IBM® WebSphere® Host Publisher Administrator's and User's Guide

Version 4.0

GC31-8728-03

IBM® WebSphere® Host Publisher Administrator's and User's Guide

Version 4.0

GC31-8728-03

Note

Before using this information and the product it supports, be sure to read the general information under "Appendix F. Notices" on page 153.

4th Edition (April, 2002)

Contents

Chapter 1. Introducing Host Publisher

This book is designed to help application *developers* use and administer the IBM WebSphere Host Publisher product. With the information in this book, you can use Host Publisher to create Web-to-host applications that interact with data sources—such as host applications and databases—and *publish* those applications on the Web.

This book helps you get started quickly and also helps you understand the underlying concepts involved in using Host Publisher. Additional information resources include the product *Readme*, online help, and the Host Publisher Web site, http://www.ibm.com/software/webservers/hostpublisher. (See "Where can I find information online?" on page 6). These resources include updates, corrections, and educational materials.

This book is available in hard copy, in *HTML* and PDF formats on the installation CD, and in HTML and PDF formats on the product Web site.

Note: Throughout this book, **iSeries**™ refers to both iSeries and AS/400®.

What is Host Publisher?

Web-to-host integration is an integral part of many e-businesses. seventy percent of business-critical data and applications reside on IBM host systems, such as zSeries, iSeries, and RS/6000®. Making this information available to new *users* and using it in new ways across intranets, extranets, and the Internet enables you to reduce costs, improve services, generate new sources of revenue, and establish a competitive advantage.

Host Publisher is a set of tools for providing access to data on a legacy data source (a terminal-oriented host application or database application, for example) on the World Wide Web or a private intranet. These legacy data sources typically involve proprietary access and complex applications. Host Publisher enables you to present end users with the data they need without exposing the way the data is accessed.

You can use Host Publisher to develop applications that, when published on a Web *application server*, enable other people to interact with data on a host. For example, if you have an existing host application that enables users to look up employees' telephone numbers, you can create a Web page that lets a user with a browser enter the name of the person he or she wants, and then displays the telephone number on another page.

You can use *Host Publisher Studio* to create *Integration Objects*, which contain logic to perform host data access and retrieval tasks using 3270, 5250, and virtual terminal (*VT*) data streams and relational databases that provide a *Java Database Connectivity (JDBC)* interface such as IBM DB2 Universal Database®, Oracle™, and Sybase. You can then use these Integration Objects to build more sophisticated Host Publisher applications based on *JavaServer Pages (JSP)* pages, *servlets*, or *Enterprise JavaBeans (EJB)*.

Host Publisher uses IBM WebSphere Application Server to provide a consistent, reliable execution environment for Host Publisher applications; for example,

servlets or EJB-based applications, and HTML/JSPs across platforms. Applications created in Host Publisher Studio can be accessed with all standard Web browsers, with or without Java.

Host Publisher provides the enterprise-class features you expect, including security, load balancing, and hot standby. Host Publisher supports *Secure Sockets Layer (SSL)* encryption and authentication, as well as DES-encrypted passwords, to provide a high level of security.

Host Publisher is divided into two major components:

- Host Publisher Studio runs on a Windows-based workstation. It provides a development environment for creating Web-based applications.
- *Host Publisher Server* runs on an application server. It provides a *runtime* environment for executing *J2EE* applications created with Host Publisher Studio.

To use Host Publisher, you create Web-to-host applications using Host Publisher Studio, *transfer* them to the production server, and provide access to the end user. Applications built with Host Publisher Studio can contain Integration Objects. When used with Host Publisher Server, Integration Objects can do the following:

- Automatically establish a connection with a host
- Navigate to and extract data from an application
- Disconnect from the host and end the connection

Host Publisher provides Integration Object chaining. Integration Object chaining involves constructing a set of Integration Objects that depend on each other to drive a connection through several states. Chaining can increase performance and reduce the administration of creating complex applications. For example, you might use chaining in a typical 3270 application that uses multi-level menus. A corporate phone directory might have several menus to focus down to the point where you can list each individual in a particular department. Suppose you want to display the office address for an individual, return to the department list and select a new name, and display the second person's office address. Chaining enables you to break the task into steps, each of which is represented as a single Integration Object. The end user does not have to navigate back through several menus to reach the department list again.

