IBM WebSphere Host Publisher V4.0 Implementation Guide

Introduction

The target audience for this paper is experienced Host Publisher administrator's and technical personnel that will be implementing applications to run with Host Publisher Version 4.0. This paper will focus on important information that you will need to know before upgrading from a previous version of Host Publisher Server to Host Publisher Server Version 4.0.

The information presented here supplements the product documentation in the *Host Publisher Planning and Installation Guide*, *Host Publisher Administrator's and User's Guide*, and *Host Publisher Programmer's Guide and Reference*. This paper also introduces the major new features in Host Publisher 4.0.

Included in this paper are two labs you can use with Host Publisher Version 4.0 and WebSphere Application Developer Version 4.0
- "Lab 1" on page 24 guides you through the process of setting up the Host Publisher Server runtime environment in WebSphere Studio Application Developer (WSAD).
- "Lab 2" on page 26 starts with a Host Access Integration Object that has been enabled for Web Services, and documents the steps needed to build a sample Web Services application. In this lab, you use WSAD to test the application in the WebSphere Test Environment (which is part of WSAD).

When discussing these new features it is assumed that you are already familiar with the terminology and have prior knowledge of Host Publisher. If you are not already familiar with Host Publisher it is suggested you read the Host Publisher 4.0 Administrator's and User's Guide first.

New features overview

IBM WebSphere Host Publisher is a key component of IBM's Host Integration software portfolio. With this new release several key new features have been added. Host Publisher Version 4.0 includes the following new function and features.
- **Compatibility with WebSphere Application Server 4.0**

  Host Publisher Server requires that WebSphere Application Server (WebSphere) 4.0.2 be installed, and supports:
  - WebSphere 4.0.2 Advanced Edition (AE) and Advanced Edition Server (AEs) for Windows NT, Windows 2000, AIX, Solaris, and iSeries
  - WebSphere 4.0.2 Advanced Edition Developer (AEd) for Windows NT and Windows 2000

  As part of the WebSphere 4.0 environment the following enhancements are made in Host Publisher:
  - **J2EE application support:** Applications produced by Host Publisher Studio comply with J2EE, and industry-standard architecture that is intended to reduce the cost and complexity of developing enterprise applications. J2EE applications can be *deployed* rapidly and enhanced easily as the enterprise

responds to competitive pressures. A J2EE application take the form of an *.ear (Enterprise Archive) file* into which all the application's pages, Java objects, and resources are *assembled*.
  - **JavaServer Pages (JSP) 1.1 support:** Host Publisher Studio now produces JSP pages at the JSP 1.1 level. Applications with JSP 1.0 tags will still run, but applications with JSP 0.91 tags (created prior to Host Publisher Version 3.5) need to be migrated. Two migration tools—one on the Studio machine and one on the server—are provided with the product.
  - **Enterprise JavaBeans (EJB) 1.1 support:** Host Publisher now builds EJB-based applications supporting the EJB 1.1 specification level. EJB Access Beans developed with an earlier version of the product must be migrated to the 1.1 level; a migration tool is provided as part of Host Publisher Studio.
- **Web Services:** Web Services, an application integration technology based on open standards and implemented in middleware, provides a way for applications to connect and interact on the Web more easily and efficiently. Host Publisher Integration Objects and EJB Access Beans are enabled to become Web Services.
- **Multi-language support:** On the server, you can use Host Publisher Server Administration and view the Host Publisher documentation in languages other than the server's default language.
- **Serviceability:** The Software Maintenance Utility, a new command-line tool, can help you apply software fixes. For situations that require the involvement of the IBM Support Center, this tool also scans the product and creates a package containing documentation and files for the IBM support team to use in troubleshooting.

Planning and Installation of Host Publisher

Prerequisites for Host Publisher Server

Software requirements:

- WebSphere Application Server (one of the following):
  - Advanced Edition (AE) 4.0.2
  - Advanced Edition Single Server (AEs) 4.0.2
  - Enterprise Edition (EE) 4.1
  - Advanced Edition for Developers (AEd) 4.0.2
- One of the following operating systems, at a level supported by WebSphere Application Server 4.0.2:
  - AIX
  - Solaris
  - Windows NT
  - Windows 2000
- A Web Server. Some examples of a Web Server are:
  - Apache Server
  - IBM HTTP Server (included with WebSphere Application Server)
  - iPlanet Enterprise Server
  - Lotus Domino Application Server
  - Microsoft Internet Information Server

Notes:

1. An FTPD product must be installed and enabled to transfer applications from Host Publisher Studio to the Server.
2. On Windows NT and Windows 2000, with Host Publisher Server and Host Publisher Studio installed on the same machine, you can use the *localhost* option. See the *Host Publisher Administrator's and User's Guide* for more information.

Hardware requirements, in addition to those required by WebSphere:

- 130 MB of disk space, add more for your applications if desired
- 128 MB RAM minimum, 256 MB RAM recommended

Note: These requirements are in addition to requirements made by WebSphere. For example, if WebSphere requires 512 MB RAM, and Host Publisher Server requires 128 MB RAM, you must have a total of 640 MB RAM installed in your machine.

Before installing Host Publisher Server

Before installing Host Publisher Server, you must ensure that you have a working WebSphere 4.0.2 installation. If you are upgrading an earlier installation of Host Publisher Server you must follow the steps below. The recommended sequence of steps for migrating from a previous release of WebSphere and Host Publisher are as follows:

1. Uninstall the previous release of Host Publisher Server. Before uninstalling Host Publisher Server, you must ensure that WebSphere and the Web server are active. After you uninstall, verify that the HostPubServer Application Server was removed from WebSphere. If it was not removed, use the WebSphere Administrative console to remove it. Your Host Publisher applications will remain on the system.

   Note: If you fail to uninstall Host Publisher prior to upgrading WebSphere, you will have a large number of unused files on your system. Refer to the *Readme* for information on removing these files.
2. Perform WebSphere migration to WebSphere 4.0.1 following WebSphere documentation. Refer to the WebSphere V4 InfoCenter instructions and to the WebSphere 4.0.1 Release Notes for migrating to WebSphere V4.0.1. At a high level, this process involves backing up the current WebSphere configuration and user data, stopping and uninstalling the current version of WebSphere, dropping the WebSphere Administrative Database, ensuring that all of WebSphere software prerequisites are installed and operational, and then installing WebSphere 4.0.1 (note that it is not recommended to overlay a 4.0.1 install on top of a directory that the previous version of WebSphere was installed in). Before continuing, ensure that WebSphere is operational by running the WebSphere sample applications.
3. Note that if you are using DB2 for WebSphere's Administrative Database, WebSphere 4.0.2 requires DB2 UDB Version 7.2 FixPak 5 (or later) to be installed before WebSphere 4.0.2 can be installed.
4. Install WebSphere Application Server Version 4.0 FixPak 2, also known as Version 4.0.2. Verify that WebSphere 4.0.2 is operational by running the WebSphere sample applications.

Now that WebSphere has been upgraded to the level required by Host Publisher 4.0 you are ready to continue with Host Publisher Server installation.

Before you begin installing Host Publisher, you must know:

- The directory path into which you will install Host Publisher.
- If you want to migrate applications from a previous version of Host Publisher, the directory path to the application directories. Refer to "Application and XML Gateway session migration" for more information.
- The desktop folder name (for Windows NT and Windows 2000).
- The name of the application server configuration file you want Host Publisher installation to update when Host Publisher is installed with WebSphere AEs. By default, this file is named server-cfg.xml.

  Note: In Windows, be sure that the WebSphere\AppServer\lib\ext directory is writeable.

Application and XML Gateway session migration

Applications and XML Legacy Gateway sessions created for previous versions of Host Publisher must be migrated in order to work with Host Publisher 3.5. You can migrate your existing applications and sessions either during installation by selecting to automatically migrate your applications, or after installation by using a migration utility (for Host Publisher Server known as AppMigrator, and for Host Publisher Studio, StudioAppMigrator). For detailed information about migration, refer to the *Host Publisher Administrator's and User's Guide*.

