
package com.ibm.iru.iru_was5_1lnx;
/**
 * @(#)WAS5_1LnxVC.java
 *
 * Licensed Materials - Property of IBM
 *
 * 5724-F71
 *
 * (C) Copyright IBM Corp. 2004
 *
 * US Government Users Restricted Rights - Use, duplication or disclosure
 * restricted by GSA ADP Schedule Contract with IBM Corp.
 *
 * This is sample code made available for use in accordance with terms set
 * forth in the license agreement document for the IBM Integrated Runtime.
 */
import com.ibm.jsdt.support.SupportLinuxBase;

public class WAS5_1LnxVC extends SupportLinuxBase {
        private static final String copyright0 = "Licensed Materials -
Property of IBM";
        private static final String copyright1 = "5724-F71";
        private static final String copyright2 = "(C) Copyright IBM Corp.
2004 All Rights Reserved.";
        private static final String copyright3 = "US Government Users
Restricted Rights - Use, duplication or disclosure restricted by GSA ADP
Schedule Contract with IBM Corp.";

        private static final String PROPERTIES_FILE =
"/properties/version/BASE.product";
        private static final String WAS_NAME = "IBM WebSphere Application
Server";
        private static final String WAS_VERSION = "5.1.0";

        private String ivWASInstallDir = null;

        // codes we have to return to SE
        static final int RETURN_INSTALLED = 1;
        static final int RETURN_NOTINSTALLED = 0;
        static final int RETURN_FAIL = -1;

        /**
         *  WebSphere  does not add an rpm package, therefore
         *  we will look in the directory where this is to be installed for a
product
         *  file which contains the current version.
         *  If the directory for WebSphere  already exists, the function
returns a one and
         *  does a System.out.println of the version number.  If the version
number matches what
         *  Solution Deployer is deploying, Solution Deployer will not
install WebSphere  and indicate
         *  it is already installed.  If a different version of WAS is found,
Solution Deployer will
         *  not install WebSphere  and indicate a failed deployment.
         *
         */

        private int checkVersion() {
                System.err.println("installid" +
getHelper().getProductInstallingId(this));
                setJarFile(getHelper().getProductInstallingId(this));
                setVariableName("wasBean.installLocation");
                setWASInstallDir(getHelper().getIbmNsiPropValue(this));
                System.err.println("installDir" +
getHelper().getIbmNsiPropValue(this));

                int rc = RETURN_NOTINSTALLED;
                // if the target install directory exists, see if this version
is the one installed
                // If the path exists set rc = 1 even if it is a different
version
                // If the path exists and this version is found, println the
version
                // If the path exists and this version is not found, println
anything other than the version
                // it may be an older version or a residual directory left
from a previous install but
                // WebSphere  will not install if the target directory already
exists
                // rc = 1 will indicate the product is already installed but
will flag an error if
                // the versions are different (or the version file is not
found)

                setFileName(getWASInstallDir());
                if (getHelper().fileExists(this)) {

                        String path = getWASInstallDir() + PROPERTIES_FILE;
                        System.err.println("path=" + path);
                        setFileName(path);
                        if (getHelper().fileExists(this)) {

                                rc = RETURN_INSTALLED;
                                int versionRc = StrSearchFile(path, WAS_NAME,
"<version>" + WAS_VERSION + "</version>");
                                System.err.println("Versionrc" + versionRc);
                                if (versionRc == 0) {
                                        System.out.println(WAS_VERSION);
                                } else {
                                        rc = RETURN_NOTINSTALLED;
                                }
                        }
                }
                return rc;
        }
        /**
         * StrSearchFile function calls the readFile method and searches the
content in the specified
         * file for the name and version.
         */
        private int StrSearchFile(String fil, String nam, String ver) {
                int rc = 1;
                try {
                        setFileName(fil);
                        String searchFile = getLinuxHelper().readFile(this);

                        if ((searchFile.indexOf(nam) != -1) &
(searchFile.indexOf(ver) != -1)) {
                                rc = 0;
                        }
                } catch (Exception ex) {
                        System.out.println(ex.toString());
                }
                return rc;
        }
        /**
*****************************************************************************
         *
         * Standard getters and setters
         *
*****************************************************************************
         */
        /**
        * Get the WAS install directory name.
        */
        private String getWASInstallDir() {
                return ivWASInstallDir;
        }
        /**
         * Set the WAS install directory name.
         */
        private void setWASInstallDir(String myDir) {
                ivWASInstallDir = myDir;
        }
        /**********************************
         ******  Main Routine   ***********
         *********************************/
        public static void main(String args[]) {
                WAS5_1LnxVC checker = new WAS5_1LnxVC();
                System.exit(checker.checkVersion());
        }

}
```

Figure 15 WAS5_1LncVC Program Listing

C: Program listing for the installation program

```
package com.ibm.iru.iru_was5_1lnx;
/**
 * @(#)WAS5_1LnxMain.java
 *
 * Licensed Materials - Property of IBM
 *
 * 5724-F71
 *
 * (C) Copyright IBM Corp. 2004
 *
 * US Government Users Restricted Rights - Use, duplication or disclosure
 * restricted by GSA ADP Schedule Contract with IBM Corp.
 *
 * This is sample code made available for use in accordance with terms set
 * forth in the license agreement document for the IBM Integrated Runtime.
 */
import java.io.InputStream;

import com.ibm.jsdt.support.SupportLinuxBase;
import com.ibm.jsdt.support.SupportLinuxHelper;
import com.ibm.iru.message.NLSKeys;

/**
 * This class installs WebSphere .
 */
public class WAS5_1LnxMain extends SupportLinuxBase {
        private static final String copyright0 = "Licensed Materials - Property of IBM";
        private static final String copyright1 = "5724-F71";
        private static final String copyright2 = "(C) Copyright IBM Corp. 2004 All Rights
Reserved.";
        private static final String copyright3 = "US Government Users Restricted Rights - Use,
duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.";

        private static final String LOG_FILE = "WAS5_1LnxMain.log";
        private static final String RUNTIME_MESSAGES = "com.ibm.iru.message.MessagesNLS";
        private static final String WAS_MESSAGES = "com.ibm.iru.iru_was5_1lnx.WASMessagesNLS";

        private static final String TAR_FILE = "WAS5.1BaseLinuxIntel_C53IPML.tar";
        private static final String PROPERTIES_FILE = "properties/version/BASE.product";
        private static final String WAS_NAME = "IBM WebSphere Application Server";
        private static final String WAS_VERSION = "5.1.0";

        // codes returned by websphere install
        static final int WAS_INSTALL_SUCCESSFUL = 1;
        static final int WAS_INSTALL_FAIL = 0;
        static final int TAR_SUCCESSFUL = 0;

        // codes we have to return to SE
        static final int RETURN_INSTALL_SUCCESSFUL = 0;
        static final int RETURN_INSTALL_FAIL = -1;

        // codes from procedure in this program
        static final int RETURN_OK = 0;
        static final int RETURN_FAIL = 1;

        private String ivWASInstallDir = null;
        private SupportLinuxHelper ivHelper = null;
        private String ivUnpackedDir = null;
        private String ivRmiPort = "";
        private String ivSoapPort = "";
        private String ivHttpPort = "";
        private String ivHttpsPort = "";
        private String ivAdminPort = "";
        private String ivAdminsPort = "";

        /**
         * Constructor
         */
        public WAS5_1LnxMain() {
                setLogFileName(LOG_FILE);
                setMainResources(RUNTIME_MESSAGES);
        }
        /**
         * Get the arguments from the properties file and invoke the install.
         * Use the default server name during installation to install the admin console
         * Stop and restart the app server to activate the admin console app
         */
        private int install() {
                int rc = RETURN_OK;
                ivHelper = (SupportLinuxHelper) getHelper();

                // get information from the properties file
                rc = getProperties();
                System.out.println("RC from getProperties" + rc);
                if (rc != RETURN_OK) {
                        return RETURN_INSTALL_FAIL;
                }

                // Untar tar file
                rc = untar();
                System.out.println("RC from untar" + rc);
                if (rc != RETURN_OK) {
                        return RETURN_INSTALL_FAIL;
                }

                // do the actual install
                rc = installWas();
                System.out.println("RC from install" + rc);
                if (rc != RETURN_OK) {
                        return RETURN_INSTALL_FAIL;
                }