Installing Host Publisher Server on AIX, Solaris, and Windows

To install Host Publisher Server on Windows NT, Windows 2000, AIX, and Solaris:

1. If you have WebSphere AE, make sure that WebSphere is running before you begin the install process. If you have WebSphere AEs, this is not necessary. If you have WebSphere AEs running during installation, you will have to stop and restart WebSphere to have configuration changes applied.
2. Insert the CD.
   a. **Windows NT and Windows 2000**: Wait for the Start window. If the autorun screen does not appear, run the installation program from the CD root directory (setupwin.exe).
   b. **AIX and Solaris**: Run the installation program from the CD root directory (setupaix.sh or setupsun.sh).
3. Click **Install Product**.
4. Proceed through the windows, making appropriate choices where necessary.
5. To read the latest product information, click the button for the *Readme* file.
6. Click **Finish** when the final window appears.
7. On Windows NT and Windows 2000, reboot after you complete installation.
8. If you have WebSphere AE, use the WebSphere Administrative Console to start the HostPubServer Application server.

Testing installation of Host Publisher Server

After you complete installation of Host Publisher 3.5, load **http://*myhost*/HPAdmin/main.jsp** in your browser, where ***myhost*** is the hostname or IP address of your server. To start Host Publisher Server Administration, load **http://*myhost*/HPAdmin/showCfg** in your browser. If Host Publisher Server Administration or showCfg does not load, check the Host Publisher Server configuration of WebSphere Application Server, as described below.

With WebSphere AE on Windows platforms:

- Confirm that HostPubServer application server is active. If it is inactive, activate it.
- Confirm that HPDoc, HPAdmin, and xmlGateway appear as installed Enterprise Applications in the WebSphere Application Server topology.
- Verify that the following files were installed to the \lib\ext directory under the WebSphere installation directory. If they are missing, copy the files manually from the Host Publisher Server root installation directory:
  - elf.jar
  - habeansnlv.jar
  - HATS.jar
  - HpRte.jar
  - HPShared.jar
  - HPubCommon.jar
  - HPubService.jar
  - log.jar
  - sslight-ex11-rsa-des.zip
  - xlgwWrap.jar
  - xmlLegacy.jar
  - xmlLegacyPortal.jar

  and configure WebSphere using AEWinDeploy.bat from the Host Publisher installation directory.
- If WebSphere is not configured properly, verify that the WebSphere service is started, then run AEWinDeploy.bat from the Host Publisher installation directory and look at the AEdeploy.log file in the temp directory that has been defined for your user account.

With WebSphere AEs on Windows platforms:

- Confirm that HPDoc, HPAdmin, and xmlGateway are installed Enterprise Applications in the WebSphere Application Server topology for the server configuration file chosen during installation.
- Verify that the following files were installed to the \lib\ext directory under the WebSphere installation directory. If they are missing, copy the files manually from the Host Publisher Server root installation directory:
  - elf.jar
  - habeansnlv.jar
  - HATS.jar
  - HpRte.jar
  - HPShared.jar
  - HPubCommon.jar
  - HPubService.jar
  - log.jar
  - sslight-ex11-rsa-des.zip
  - xlgwWrap.jar
  - xmlLegacy.jar
  - xmlLegacyPortal.jar

  and configure WebSphere using AEsWinDeploy.bat from the Host Publisher installation directory.

- If WebSphere is not configured properly, verify that WebSphere HostPubServer is started, then run AEsWinDeploy.bat from the Host Publisher installation directory, and look at the AESdeploy.log file in the temp directory that has been defined for your user account. Stop and restart WebSphere AEs.

With WebSphere AE on AIX and Solaris platforms:

- Confirm that HostPubServer application server is active. If it is inactive, activate it.
- Confirm that HPDoc.ear, HPAdmin.ear, and xmlLegacyGW.ear appear as installed Enterprise Applications in the WebSphere Application Server topology.
- If WebSphere is not configured properly, run AEDeploy.sh from the Host Publisher installation directory, and look at the AEdeploy.log file in your `/tmp` directory.

With WebSphere AEs on AIX and Solaris platforms:

- Confirm that HPDoc.ear, HPAdmin.ear, and xmlLegacyGW.ear appear as installed Enterprise Applications in the WebSphere Application Server topology for the server configuration file chosen during installation.
- If WebSphere is not configured properly, run AEsDeploy.sh from the Host Publisher installation directory, and look at the AESdeploy.log file in your `/tmp` directory. Stop and restart WebSphere AEs.

Installing Host Publisher Studio

Note: If you are upgrading from an earlier version of Host Publisher Studio, you should uninstall the previous version prior to installing 3.5. Uninstalling Host Publisher Studio will not remove applications created with the previous version of Host Publisher Studio. They can be migrated to run with Host Publisher 3.5. For detailed information about migration, refer to the *Host Publisher Administrator's and User's Guide*.

1. Insert the CD.
2. Wait for the Start window. If the autorun screen does not appear, run the installation program from the CD root directory (setup.exe).
3. Proceed through the installation windows.
4. Click **Finish** when the final window appears.

Uninstalling Host Publisher

To uninstall Host Publisher Studio:

1. From the Control Panel, click **Add/Remove Programs**.
2. Select **IBM Host Publisher Studio**.
3. Click **Change/Remove**.

To uninstall Host Publisher Server:

Note: If you have WebSphere AE, make sure WebSphere is running before you begin the uninstall process. If you have WebSphere AEs, this is not necessary. If you have WebSphere AEs running during the uninstallation process, you will have to stop and restart WebSphere to have configuration changes applied.

Note: During uninstall, applications deployed to HostPubServer are stopped and backed up in the directory

```
HostPub\backup
```

on Windows, or on Unix

```
/usr/HostPublisher.saved
```

This is done so that HostPubServer may be stopped, then removed. During a subsequent Host Publisher installation, all of the applications to HostPubServer are restored, but are not started. You must start your applications if you desire them to run.

On Windows NT or Windows 2000:

1. From the Control Panel, click **Add/Remove Programs**.
2. Select **IBM WebSphere Host Publisher Server**.
3. Click **Change/Remove**.

On AIX do one of the following:

- On the command line, type:

  ```
  installp -u HostPublisher.rte
  ```

or

- Through SMIT, remove the filesets named HostPublisher.

On Solaris:

- On the command line, type: `pkgrm IBMhpsvr`

Using Host Publisher Studio to develop J2EE applications

There are several changes, new for this release, in the way application files are packaged by Application Integrator (formerly referred to as the Studio). This section helps you understand the concepts and the steps involved in building J2EE applications with Host Publisher.

Use Host Publisher Studio, running on a workstation, to build J2EE applications that make specific data from the host or database available to end users. Host Publisher Studio is made up of three components:

- Host Access
- Database Access
- Application Integrator

Applications built in Host Publisher Studio contain Integration Objects. Integration Objects are Java beans that encapsulate interactions with a data source, such as a database or a 3270 application, and return specific data from that source for use as output to Web pages, EJB applications, J2EE application clients, or Java thin application client. Integration Objects connect to the data source, navigate to the desired data, extract the data, and store the data in properties of the Integration Object, which can then be accessed by JavaServer Pages (JSP) pages.

A J2EE application is packaged as an .ear (Enterprise Archive) file. The .ear file includes everything necessary to run the application on Host Publisher Server. The elements that make up the .ear file cannot be shared with other applications on the server. However, Host Publisher Studio allows some elements created for one application to be reused in the creation of other applications. When the application is built, a copy of the reused element is included in the application's .ear file.

In a Host Publisher application, the elements that make up the .ear file include Integration Objects, JSP pages, and support files. Here is a summary of the steps for creating and building a J2EE application in Host Publisher Studio:

1. Create an Integration Object in Host Publisher Studio to access a data source and return desired data. See "Integration Objects" in the *Host Publisher Administrator's and User's Guide* for more information.
2. Build Web pages that use one or more Integration Objects to form a Web application.
3. Transfer the Web application to one or more Host Publisher Servers.
4. Deploy (install) the application using the WebSphere Administrative Console. Refer to the WebSphere documentation for more information.