                // check that the return codes do not lie
                rc = verifyInstall();
                System.out.println("RC from verify" + rc);
                if (rc != RETURN_OK) {
                        return RETURN_INSTALL_FAIL;
                }

                return RETURN_INSTALL_SUCCESSFUL;
        }

        /**
           * Extract files from WebSphere tar.
           * We have to use the tar because the tarball has symbolic links. Else we could rely
on the jar.
           *
           * Here is an excerpt from info tar
           * GNU tar' returns only a few exit statuses.  I'm really aiming simplicity in that
area, for now.  If you are not
           * using the --compare' (--diff', -d') option, zero means that everything went well,
besides maybe innocuous warnings.
           * Nonzero means that something went wrong. Right now, as of today, "nonzero" is
almost always 2, except for remote
           * operations, where it may be 128.
           */
        private int untar() {
                int rc = 0;
                String completeTarFilePath = ivHelper.getUnpackedDir(this) + TAR_FILE;
                String command = "tar -xf " + completeTarFilePath + " -C " +
ivHelper.getUnpackedDir(this);
                setMessage(getResourceString(NLSKeys.TAR_EXTRACT, completeTarFilePath));
                rc = invokeCommand(command, false);

                if (rc != 0) {
                        setMessage(getResourceString(NLSKeys.TAR_EXTRACT_FAIL,
completeTarFilePath));
                        ivHelper.log(this);
                        setMessage("RC=" + rc);
                        ivHelper.log(this);
                        return RETURN_FAIL;
                } else {
                        return RETURN_OK;
                }
        }

        /**
         * Install the WebSphere
         * Use setupLinux in the unpacked directory
         * Format is setupLinux -options <response file>
         */
        private int installWas() {
                // Run the install and log to temp log file
                //  ./install -options
/opt/IBM/Runtime/SolutionEnabler/workspace/IRU_WAS5_1Lnx/src/IRU_WAS5_1Lnx/baseresponsefile.txt

                String command = ivHelper.getUnpackedDir(this) + "linuxi386/install -options " +
getResponseFileName();

                int rc = invokeCommand(command, false);

                // install WebSphere. NOTE WebSPhere install returns a 1 if it worked OK.
                // have to flip it around
                // BTW this is a pretty bad situation because if installshield fails,
                // e.g. if the destination directory is unwriteable , it returns a 1
                // So it is anybody's guess whether it worked or failed.
                if (rc == WAS_INSTALL_SUCCESSFUL) {
                        // Append the log to the wrapper log
                        setFileName(getWASInstallDir() + "/logs/log.txt");
                        ivHelper.logAppendFile(this);
                        return RETURN_OK;
                } else {
                        setMessage(getResourceString(WASNLSKeys.INSTALL_WAS_FAIL));
                        ivHelper.log(this);
                        return RETURN_FAIL;
                }
        }

        /**
         * Get variables from the properties file
         */
        private int getProperties() {
                int ret = RETURN_OK;

                // get the port values
                setJarFile(ivHelper.getProductInstallingId(this));

                setMainResources(WAS_MESSAGES);

                // like /opt/IBM/WebSphere/AppServer
                setVariableName("wasBean.installLocation");
                ivWASInstallDir = ivHelper.getIbmNsiPropValue(this);
                setMessage(getResourceString(WASNLSKeys.WAS_INSTALL_DIR, ivWASInstallDir));
                ivHelper.log(this);

                setVariableName("coexistencePanelBean.bootstrapPort");
                ivRmiPort = ivHelper.getIbmNsiPropValue(this);
                setMessage(getResourceString(WASNLSKeys.WAS_PORT_ASSIGNMENT, "RMI," +
ivRmiPort));
                ivHelper.log(this);

                setVariableName("coexistencePanelBean.soapConnectorAddress");
                ivSoapPort = ivHelper.getIbmNsiPropValue(this);
                setMessage(getResourceString(WASNLSKeys.WAS_PORT_ASSIGNMENT, "SOAP," +
ivSoapPort));
                ivHelper.log(this);