You can also use Host Publisher Studio to create Integration Objects that are run remotely. The two principal ways to do this are:

- You can create J2EE EJB-based applications. The Web module containing the EJB Access Beans can be deployed separately from the EJB module, which runs Integration Objects located on the Host Publisher system.
- You can create Integration Objects that take advantage of Web Services function provided by WebSphere Studio tools at the 4.0.2 level or above, such as WebSphere Studio Application Developer. With Web Services function, your Integration Objects can access data from a Java program (applet or application) running on a remote machine.

For detailed information about creating J2EE EJB based applications or creating Integration Objects that take advantage of Web Services, refer to the *Host Publisher Administrator's and User's Guide*.

Setting preferences in Host Access

As you become experienced with using Host Access, you can customize it according to your preferences. This section describes the contents of the **Options** menu and describes how to change keyboard settings.

Using the Options menu

This section describes the selections you can make on the **Options** menu in Host Access.

You can make these selections at any time, except when a pop-up window has focus. You can undo them simply by reselecting the appropriate selection. The current setting is saved by Host Publisher Studio and remains in effect every time Host Access is started.

Configure Object Chaining: When you select this option, you can enable and define *Integration Object chaining* for the Integration Objects you create.

Automatically Generate Integration Objects on Save: When you select this option, Host Access creates an Integration Object each time you save your macro. If you do not select this option, you must use **File > Create Integration Object** to create an Integration Object.

The default is to create an Integration Object each time you save your macro.

Prompt on Unrecognized Screens: When you select this option, Host Access prompts you to define the screen whenever you navigate the terminal to an unrecognized screen while recording.

The default is to prompt.

Show Hidden Fields on Terminal: When you select this option, you can see hidden fields, such as passwords, on the terminal screen. This helps you navigate around text areas that contain hidden fields when you extract data.

The default is not to show hidden fields.

Display Warning messages: If you are an experienced user of Host Publisher Studio, you might not want to see the warning messages that are intended for less experienced users. When you use Host Access, you can suppress the display of:

- **Confirmation messages:** Messages that allow you to confirm or cancel a request, for example, ″Are you sure you want to stop recording?″

  To suppress the display of confirmation messages, deselect **Options > Display Warning Messages > Display confirmation messages**.
- **Validation messages:** Messages that warn you about macro conditions that might cause problems when you play a macro, for example, ″The first screen of the Connect macro is not defined. The Integration Object might not run. Do you want to save your changes?″

  To suppress the display of validation messages, deselect **Options > Display Warning Messages > Display macro validation messages**.

The default is to display all messages. We recommend that you use the default unless you are an advanced user of Host Publisher Studio.

Play Another Macro: When you are defining a chained Integration Object that is middle or last in the chain, this option enables you to play the data macros of preceding Integration Objects in the chain. Select **Play Another Macro** after the terminal screen shows a connection and you have stopped all other macros. The terminal screen must match the first screen of the macro you are playing.

Create EJB 1.1 Integration Object Support: If you select **Create EJB 1.1 Integration Object Support**, Host Publisher creates the supporting Java files that enable Integration Objects to be processed in an EJB 1.1 environment. These files are created when you save the Integration Object.

The default is to not create the files for EJB 1.1 support.

EJB Integration Object Properties: You can change the suffixes on the names of some files created for EJB Integration Object support.

The default suffixes are:

- *Properties* for the Properties Object file
- *Helper* for the Helper Object
- *Access1* for the EJB 1.1 Access Bean

Create Web Services Integration Object Support: If you select **Create Web Services Integration Object Support**, Host Publisher creates the supporting Java files that enable Integration Objects to be processed in a Web Services environment. These files are created when you save the Integration Object.

The default is to not create the files for a Web Service.

Web Services Integration Object Properties: You can change the suffixes on the names of some files created for Web Services support.

The default suffixes are:
- *Properties* for the Properties Object file
- *Helper* for the Helper Object

The suffixes are used only for the Web Services Integration Object associated with this Integration Object. You can specify different suffixes for each Web Services Integration Object you create. If you open a different Integration Object, the suffix is reset to the value you assigned previously to that Integration Object. If you create a new Integration Object, the suffix defaults to the value in Studio.ini.

Create Remote Integration Object: If you select **Create Remote Integration Object**, Host Publisher creates the supporting Java files and a sample Java program that retrieves Integration Object data from a remote machine. These files are created when you save the Integration Object.

We recommend that you use Web Services rather than Remote Integration Objects—see ″Accessing a remote machine using Web Services″ in the *Host Publisher Administrator's and User's Guide*.

For information about creating Remote Integration Objects, see ″Accessing a remote machine using Remote Information Objects″ in the *Host Publisher Administrator's and User's Guide*.

The default is to not create the files for a Remote Integration Object.

Remote Integration Object Properties: Use this option to change the prefix on the names of the files created for each Remote Integration Object.

The default prefix is *Remote*.

Changing keyboard settings

When the terminal pane is visible, you can edit keyboard settings to assign keys or key combinations as shortcuts to functions in the terminal pane. If you want to edit keyboard settings, from the menu bar, click **Terminal > Keyboard > Edit Keyboard Settings**. Each keyboard configuration is saved as *filename*.hpk in *install_dir*\Studio\Preferences.

You should not delete any predefined host functions or change the keystrokes associated with a predefined host function. If you delete a predefined host function or change its keystrokes, the original values are restored when you save the keyboard settings. However, you can change or unassign the key assigned to a predefined host function.

For example, to assign a new function to the F2 key in the Edit Keyboard Settings window, you would:
1. Select Host Functions in the **Category** box.
2. Press the **Add** key to add a new key definition.
3. Assign a new function name—for example, *Delete Field*—and specify the appropriate keystrokes. Click **OK**.
4. Highlight **Delete Field** in the list, and assign the F2 key to it. When the warning prompt appears, asking whether you want to reassign F2 to the new *Delete Field* function, click **Yes**.

Note: To delete a keyboard configuration, delete *filename*.hpk for that configuration.

Using the keypad

Click **Terminal > Keyboard> Show Keypad** to display the keypad. The keypad is displayed in the terminal pane and allows you to select function keys such as PF1 through PF24, PA1, PA2, and Attn.

To send a key to the host, click the key in the keypad or use Tab to move focus to the key and then press the Spacebar.

Transferring and deploying a completed application

This section provides step-by-step instructions for transferring a completed application to the Host Publisher Server machine and making it ready for use as a WebSphere application. In the following steps, the <u>underlined</u> text corresponds to the title that shows in the wizard pane as you perform each step.

Transferring the application to a server

From the Application Integrator menu bar, click **File > Transfer to Server** to begin the process of assembling your finished application and transferring it to a server.

<u>Transfer to Server</u>

You will use this wizard to create a J2EE .ear (Enterprise Archive) file. The .ear file will then be transferred to a Host Publisher Server, where you will be able to execute the application.

The first window in the wizard displays a list of the servers that are already defined. This example assumes that you have not yet defined a server.

1. Click **Server Info**.
2. Click **Add**.

<u>Host Publisher Server Definition</u>

The information you specify here is used to transfer the application .ear file to the server where it will execute. A Host Publisher Server can either be local (on the same machine where you created the application, or connected to it by a Local Area Network) or remote (you must use FTP to reach it).

To define a local server:

1. In the **Host Publisher Server TCP/IP host name** field, type *localhost*.
2. In the **Login user ID** field, type *anonymous*.
3. Select the server platform—the operating system on which Host Publisher Server is installed—from the list provided.
4. Modify the text in the **WebSphere installation directory** field. You will need to insert the drive letter (for example, C:\) to the left of the text. You will also need to correct the directory name if WebSphere is installed in a different directory than the one shown.
5. Click **OK**.

To define a remote server:

1. In the **Host Publisher Server TCP/IP host name** field, type the TCP/IP name for the server, for example servername.mycompany.com.
2. Accept the default provided in the **Port number** field, unless your system administrator tells you to use a different value.

3. In the **Login user ID** field, type a user name which has been defined for FTP to the WebSphere installation directory on the remote server.
4. Select the server platform—the operating system on which Host Publisher Server is installed—from the list provided.
5. If the default directory name was not used when WebSphere was installed on the server, you will need to change the WebSphere installation directory to the name that was used.
6. Click **OK**.

Preferences

Click OK to return to the Transfer to Server wizard.

Transfer to Server

1. Select one or more servers from the list, or click **Select All**.
2. Click **Next**.
3. Click **Transfer** to begin the transfer process. For each remote server you have selected, you are prompted to provide a password.

   Status messages for each transfer are displayed in the status window to inform you of problems and of successful file transfers. If you need to stop the file transfer and correct a problem, click **Stop**.
4. When the transfer is complete, click Save if you want to save the status messages to a file.
5. Click **Finish**.

The application is now packaged as an .ear (Enterprise Archive) file and has been sent to a WebSphere application directory where it is available to be deployed.

Deploying the application on WebSphere Application Server

Refer to the WebSphere InfoCenter for detailed information about deploying or installing an enterprise application. For example, if you are using WebSphere Application Server Advanced Edition on Windows NT, bring up the WebSphere Advanced Administrative console.

From the menu bar, click **Console > Wizards > Install Enterprise Application**, and follow the instructions. When you reach the Select Application Server window, select HostPubServer as the application server on which you want to install the modules contained in your application.

Ensure that you regenerate the Web Server plug-in and that the Enterprise Application is started.

Accessing the application from a browser

To access the Web application, load this URL in your browser: http://server_name/application_name/first_page.jsp.

For example, if your application is named `example`, the starting page is named `page1`, and you transferred the application to a server named `abs17`, the URL is http://abs17/example/page1.jsp.

Migrating from previous versions of Host Publisher Studio

This section describes what you need to do if you are migrating from a previous version of Host Publisher Studio to Host Publisher Version 4.0. See "Migrating from previous versions of Host Publisher on the server" on page 16 for information about migrating in Host Publisher Server.

Installing Host Publisher Version 4.0 on the Studio machine

When migrating from previous versions of Host Publisher Studio to Host Publisher Version 4.0, you should install Version 4.0 in the same directory path on the Studio machine if you plan to access your existing applications. It is also strongly recommended that you uninstall the previous version of Host Publisher before you install Version 4.0.

Migrating applications in Host Publisher Studio

If you intend to use applications created with previous versions of Host Publisher, you should consider the following:

- Applications produced by Host Publisher Studio in Version 4.0 follow the J2EE architecture; as a result, each application exists on the server as an .ear (Enterprise Archive) file. The .ear file contains one or more J2EE modules, an application deployment descriptor, and other files referenced by the J2EE module.

  Before they can be run on Version 4.0, *all* Host Publisher applications on the Studio machine must be:
  - Assembled into .ear files using the Application Integrator component of Host Publisher Studio
  - Transferred to the server using the Transfer to Server wizard
- Some application components created with previous versions of Host Publisher must be upgraded before they can run on Version 4.0. For example, applications with JavaServer Pages (JSP) pages and tags at the JSP .91 level must be upgraded to the JSP 1.1 level. (When you open such an application in Application Integrator, the Migrate Application window notifies you that the application needs to be migrated.)
- You might need to change relative path names within an application because, in a J2EE-compliant server environment, only relative path names within the application's directory structure are allowed. For example `IMG="images\mailbox.gif"` is allowed because the `mailbox.gif` image file resides in a subdirectory called `images` within the application's directory. But a relative path of `"..\images\"` needs to be recoded and the `mailbox.gif` file moved to a subdirectory within the application's directory.

Note: If an application containing Database Access Integration Objects will attempt to connect to a remote DB2 database, the JDBC drivers on the server must be at the same FixPak level as the JDBC driver on the Studio machine where the Integration Objects were generated. Therefore, you cannot migrate an application that was built using JDBC 1.0 drivers. You must upgrade the JDBC driver on the Studio machine, regenerate the Integration Objects in Database Access, and reassemble the application in Application Integrator.

Using the migration utility in Host Publisher Studio

You can perform migration in Host Publisher Studio in either of two ways:

- Implicitly, when you use Application Integrator with **Application Migration** selected on the **Options** menu. (This is the default selection.)

- Explicitly, by issuing the *StudioAppMigrator* command from the command line.

Migration performs the following steps for each existing Host Publisher application:

1. It migrates JavaServer Pages (JSP) pages and tags from previous levels to the JSP 1.1 level. See "Details of migrating JSP pages" on page 15 for details about this stage of the migration.

   Note: If your application contains custom JSP tags, you must update those tags before running the migration tool. You can update the tags manually or by running a customized JSP migration utility as described in the *Host Publisher Programmer's Guide and Reference*.
2. It ensures that all JSP pages, HTML files, macro files, and session files generated by Host Publisher Studio are written with UTF-8 encoding. For example, in each JSP file, the value for the *charset* parameter is set to UTF-8.
3. It migrates Enterprise JavaBeans (EJB) Access Beans in the application to the EJB 1.1 level. See "Details of migrating EJB Access Beans" on page 16 for details about this stage of the migration.
4. It migrates Host Publisher error pages to comply with the JSP 1.1 and Java Servlet 2.2 specifications.
5. It moves all application-specific files and subdirectories from the \Studio directory to the \Studio\Applications directory.

The migration tool issues a status message when the migration is complete. You can check the log file for additional informational and error messages. If you perform the migration from within Application Integrator, the log file is named *appname*migration.log (where *appname* is the name of the application).

Migrating an application in Application Integrator

When you open an application in Application Integrator, it is checked by default to see whether migration is needed; if so, you are required to migrate the application.

For best performance we recommend that you migrate all of your applications immediately after you install Version 4.0, and then use Application Integrator with **Application Migration** cleared (unchecked) on the **Options** menu.

Using StudioAppMigrator from the command line

After you install Host Publisher Version 4.0, you can use the StudioAppMigrator command to perform migration on Host Publisher Version 3.5 and Version 2.2.1 applications. The syntax of the command is:

```
StudioAppMigrator -s file_list [-l log_file]
```

The parameters are:

-s *file_list*
: A required parameter specifying the names of one or more Host Publisher applications to be migrated. When specifying multiple applications, delimit the applications using semicolons and enclose the list in double quotes.

  To migrate all of the Host Publisher applications in a directory, specify the path name for the directory.

-l *log_file*
: An optional parameter specifying the fully-qualified name of the log file. The log file contains a detailed record of the migration steps, including any errors which occurred. If you specify an existing file, the log file is appended.

If you do not specify this parameter, the log file is saved in the current directory as StudioAppMigrator.log.

For example:

```
StudioAppMigrator -s "\myApp\myApp.hpa;\myotherapp\myotherapp.hpa"
            -l D:\HPlog\HP.log

StudioAppMigrator -s e:\hostpub\studio
```

Details of migrating JSP pages

Host Publisher Version 4.0 supports 1.0 and 1.1 JSP pages. Because Host Publisher Version 4.0 does not support .91 JSP pages, migration of pages created by versions 2.2.1 and earlier of Host Publisher is required. Host Publisher migration converts many JSP .91 tags and attributes to JSP 1.1 for you. A list of those tags and attributes appears later in this section.

Host Publisher migration does not handle every JSP .91 tag and attribute. It handles all of the .91 tags and attributes that were generated by earlier releases of Host Publisher Studio. If your JSP page contains tags and attributes that were added by some means other than creating the page using Host Publisher Studio, and these tags and attributes are not in the following list, the migration utility does not convert them. In this case you can do either of the following things:

- Create a customized JSP migration utility as described in the *Host Publisher Programmer's Guide and Reference*. You must execute your customized migration utility before you execute StudioAppMigrator.
- Convert the tags manually, using a text editor compliant with UTF-8.

We recommend that you check the JSP files in your application after performing Host Publisher migration to make sure all tags were converted.

Here is a list of JSP .91 tags and how they are converted when you migrate your JSP page to the JSP 1.1 format.

<BEAN>
Replaced with the <jsp:useBean> tag.

```
<jsp:useBean>       id="dBAcc">       type="IntegrationObject.DBAcc"
class="IntegrationObject.DBAcc" scope="request"> </jsp:useBean>
```

<INSERT></INSERT>
Information between <INSERT> and </INSERT> is replaced with in-line Java code. For example:

```
<%= dBAcc.getDB2ADMINEMPLOYEEBIRTHDATE_(_i0) %>
```

<REPEAT></REPEAT>
Information between <REPEAT> and </REPEAT> is replaced with in-line Java code.