                setVariableName("coexistencePanelBean.httpTransportPort");
                ivHttpPort = ivHelper.getIbmNsiPropValue(this);
                setMessage(getResourceString(WASNLSKeys.WAS_PORT_ASSIGNMENT, "HTTP," +
ivHttpPort));
                ivHelper.log(this);

                setVariableName("coexistencePanelBean.httpsTransportPort");
                ivHttpsPort = ivHelper.getIbmNsiPropValue(this);
                setMessage(getResourceString(WASNLSKeys.WAS_PORT_ASSIGNMENT, "HTTPS," +
ivHttpsPort));
                ivHelper.log(this);

                setVariableName("coexistencePanelBean.adminConsolePort");
                ivAdminPort = ivHelper.getIbmNsiPropValue(this);
                setMessage(getResourceString(WASNLSKeys.WAS_PORT_ASSIGNMENT, "ADMIN," +
ivAdminPort));
                ivHelper.log(this);

                setVariableName("coexistencePanelBean.secureAdminConsolePort");
                ivAdminsPort = ivHelper.getIbmNsiPropValue(this);
                setMessage(getResourceString(WASNLSKeys.WAS_PORT_ASSIGNMENT, "ADMINS," +
ivAdminsPort));
                ivHelper.log(this);

                return ret;
        }
        /**
          * Invoke the command
          */
        private int invokeCommand(String command, boolean isScript) {
                int rc = 0;
                setMessage(getResourceString(NLSKeys.CMDINVOKED, command));
                ivHelper.log(this);

                try {
                        Process p = null;

                        if (isScript)
                                p = Runtime.getRuntime().exec(new String[] { "/bin/sh", "-c",
command });
                        else
                                p = Runtime.getRuntime().exec(command);

                        // log the stderr if any, helps debugger find problems
                        InputStream is = p.getErrorStream();
                        byte[] b = new byte[4096];
                        while (is.read(b) != -1) {
                                String msg = new String(b);
                                setMessage(msg);
                                ivHelper.log(this);
                        }
                        rc = p.waitFor();
                } catch (Exception e) {
                        rc = 1;
                        setMessage(getResourceString(NLSKeys.CMD_EXCEPTION, e.toString()));
                        ivHelper.log(this);
                }

                return rc;
        }
        /**
         * Verifies the install of WebSphere  was successful
         */
        private int verifyInstall() {
                setFileName(getWASInstallDir());
                if (ivHelper.fileExists(this)) {
                        setGrepString("INSTFIN");
                        setGrepFileName(getWASInstallDir() + SLASH + "logs/log.txt");
                        System.out.println("logfile" + getGrepFileName());

                        if (ivHelper.fileGrep(this)) {
                                // ok install is finished
                                // check for version string
                                setGrepString(WAS_VERSION);
                                setGrepFileName(getWASInstallDir() + SLASH + PROPERTIES_FILE);
                                System.out.println("version file=" + getGrepFileName());
                                if (ivHelper.fileGrep(this)) {
                                        System.out.println("all OK");
                                        return RETURN_OK;
                                }
                        } else {
                                // install did not complete
                        }
                }
                return RETURN_FAIL;
        }
        /**
          ***************************************************************************
          *
          * Standard getters and setters
          *
          ***************************************************************************
          */
        /**
          * Get the WAS install directory name.
          */
        private String getWASInstallDir() {
                return ivWASInstallDir;
        }
        /**
         * Set the WAS install directory name.
         */
        private void setWASInstallDir(String myDir) {
                ivWASInstallDir = myDir;
        }
        /**********************************
         ******  Main Routine   ***********
         ************************************/
        public static void main(String args[]) {
                WAS5_1LnxMain wasMain = new WAS5_1LnxMain();
                if (args.length == 1) {
                        wasMain.setResponseFileName(args[0]);
                }
                int rc = wasMain.install();
                System.out.println("return code from main=" + rc);
                System.exit(rc);
        }
}
```

Figure 16 WAS5_1LnxMain Program listing

D: Program listing for the message catalog