<REPEAT> is replaced with:

```
<%
for (int _i0 = 0; _i0 <= 2147483647;_i0++){
try {
%>
```

where *_i0* is the name of the index used in the original REPEAT tag.

</REPEAT> is replaced with:

```
<%
}
catch (java.lang.ArrayIndexOutOfBoundsException _e0)
 {
  break;
 }
catch (Java.lang.NullPointerException _e)
 {
  break;
 }
}
%>
```

<%@ content_type="text/html;charset=ISO-8859–1" %>
Replaced with the following syntax:

```
<%@ page contentType="text/html;charset=UTF–8" />
```

In error pages created with Host Publisher Studio V2.2.1:

- The word **session** is converted to **hp_session**. The out.close() invocation is removed.
- The tag **com_ibm_HostPublisher_emsg** is converted to **com_ibm_HostPub_emsg**.

Details of migrating EJB Access Beans

Host Publisher migration creates EJB 1.1 support code in .class and .java files in the \Studio\IntegrationObjects directory.

Migration generates new EJB 1.1 Access Beans to update EJB 1.0 files. The names of the files depend on the file suffix you have chosen for your EJB Access Bean. For example, if you use the default file suffixes (*Access1* in Host Publisher Version 4.0 and *Access0* in Host Publisher Version 3.5), then a new EJB Access Bean named *IOName*Access1BeanInfo.java is generated, containing updates to *IOName*Access0BeanInfo.java.

Migration also updates .hpa files and JSP pages in the application so they reference the correct EJB 1.1 file names.

The old EJB Access Bean files (.java and .jar files) are saved in the same directory with a file extension of .old.

Migrating from previous versions of Host Publisher on the server

This section describes what you need to do at the server if you are migrating from a previous version of Host Publisher to Host Publisher Version 4.0. Refer to "Migrating from previous versions of Host Publisher Studio" on page 13 for information about migrating in Host Publisher Studio.

Installing Host Publisher Version 4.0 on the server

If you are migrating from a previous version of Host Publisher (Version 3.5 or Version 2.2.1), we recommend that you follow the procedures in the *IBM WebSphere Host Publisher Planning and Installation Guide*. It is especially important that you uninstall the prior version of Host Publisher before you attempt to upgrade WebSphere Application Server.

As you install Host Publisher Server Version 4.0, migrate your existing Host Publisher applications to ensure that they are compatible with WebSphere 4.0. If you plan to modify the applications—for example to take advantage of new functions in Version 4.0—you should migrate them on the Studio machine using the instructions in "Migrating from previous versions of Host Publisher Studio" on page 13.

However, if you plan to use the applications without making any changes to them, migrate and deploy them on the server using the instructions in "Migrating applications on the server".

This section also describes optional migration steps you can follow on the server after installing Host Publisher Version 4.0:
- "Removing HTTP session-affinity code" on page 19.
- "Updating server properties files" on page 20.

After the applications have been migrated, deploy them using WebSphere. They can now be executed in the WebSphere environment.

Migrating applications on the server

All Host Publisher Version 3.5 (and earlier) applications on the server must be migrated. A Server application migrator utility, provided with Host Publisher Version 4.0, converts the applications to J2EE applications and updates them so they are compatible with WebSphere 4.0. After migrating the applications, you must deploy them using WebSphere before they can be run.

If you have not deleted your existing applications from the server, the installation process for Host Publisher Version 4.0 gives you the option to invoke the Host Publisher Server application migrator utility.

However, you can choose to invoke the application migrator utility later, from the command line, using the *AppMigrator* command. The command line migration gives you flexibility in specifying which applications to migrate.

Note: If an application containing Database Access Integration Objects will attempt to connect to a remote DB2 database, the JDBC drivers on the server must be at the same FixPak level as the JDBC driver on the Studio machine where the Integration Objects were generated. Therefore, you cannot migrate an application that was built using JDBC 1.0 drivers. You must upgrade the JDBC driver on the Studio machine, regenerate the Integration Objects in Database Access, and reassemble the application in Application Integrator.

What the application migrator utility does

The Host Publisher Server application migrator utility does the following things for a Host Publisher application that has an application manifest file in the *install_dir*\Server\production\appmanifest directory:
- Converts the application to a J2EE application, and packages it in an .ear file.
- Migrates JSP pages in the application to the JSP 1.1 level. It replaces JSP .91 tags and their attributes with either JSP 1.1 tags and attributes or with Java code. (See "Details of migrating JSP pages" on page 15 for details about this part of the migration.)
- Migrates XML Legacy Gateway sessions to XML Gateway sessions and saves them in the file *install_dir*\Server\hPubPortalData.xml.
- Saves the migrated application .ear file in the *install_dir*\Server\migration\migratedApps directory.
- Produces a log file in the *install_dir*\Server\migration\migratedApps directory.

You can invoke the Host Publisher Server application migrator utility when prompted during installation time, which is recommended, or you can invoke it later using the AppMigrator command.

Command-line invocation of the application migrator utility

After you install Host Publisher Version 4.0, you can use the AppMigrator command to perform migration on Host Publisher Version 3.5 (and earlier) applications. The syntax of the command is:

Windows platforms
AppMigrator -i *source_dir* [-o *hp40_dir* -s *file_list* -l *log_file* -?]

AIX, Solaris, and OS/400
sh AppMigrator.sh -i *source_dir* [-o *hp40_dir* -s *file_list* -l *log_file* -?]

The parameters are:

-i *source_dir*
A required parameter specifying the source Host Publisher (Version 3.5 or Version 2.2.1) directory containing the applications to be migrated. It is assumed that the applications are in the \Server\production\appmanifest subdirectory. For example, specify `C:\HostPub` if the files are in C:\HostPub\Server\production\appmanifest.

-o *output_dir*
An optional parameter specifying the Host Publisher Version 4.0 installation directory under which the migrated application .ear files will be stored. The files will be stored in the \Server\migration\migratedApps subdirectory. For example, specify `C:\HostPub` if you want the files stored in C:\HostPub\Server\migration\migratedApps.

If you do not specify this parameter, the migrated applications are stored in the \source_dir\Server\migration\migratedApps subdirectory.

-s *file_list*
An optional parameter specifying the names of one or more Host Publisher applications. When specifying multiple files, delimit the files using a semicolon and enclose the list in double quotes.

Specify hPubPortalData.xml to migrate XML Gateway sessions.

If you do not specify this parameter, all Host Publisher applications in the source directory (except XML Gateway sessions) are migrated.

-l *log_file*
An optional parameter specifying the fully-qualified name of the log file. The log file contains a detailed record of the migration steps, including (at the end of the file) a summary of all errors and the applications for which they occurred.

If you do not specify this parameter, the log file is saved in the \output_dir\Server\migration\migratedApps subdirectory with the name Migrate.log. If you specify a file name without a directory path, the log file is saved in the same subdirectory with the name you specified.

-? Displays help information.

Examples

Note: The following examples show the Windows command syntax for the Server application migrator utility.

Example 1: To migrate two applications, *fulist* and *phone*, which are stored in C:\HostPub\Server\production\appmanifest, enter the following:

```
AppMigrator -i C:\HostPub -s "fulist.application;phone.application"
```

The two applications are migrated and their .ear files are stored in C:\HostPub\Server\migration\migratedApps. The log file is stored in the same subdirectory as Migrate.log.

Example 2: To migrate all of the existing applications in C:\HostPub\Server\production\appmanifest and store the migrated application .ear files in D:\HP40\Server\migration\migratedApps, enter the following:

```
AppMigrator -i C:\HostPub -o D:\HP40 -l MyMigrate.log
```

The log file is stored in D:\HP40\Server\migration\migratedApps as MyMigrate.log.

Example 3: To migrate all of the existing applications in C:\HostPub\Server\production\appmanifest, store the migrated application .ear files in D:\HP40\Server\migration\migratedApps, and store the log file in a different directory, enter the following:

```
AppMigrator -i C:\HostPub -o D:\HP40 -l D:\HP40\Server\migration\log\MyMigrate.log
```

The log file is stored in D:\HP40\Server\migration\log as MyMigrate.log.

Example 4: To migrate XML Gateway sessions in C:\HostPub\Server\production\appmanifest, enter the following:

```
AppMigrator -i C:\HostPub -s hPubPortalData.xml
```

The log file is stored in C:\HostPub\Server\migration\migratedApps as Migrate.log.

Removing HTTP session-affinity code

To enforce HTTP session affinity in Host Publisher Version 3.5 and earlier releases, it was necessary to add code to your JSP pages. Although you do not have to remove this extra code from your migrated applications in Version 4.0 , you might experience a slight performance improvement if you do so.

To remove the session-affinity configuration code, edit the JSP file (using a text editor compliant with UTF-8) and manually remove the following:

```
//----------------------------------------------------------------------------
// Establish session affinity to insure the execution of chained bean application
// is performed on the same JVM, giving all IO's in chain access to the same
// connection. If this is the target of form page, the session will have
// been previously established and isNew()will return false.
HttpSession hp_session =request.getSession(true);
if (hp_session.isNew()){
  String targetURL =request.getServletPath();
  String queryString =request.getQueryString();
  if ((queryString !=null)&&(queryString.length()>0)){
     targetURL =targetURL +"?"+queryString;
  }
  response.sendRedirect(response.encodeRedirectURL(targetURL));
  return;
}
//----------------------------------------------------------------------------
```

Updating server properties files

When you install Host Publisher Version 4.0, new server properties files are created, and they contain the default values shown in Appendix B of the *Host Publisher Administrator's and User's Guide*. The server properties files are named server.properties and ras_*xxx*.properties (where *xxx* is the name of a particular application server).

If, while using previous versions of Host Publisher, you modified the default values in the server properties files, you can edit the new files and restore the values you were using.

Using Web Services

Web Services provide a way for applications to connect and interact on the Web more easily and efficiently. Web Services are self-contained, modular applications that can be described, published, located, and invoked over the Web. Platform-neutral and based on open standards, Web Services can be combined with each other in different ways to create business processes that enable you to interact with customers, employees, and suppliers.

Typically, Web Services use Internet protocols such as HTTP, use XML message formats, and are plugged into Web Service registries where other developers can combine and deploy them. Think of them as strategic building blocks for automated business processes that can be deployed across your enterprise and shared with other enterprises.

Support for Web Services has been implemented in a number of IBM software products, including WebSphere Application Server and WebSphere Studio tools (such as WebSphere Studio Application Developer) at the 4.0.2 level or above.

You can use the Host Access and Database Access components of Host Publisher Studio to create supporting files that enable Host Publisher Integration Objects and EJB Access Beans to be deployed as Web Services. Application Integrator assembles these Java objects (and optionally JSP pages that reference the Java objects) into a J2EE .ear file. You can then import the .ear file into a new or existing WebSphere Studio project, where you can create and deploy Web Services.

Creating and deploying a Web Service

To create an Integration Object in Host Publisher and then enable it to become a Web Service, you perform the following steps:

1. Using Host Publisher Studio, create one or more Integration Objects or EJB Access Beans with **Options > Create Web Services Integration Object Support** checked.
2. Import the Web Services Integration Objects or EJB Access Beans into Application Integrator. Create an application that consists of Web Services Integration Objects, EJB Access Beans, or both, and then generate an .ear file for the application by clicking **File > Create J2EE Archives**.
3. Import the .ear file into a J2EE-enabled WebSphere Studio tool, such as WebSphere Studio Application Developer.
4. Use the WebSphere Studio tool to create a Web Service.
5. Generate a sample application and run it to test your Web Service.
6. Complete development and testing of the application you imported in step 3.
7. Deploy the completed application.

For details on performing steps 1 on page 20 and 2 on page 20, see "Chapter 2. Using Host Publisher Studio to develop J2EE applications" in the *Host Publisher Administrator's and User's Guide*.

For a detailed description of the rest of this process, refer to *Host Publisher Programmer's Guide and Reference*.

Accessing Host Publisher from a remote machine using Web Services

You might want to write a program that does not execute in WebSphere but that requires access to an Integration Object. In earlier versions of Host Publisher, you developed Remote Integration Objects (RIOs) to access Integration Object data from a Java program (applet or application) running on a remote machine. With the current version, however, you can use Web Services to perform this same function.

The primary advantages of using Web Services over RIOs are:

- Web Services are strategic. Communication is based on a current, industry-standard suite of standards, APIs, and implementations such as Web Services technology framework or service oriented architecture (SOA). They are based on self-describing Web Service Description Language (WSDL), which is a descriptive interface and protocol binding language.
- The messaging of Web Services is XML messaging, based on the industry standard Simple Object Access Protocol (SOAP). SOAP is language-independent and interoperable between different programming languages executing on different operating systems.
- Web Services can be published and dynamically located. Universal Description, Discovery, and Integration (UDDI) is a registry mechanism that you can use to perform lookups for Web Services descriptions. After lookup, a client application can dynamically bind directly to Web Services provided by the service provider.

Specifying properties for Web Services Integration Objects

In Host Access and Database Access, the **Options** menu has a **Web Services Integration Object Properties** selection. This option specifies file suffixes that are added to the names of the files (.class files and .java files) associated with the Integration Object. The suffixes help you easily locate the Web Services-specific files that are generated by Host Publisher Studio. Default suffixes are provided, but you can use the **Web Services Integration Objects Properties** option to specify different suffixes for the Web Services Integration Object you are currently working on.

The default suffixes are:

- *Properties* for the Properties Object file
- *Helper* for the Helper Object

To change the default suffixes for all Web Services Integration Objects, edit the Studio.ini file.

Programming using Web Services

When you create Web Services Integration Object support in Host Publisher Studio with Host Access or Database Access, Web Services support files are generated during creation of the Integration Object.

The Web Services support files consist of:

- A properties object (named *IONamePropertiesObjectSuffix*), where *IOName* is the name of the Integration Object and *PropertiesObjectSuffix* is the name specified for the Web Services properties object suffix. The properties object contains all possible input and output properties for the Integration Object.
- A helper object (named *IONameHelperObjectSuffix*), where *IOName* is the name of the Integration Object and *HelperObjectSuffix* is the name specified for the Web Services Helper Object Suffix. This helper object extends the Integration Object. It initializes the input properties using an instance of the properties object, invokes the Integration Object with the specified input properties, and then returns the output properties via a new instance of the properties object. If you plan to create an Integration Object Web Service within a web application, you will use the helper object to create the Web Service using WebSphere Studio tools.

If you want to create Web Services using Host Publisher EJB Access beans, you must create EJB 1.1 Integration Object support, which generates Web Services support files. Use the EJB Access Bean to create the Web Service using WebSphere Studio tools.

Programming with Web Services Integration Objects and EJB Access Beans

Host Publisher Web Services support differs from programming with Integration Objects because only the following method is used to create the Web Service:

IONamePropertiesObjectSuffix **processWSRequest(***IONamePropertiesObjectSuffix***)**
throws BeanException

The *processWSRequest* method is contained in the helper object when **Create Web Services support** is checked on the Host Access or Database Access Options menu. The *IONamePropertiesObjectSuffix* object contains the inputs and outputs of an Integration Object. This object is passed to and from the Web Services *processWSRequest* method. It contains getter and setter methods, but no additional methods.

IOName is the name you gave to the Integration Object, and *PropertiesObjectSuffix* is the name specified for the Web Services Properties Object Suffix.

The processWSRequest method takes the properties object as input, drives the Integration Object with those input properties, and returns the output properties of the Integration Object through a new instance of the properties object. The advantage of using the processWSRequest method is that it provides a single method that sets all inputs, drives the Integration Object, and returns all outputs.

Integration Object Chaining with Web Services: If your application requires chaining, you must retrieve the hPubLinkKey property from the first Integration Object in the chain, and set it for all subsequent Integration Objects in the chain.

EJB Access Bean Chaining with Web Services: If your application requires chaining, you must retrieve both the hPubLinkKey and the hPubAccessHandle properties from the first EJB Access Bean in the chain, and set them for all subsequent EJB Access Beans in the chain.