```
package com.ibm.iru.iru_was5_1lnx;
/**
* Licensed Materials - Property of IBM
*
* 5724-F71
*
* (C) Copyright IBM Corp. 2004
*
* US Government Users Restricted Rights - Use, duplication or disclosure
* restricted by GSA ADP Schedule Contract with IBM Corp.
*
* This is sample code made available for use in accordance with terms set
* forth in the license agreement document for the IBM Integrated Runtime.
*/
import java.util.ListResourceBundle;
/**
 * WASMessagesNLS contains the US/English literal objects for WAS
 */
public class WASMessagesNLS extends ListResourceBundle {
        private static final String copyright0 = "Licensed Materials - Property of IBM";
        private static final String copyright1 = "5724-F71";
        private static final String copyright2 = "(C) Copyright IBM Corp. 2004 All Rights
Reserved.";
        private static final String copyright3 = "US Government Users Restricted Rights - Use,
duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.";
        /**
         * Localize the second argument in each pair of brackets.
         * @since JDK1.3
         */
        static final Object[][] messages = {

                // ----------------------------------------------------------------
                // MESSAGES for WAS - range 2000-2099
                // ----------------------------------------------------------------
                { WASNLSKeys.INSTALLING_WAS, "IRU02000: Installation of WebSphere  is in
progress." }, {
                        WASNLSKeys.INSTALL_WAS_SUCCESS, "IRU02001: Installation of WebSphere  was
successful." }, {
                        WASNLSKeys.INSTALL_WAS_FAIL, "IRU02002: Installation of WebSphere
failed." }, {
                        WASNLSKeys.CONFIGURE_WAS_PORTS, "IRU02003: Configuring WebSphere ports."
},
                // {0} represents the port type (RMI, SOAP, etc.) and {1} represents the port
value
                {
                        WASNLSKeys.WAS_PORT_ASSIGNMENT, "IRU02004: Port assignment {0} = {1}." },
{
                        WASNLSKeys.CONFIGURE_WAS_ADMIN_PORTS, "IRU02005: Configuring WebSphere
Administration ports." }, {
                        WASNLSKeys.INSTALLING_WAS_ADMIN_CONSOLE, "IRU02006: Installation of
WebSphere Administration console is in progress." }, {
                        WASNLSKeys.INSTALL_WAS_ADMIN_SUCCESS, "IRU02007: Installation of WebSphere
Administration console was successful." }, {
                        WASNLSKeys.INSTALL_WAS_ADMIN_FAIL, "IRU02008: Installation of WebSphere
Administration console failed." }, {
                        WASNLSKeys.START_WAS, "IRU02009: Starting WebSphere  Application Server."
}, {
                        WASNLSKeys.STOP_WAS, "IRU02010: Stopping WebSphere  Application Server."
},
                // {0} represents the directory where WebSphere  was installed
                {
                        WASNLSKeys.WAS_INSTALL_DIR, "IRU02011: WebSphere  install directory: {0}."
}, {
                        WASNLSKeys.CONFIGURE_WAS_PORTS_FAIL, "IRU02012: Configuring WebSphere
ports failed." }, {
                        WASNLSKeys.CONFIGURE_WAS_ADMIN_PORTS_FAIL, "IRU02013: Configuring
WebSphere Administration ports failed." }, {
                        WASNLSKeys.START_WAS_SUCCESS, "IRU02014: Started WebSphere  Application
Server." }, {
                        WASNLSKeys.STOP_WAS_SUCCESS, "IRU02015: Stopped WebSphere  Application
Server." }, {
                        WASNLSKeys.START_WAS_FAIL, "IRU02016: Failed to start WebSphere
Application Server." }, {
                        WASNLSKeys.STOP_WAS_FAIL, "IRU02017: Failed to stop WebSphere  Application
Server." },
                // {0} represents the name of the application server (eg: "MyAppServer")
                {
                        WASNLSKeys.CREATE_WAS_SUCCESS, "IRU02018: Creation of WebSphere
Application Server {0} was successful." },
                // {0} represents the name of the application server (eg: "MyAppServer")
                {
                        WASNLSKeys.CREATE_WAS_FAIL, "IRU02019: Creation of WebSphere  Application
Server {0} failed." },
                // {0} represents the name of the application server (eg: "MyAppServer")
                {
                        WASNLSKeys.CREATE_WAS_DUPLICATE, "IRU02020: There is already a WebSphere
Application Server named {0} on the target system." }, {
                        WASNLSKeys.SNOOP_WAS_SUCCESS, "IRU02021: Deployment of sample application
snoop was successful." }, {
                        WASNLSKeys.SNOOP_WAS_FAIL, "IRU02022: Deployment of sample application
snoop failed." }, {
                        WASNLSKeys.CREATE_WAS_BAD_PORT, "IRU02023: The range of WebSphere ports
{0} overlaps with the HTTP server port {1}." },