Creating and Deploying Web Services using WebSphere Studio Application Developer (WSAD)

Import the required Integration Objects and EJB Access Beans into Host Publisher Studio Application Integrator. Choose **Create J2EE Archives** to create the .ear file.

Follow the instructions in "Using Host Publisher Integration Objects" in Chapter 1 of the *Host Publisher Programmer's Guide and Reference* to build and test server-side components in WSAD. If you complete these steps successfully, an application server instance exists where the Host Publisher Server runtime has started successfully. You also have folders that represent the following:

- The application_name.ear file you created with Host Publisher Studio
- The application_name.war file contained in the application_name.ear file
- If your .ear file contained EJB Access Beans, the EJB .jar file contained in the application_name.ear. Ensure that you generated the deployment code for the EJB .jar.

Creating Web Services from an Integration Object:

1. Import the helper object (from *IONameHelperObjectSuffix*.jar in the directory *install_dir*\Studio\IntegrationObjects\WS\IOName) into the WEB-INF\lib directory of your Web application.
2. Update the Java Build path:
   - Add the XERCES classpath variable.
3. Create a Web Service in your web application from the helper object:
   a. Create a Java bean Web Service from the *IONameHelperObjectSuffix*.jar.
   b. Use Request scope.
   c. Make processWSRequest the Web Service Java Bean method. Other methods are shown as choices, and you must deselect them.
   d. Choose to generate a sample to test your Web Service. When the sample is launched, choose the processWSRequest method.
   e. For inputs, set the following input properties in addition to the Integration Objects inputs you created in Host Access or Database Access:
      - HPubStartType=0
      - HPubEndType=0
      - hPubErrorOccurred=0
   f. After you have completed development and test of your application, export it from WSAD and deploy it to your Host Publisher Server.

Creating Web Services from an EJB Access Bean:

1. Update the Java Build path:
   - Add the XERCES classpath variable.
2. Create a Web Service in your web application from the EJB Access Bean:
   a. Create a Java bean Web Service from the *IONameEJBAccessBeanSuffix*.jar.
   b. Use Request scope.
   c. Make processWSRequest the Web Service Java Bean method. Other methods are shown as choices, and you must deselect them.
   d. Choose to generate a sample to test your Web Service. When the sample is launched, choose the processWSRequest method.
   e. For inputs, set the following input properties in addition to the Integration Objects inputs you created in Host Access or Database Access:
      - HPubStartType=0
      - HPubEndType=0
      - hPubErrorOccurred=0
   f. After you have completed development and test of your application, export it from WSAD and deploy it to your Host Publisher Server.

The Host Publisher Portlet

Host Publisher V4.0 adds new WebSphere Portal support. The Host Publisher Portlet enables an end user to run Host Publisher V4.0 applications in a portlet.

You can download the Portlet code, along with instructions for setting up and using the Portlet, from http://www7b.software.ibm.com/webapp/portlets/portletemarketplace.

Any Host Publisher 4.0 application can be viewed using the Portlet, except for applications to which frames have been added.

Client Requirements

The Host Publisher Portlet supports browsers capable of rendering HTML 4.0, such as Microsoft Internet Explorer 5.x and later, or Netscape 4.x and later.

Server Requirements

The Host Publisher Portlet requires two servers:

- The first server requires WebSphere Portal 2.1 and WebSphere Application Server V3.5.4.
- The second server requires Host Publisher V4.0 and WebSphere Application Server V4.0.2.

Deploying and installing the portlet

See the WebSphere Portal documentation for installation instructions. After the Host Publisher Portlet is installed, the user must add an instance of the Portlet to one or more WebSphere Portal pages for each Host Publisher application he or she wants to run. When the user visits these pages, he or she must enter edit mode to identify the application by its URL.

There are no configuration parameters to set for the Host Publisher Portlet.

Lab 1

Setting Up Host Publisher Server Runtime in WebSphere Studio Application Developer

This lab describes how to set up WebSphere Studio Application Developer (WSAD) to work with Host Publisher Integration Objects. You will learn how to import an existing WAR (Web archive) file (ssRTE.WAR) into WSAD and how to use sample code to start and stop the Host Publisher runtime server. This WAR file includes the HPRTEStart and HPRTEStop Servlet classes.

To complete Lab 1 you will need a copy of the file ssRTE.WAR, which is included in ssRTE.exe (a self-extracting zip file). You can download ssRTE.exe from the WebSphere Host Publisher Library Web page where this whitepaper is located.

Host Publisher runtime setup

A. Open WebSphere Studio Application Developer

1. Start > Programs > IBM WebSphere Studio Application Developer > IBM WebSphere Studio Application Developer.
2. Perspective > Open > Web to open a Web perspective.

B. Import the Host Publisher runtime .war file

1. File > Import > Select WAR File.
2. Click Next.
3. Browse for the .war file ssRTE.war and select it.
4. Web Project – supply a project name, notice the context root is filled in for you (example: ssRTE).
5. Enterprise Application Project Name – supply a name, for example: ssRTE_EA. (Project name and Enterprise Application names cannot be the same.)
6. Click Next.
7. Module Dependencies – Click Next.
8. Define Java Build Settings – Click the Libraries tab.

   Now you will add the Host Publisher runtime files to the classpath.
   a. Click Add Variable.
   b. Variable Selection –
      1) Click New.
      2) Supply the name: habeansnlv.
      3) Click File to locate the file: habeansnlv.jar in the C:\Hostpub\Common directory and highlight it.
      4) Click Open. Notice that the path is shown in the window.
      5) Click OK.
      6) Notice that habeansnlv is added under Defined classpath variables.
      7) Continue adding the following Host Publisher runtime files to the classpath. Name them without the file extension. You should have nine files added to the classpath, including the habeansnlv.jar file.
         a) HPRte.jar
         b) HPShared.jar
         c) HPubCommon.jar
         d) HPubService.jar
         e) Log.jar
         f) Sslight-ex11-rsa-des.zip – needs to be added as sslight_ex11_rsa_des for name
         g) XmlLegacyPortal.jar
         h) Elf.jar
9. Click Finish to add the project folder.
10. Highlight the ssRTE folder and expand it. Notice that you have several folders and files. Notice the following:
    - Source
      a. StartHPRTE.java
      b. StopHPRTE.java
    - WebApplication
      a. WEB-INF
         1) Classes
            a) StartHPRTE.class
            b) StopHPRTE.class
11. Double click on StartHPRTE.java.
    a. Notice that the source file is opened.

b. Look for the following line of code:
   ```
   private static String HP_INSTALL_DIR = "c:\\Hostpub";
   ```
c. Verify that this drive on is where Host Publisher is installed. If it is not, change the drive letter and then File > Save > StartHPRTE.java, then Project > Rebuild All.

C. Run on the server

1. Right Click on StartHPRTE.class:
   a. Select Run on Server.
   b. You should see a series of messages in the console window as the WebSphere Test Environment is loaded and running.
   c. Eventually a Web Browser loads with the following message:
      ```
      HP RTE started successfully at: date and time, etc.
      ```
      The browser URL is http://localhost:8080/ssRTE/servlet/StartHPRTE.
   d. Change the browser to StopHPRTE and press the Enter key. The Browser loads with the following message:
      ```
      HP RTE stopped successfully at: date and time, etc.
      ```
      The browser URL is http://localhost:8080/ssRTE/servlet/StopHPRTE.
2. If you can successfully start and stop the Host Publisher Server runtime environment, you can proceed to the next lab.

   Remember that any time you need to execute an application that uses Host Publisher Integration Objects in the WebSphere Test Environment, you will need to start Host Publisher Server runtime before invoking your application.

Lab 2

Programming with Web Services Integration Objects

This lab introduces you to the WebSphere Studio Application Developer (WSAD) tool and how to use it in conjunction with Host Publisher Integration Objects (Java beans) to build a web services sample application and test it in the WebSphere Test Environment which is part of WSAD. You will be importing Host Publisher runtime server files into the WSAD project for testing, so that a Host Publisher runtime server will be running locally in your WSAD environment.

Limitations

Integration Objects that have Indexed properties do not work with the current release of WSAD

We are working to address this with the WSAD development team now. Host Access creates output indexed properties when you indicate that you want to extract data as a table. In WSAD, multiple messages are generated when you attempt to generate the Web Services sample when you have specified table output in Host Access.

For instructions on how to manually construct a sample to test your Host Access Web Services application, refer to the Host Publisher Technical Notes database, and search for Technical Note 20347. (Not currently available - will be available at a future date)

Database Access limitations

When you generate a sample application in WebSphere Studio Application Developer to test a Web Service you have developed using the Database Access application in Host Publisher Studio, the test is likely to fail. This is because indexed properties might not be supported in the current release of WebSphere Studio Application Developer. A message is generated if the test fails.

To test your Database Access Web Services applications, we recommend either writing your own sample application to invoke the Web Service or using the Database wizard in WebSphere Studio to avoid the use of indexed properties.

Host Publisher Studio

A. Create an Integration Object and .ear file

1. Build an Integration Object (such as ″Simple″) in Host Access and test it. Note that for Integration Objects with indexed properties (tables) the samples are not generated correctly. This is a WSAD limitation. Therefore, for this lab, you should create an Integration Object that does simple string extracts, without tables.
2. From the toolbar, select Options -> Create Web Services Integration Object Support. Make sure this is checked
3. Save the Integration Object. Notice that the Web Services support files are generated. The files reside in c:\Hostpub\Studio\IntegrationObjects\WS\Simple\SimpleHelper.jar.
4. Close Host Access.
5. Open Application Integrator and import the Integration Object into a new application.
6. Create a J2EE Archive (.ear) file, for example Simple.ear.

WebSphere Application Developer (WSAD)

A. Import the Host Publisher .ear file into WSAD

1. Import the Host Publisher .ear file into WSAD: File > Import > EAR file.
2. Enter the name of the Host Publisher .ear file in the EAR File field, for example (for an application named ″Simple″) C:\Hostpub\Studio\Applications\Simple\Simple.ear.)
3. Enter an Enterprise Application project name. Use any name except the name of the .ear file (for example, SimpleApp).
4. Click Next until the file is imported.

B. Import Web Services Helper Object (SimpleHelper.jar) into the web project (Simple)

1. Highlight the lib directory in the project – Simple/webApplication/WEB-INF/lib
2. Select File > Import > File system.
3. Click Next.
4. For the directory, select \Hostpub\Studio\IntegrationObjects\WS\Simple (where *Simple* is the name of the Integration Object).
5. In the left pane, click (but don't expand) the name of the directory. In the right pane, check the box next to SimpleHelper.jar.
6. Click Finish.

C. Add Host Publisher dependent .jar files to the build path for the project

1. Right click the newly created project (Simple) and select Properties.
2. Select Java build path.
3. Select the Libraries tab.
4. While still on the Libraries tab, click Add External Jars.
5. Select all of the following from \Hostpub\Common:
   a. habeansnlv.jar
   b. HPRte.jar
   c. HPShared.jar
   d. HPubCommon.jar
   e. HPubService.jar
   f. Log.jar
   g. Sslight-ex11-rsa-des.zip – needs to be added as sslight_ex11_rsa_des for name
   h. XmlLegacyPortal.jar
   i. Elf.jar
6. Click OK. The Java build path is now updated.

D. Create Web Service from a Java Bean (Helper Object)

1. From a Web perspective, select the project (named Simple in this example).
2. Click File > New > Web Service.
   a. The Web Service type should be Java bean Web service.
   b. Leave the Web project the same name as your project.
   c. Click Next.
3. For the Bean name in the Web Services Java Bean Selection window, click Browse classes and select SimpleHelper. (Alternatively, you can type `IntegrationObject.SimpleHelper` into the entry field instead of using Browse.)
   a. Be sure SimpleHelper is the class chosen.
   b. Click OK.
   c. Click Next.
4. Web Service Java Bean Identity:
   a. Change Scope to Request.
   b. Click Next.
5. Web Service Java Bean Methods:
   a. Deselect all methods except processWSRequest().
   b. Click Next.
6. Web Service Binding Proxy generation: accept the defaults and click Next.
7. Web Service Test Client:
   a. Check Launch the test client.
   b. Click Next.
8. Web Service sample generation:
   a. Check Generate a sample.
   b. Check Launch the sample.
   c. Click Next.
9. Web Service Publication:
   a. Leave Launch the UDDI... unchecked.
   b. Click Finish

When you have finished, the sample client launches. The URL is http://*hostname*/*projectname*/sample/*IOname*Helper/TestClient.jsp. For example:

```
http://localhost:8080/simple/sample/SimpleHelper/TestClient.jsp
```

10. If WebSphere Studio Application Developer (WSAD) is set up correctly, you can run the sample application from this environment. However, you must first modify the sample application to prevent it from failing with a Null Pointer Exception.

    To modify the application, find the Result.jsp that WSAD generates to display output from the processWSRequest method. The Results.jsp is located in the Simple\webApplication\sample\SimpleHelper directory. Double-click on the Results.jsp and use the editor to make the changes shown below. Remember to Save the Results.jsp after making the changes.

    Change the following code:

    ```
    <TR>
    <TD WIDTH="5%"></TD>
    <TD COLSPAN="1" ALIGN="LEFT">hpuberrorexception:</TD>
    <TD>
    <%
     java.lang.Exception typehpuberrorexception23 =
     mtemp.getHPubErrorException();
    String temphpuberrorexception23 = typehpuberrorexception23.toString();

     %>
    <%=temphpuberrorexception23%>
    <%
     %>
    </TD>
    ```

    To:

    ```
    <TR>
    <TD WIDTH="5%"></TD>
    <TD COLSPAN="1" ALIGN="LEFT">hpuberrorexception:</TD>
    <TD>
    <%
    java.lang.Exception typehpuberrorexception23 =
    mtemp.getHPubErrorException();
    if (typehpuberrorexception23!= null)
    {
       String temphpuberrorexception23 = typehpuberrorexception23.toString();
            %>
            <%=temphpuberrorexception23%>
            <%
    }
     %>
    </TD>
    ```

11. Now you can run the sample application without errors:
    a. With the Host Publisher Runtime environment started, in the ssRTE application folder, StartHPRTE.class > Run on Server.
    b. In the navigation window, highlight the TestClient.jsp associated with your Web Services sample application. For example :

       ```
       Simple/webApplication/sample/SimpleHelper/TestClient.jsp
       ```

    c. Click Run on Server.
    d. A browser window opens with the URL http://localhost/simple/sample/SimpleHelper/TestClient.jsp.
12. In the Methods panel, click the processWSRequest method.

13. In the Input panel, fill in the input variables required for your Integration Object.
14. Enter these input variables for the following fields:
    a. hpubendtype 0
    b. hpuberroroccurred 0
    c. hpubstarttype 0

    Note: These values are required because of the way WSAD builds the sample Host Publisher Web Services application.
15. For applications that use chained Integration Objects, fill in the value for hPubLinkKey on subsequent Integration Objects from the value of the first (using copy/paste).
16. Make sure the Host Publisher runtime environment is started.
17. Click Invoke.

    If your application runs successfully you will see the output results from your Integration Object running in the Results.jsp in the browser window. Notice that the Integration Object output data appears as XML in the hpubxmldata output. This host data, represented here as XML data, could be utilized by another program for further processing and displayed to the user.

Exporting the Web Services application to the production environment

A. Export the .ear file and import it to a production Host Publisher environment

1. Change the proxy URL to correctly point to your SOAP Server URL:
   a. Select your project (for example, Simple).
   b. Open these folders using the + source/proxy/soap/IntegrationObject
   c. Double-click on the *IOName*HelperProxy.java file and open it (SimpleHelperProxy.java).
   d. On line 16, change the `stringURL` variable to the right value for your server.

      For example, change 8080 to 80. 8080 is the WSAD internal WebSphere port, but most production ones are on 80.
2. Export the .ear file:
   a. Select your project (for example, Simple).
   b. Click File > Export > EAR file.
   c. Select application name (SimpleApp) for resources to export.
   d. In response to ″Where do you want to export resources to″ select \Hostpub\Simple.ear.
3. Install and start \Hostpub\Simple.ear in your WebSphere environment just as you would any other Host Publisher application.