                // End Messages translations.
        };

        /**
         */
        public Object[][] getContents() {
                return getMessages();
        }

        /**
         * Convenience static method to get the messages array.  It is
         * public because JUnitMessageAbstraction needs access to this
         * method.
         */
        public static Object[][] getMessages() {
                return messages;
        }
}
```

Figure 17 WASMessagesNLS.java Program listing

E: Program listing for message catalog key

```
package com.ibm.iru.iru_was5_1lnx;
/**
 *
 * Licensed Materials - Property of IBM
 *
 * 5724-F71
 *
 * (C) Copyright IBM Corp. 2004
 *
 * US Government Users Restricted Rights - Use, duplication or disclosure
 * restricted by GSA ADP Schedule Contract with IBM Corp.
 *
 * This is sample code made available for use in accordance with terms set
 * forth in the license agreement document for the IBM Integrated Runtime.
 */

/**
 * WASNLSKeys contains Static keys for the Resource Bundles
 */

public class WASNLSKeys {
	private static final String copyright0 = "Licensed Materials - Property of IBM";
	private static final String copyright1 = "5724-F71";
	private static final String copyright2 = "(C) Copyright IBM Corp. 2004 All Rights
Reserved.";
	private static final String copyright3 = "US Government Users Restricted Rights - Use,
duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.";

	public static final String INSTALLING_WAS = "02000";
	public static final String INSTALL_WAS_SUCCESS = "02001";
	public static final String INSTALL_WAS_FAIL = "02002";
	public static final String CONFIGURE_WAS_PORTS = "02003";
	public static final String WAS_PORT_ASSIGNMENT = "02004";
	public static final String CONFIGURE_WAS_ADMIN_PORTS = "02005";
	public static final String INSTALLING_WAS_ADMIN_CONSOLE = "02006";
	public static final String INSTALL_WAS_ADMIN_SUCCESS = "02007";
	public static final String INSTALL_WAS_ADMIN_FAIL = "02008";
	public static final String START_WAS = "02009";
	public static final String STOP_WAS = "02010";
	public static final String WAS_INSTALL_DIR = "02011";
	public static final String CONFIGURE_WAS_PORTS_FAIL = "02012";
	public static final String CONFIGURE_WAS_ADMIN_PORTS_FAIL = "02013";
	public static final String START_WAS_SUCCESS = "02014";
	public static final String STOP_WAS_SUCCESS = "02015";
	public static final String START_WAS_FAIL = "02016";
	public static final String STOP_WAS_FAIL = "02017";
	public static final String CREATE_WAS_SUCCESS = "02018";
	public static final String CREATE_WAS_FAIL = "02019";
	public static final String CREATE_WAS_DUPLICATE = "02020";
	public static final String SNOOP_WAS_SUCCESS = "02021";
	public static final String SNOOP_WAS_FAIL = "02022";
	public static final String CREATE_WAS_BAD_PORT = "02023";
	public static final String MAX_CHARS_IN_DEST_DIR = "MaxCharsInDestination";

}
```

Figure 18 WASNLSKeys.java Program isting

F: Program listing for field labels

```
<IRU_WAS5_1Lnx>
        <configureText>Provide the following information and click Next or OK to proceed. Click
Field Help for additional information about the fields</configureText>
        <providerName>IBM</providerName>
        <name>WebSphere Application Server</name>
        <prompt>Specify the fully qualified path name where the WebSphere install image is
located (for example, /downloads/was).</prompt>

        <nodeLabel>WebSphere Node Name</nodeLabel>
        <nodeHelp>Please enter a node name and hostname for this installation. The node name is
used for administration, and must be unique within its group of nodes (cell). You must replace
the "DefaultNode" with the node name that you want the default node to be.</nodeHelp>

        <hostLabel>DNS Hostname</hostLabel>
        <hostHelp>The hostname is the DNS name or IP address for this computer. You must set it
to the fully qualified, resolveable hostname of the target machine.</hostHelp>

        <dirLabel>WebSphere destination directory</dirLabel>
        <dirHelp>The directory where you want to install WebSphere Application Server.</dirHelp>

        <ihsLabel>Install IBM HTTP server ?</ihsLabel>
        <ihsHelp>Do you wish to install the IBM HTTP server(IHS). You would chose yes unless you
already have the IHS installed.</ihsHelp>

        <IHSdirLabel>HTTP server destination directory</IHSdirLabel>
        <IHSdirHelp>The directory where you want to install the IBM HTTP server.</IHSdirHelp>

        <IHSconfLabel>IHS Config File Location</IHSconfLabel>
        <IHSconfHelp>Full path to the HTTP server configuration file httpd.conf.</IHSconfHelp>

        <rmiPortLabel>RMI Connector/Bootstrap Port</rmiPortLabel>
        <rmiPortHelp>Please see the WebSphere documentation for more detail</rmiPortHelp>

        <soapPortLabel>SOAP Connector Port</soapPortLabel>
        <soapPortHelp>Please see the WebSphere documentation for more detail</soapPortHelp>

        <httpPortLabel>HTTP Transport Port</httpPortLabel>
        <httpPortHelp>WAS webcontainer services HTTP requests at this port. Please see the
WebSphere documentation for more detail</httpPortHelp>

        <httpsPortLabel>Secure HTTP Transport Port</httpsPortLabel>
        <httpsPortHelp>WAS webcontainer services HTTPS requests at this portPlease see the
WebSphere documentation for more detail</httpsPortHelp>

        <ihsPortLabel>IHS HTTP Port</ihsPortLabel>
        <ihsPortHelp>IBM HTTP server listens at this port. Strongly reccommend default. Please
see the WebSphere documentation for more detail</ihsPortHelp>

        <adminPortLabel>Admin Console Port</adminPortLabel>
        <adminPortHelp>Please see the WebSphere documentation for more detail</adminPortHelp>

        <adminsPortLabel>Admin Console secure Port</adminsPortLabel>
        <adminsPortHelp>Please see the WebSphere documentation for more detail</adminsPortHelp>

        <samplesLabel>Install WAS samples</samplesLabel>
        <samplesHelp>These are sample applications that install into the WebSphere Application
Server</samplesHelp>
</IRU_WAS5_1Lnx>
```

Figure 19 Modified applicationdef_english.xml

References

[1] Response file installation of WebSphere Application Server overview , Customizing the base options response file in the infocenter at http://publib.boulder.ibm.com/infocenter/ws51help/index.jsp

Acknowledgements

The author would like to take this opportunity to thank the following persons for their contribution to this document:

- **Sahdev Zala** for painstakingly following all the steps laid out in this document and verifying the accuracy of the instructions.

About the author

Har Puri is a developer on the IBM Integrated Runtime Development team in Rochester, Minnesota. He has seven years Java development experience mainly with the WebSphere family of products, including significant time developing resource adapter code for the WebSphere Application Server. Additionally, Har developed sample applications for several Java projects, including IBM SanFrancisco (an enterprise-level Java framework). A technical paper that Har coauthored was presented at eBU 2004, a convention of IBM Software Group Sales Staff and Business Partners at Las Vegas (access available to IBM's Business Partners through **Partnerworld for Software System**). The paper may be viewed or downloaded at the IBM developerWorks web site (http://www.ibm.com/developerworks/websphere/library/techarticles/0401_heins/heins.html). Har is certified on Red Hat Linux (609004545108107) and can be contacted at hpuri@us.ibm.com.

IBM United States of America

Produced in the United States of America

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PAPER “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

Information in this paper as to the availability of products (including portlets) was believed accurate as of the time of publication. IBM cannot guarantee that identified products (including portlets) will continue to be made available by their suppliers.

This information could include technical inaccuracies or typographical errors. Changes may be made periodically to the information herein; these changes may be incorporated in subsequent versions of the paper. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this paper at any time without notice.

Any references in this document to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
4205 South Miami Boulevard
Research Triangle Park, NC 27709 U.S.A.