You can optimize connection establishment for each Integration Object by using *connection pooling*. *Connection pools* are defined in Host Publisher Studio and published to the server. Connection pooling is used to cache ready-to-use connections in the server, and thereby avoid overhead for the connection setup. This improves the Web browser response time for displaying a dynamically generated Web page. A developer-defined number of connections can remain active in the pool for subsequent requests from any user. Connection pools can eliminate the overhead of establishing new connections to the host or database for every Web page request.

Host Publisher provides support for executing Integration Objects remotely. When you create an Integration Object in Host Publisher Studio, you can define it to use the *Web Services* functions provided by WebSphere Studio tools, such as WebSphere Application Developer. Web Services are standards-based, easily extended software components. For a description of how Host Publisher provides support for Web Services, see "Chapter 4. Using Web Services" on page 71.

Host Publisher provides support for executing Integration Objects in *EJB containers* to take advantage of the server-side characteristics provided by the EJB

architecture. The Host Publisher Studio provides an option to create a Host Publisher EJB and its support files when you create an Integration Object. The Host Publisher EJB is a stateful-session EJB capable of running Integration Objects in an EJB environment. You can use Host Publisher Studio to build EJB-based applications.

Host Publisher also provides a Web-based terminal emulator function called XML Gateway. While the key focus of terminal-oriented Integration Objects in Host Publisher is to encapsulate a complex screen navigation sequence and hide that detail from the end user, the purpose of XML Gateway is to give the end user access to each terminal screen of the application and to let him or her control the screen navigation using a browser-based emulator. This function can be used by a person who is familiar with a particular terminal-oriented application but who only has access to a browser with no Java support.

For a detailed description of how a Host Publisher Integration Object executes, see "Appendix A. Technical overview" on page 119.

How does Host Publisher compare with WebSphere Application Server?

These two products are complementary. Host Publisher uses the WebSphere Application Server environment to support J2EE applications that include Integration Objects created by Host Publisher. You can reuse Integration Objects within new servlets or custom EJB-based applications using WebSphere Application Server and your favorite Java *interactive development environment (IDE)*, such as WebSphere Studio Application Developer. For information about using Integration Objects in Java servlets and custom EJB-based applications, refer to the *IBM WebSphere Host Publisher Programmer's Guide and Reference*.

How does Host Publisher compare with Host On-Demand?

Host Publisher is a server-based Web-to-host solution. The client workstation communicates with the application server and the server makes a connection to a host, accesses an application, and presents data back to the client in the form of an HTML page.

Host-On-Demand is a client-based Web-to-host solution. After a Java applet emulator is downloaded from the application server to the client, the client connects through a TN3270 server to access a host application directly.

Host Publisher is designed primarily to build applications for end users who are not familiar with typical host screens or how to navigate through legacy applications and for whom a new, easy-to-use graphical interface is critical. Host Publisher also addresses the needs of those who are familiar with host applications but who do not have Java-enabled browsers and therefore require HTML, or who prefer not to use the green-screen interface. As with Web self-service applications, these users are familiar with their standard HTML browser, and they are accustomed to Web response times. Applications for these users might need to access multiple hosts.

Host On-Demand is IBM's answer for Java-based host access primarily designed to meet the needs of intranet and extranet users. These users are familiar with the original host application screens and can be considered power users who require a full function emulator. User desktop software is typically well controlled and can include a Java-enabled browser. Users typically connect for extended periods of time.

What are the advantages of Host Publisher?

Host Publisher is built on open-industry standards, such as Java and HTML. Host Publisher Studio also creates Web-based applications that are J2EE compliant. Integration Objects are reusable components that can be used in *WebSphere* applications created outside Host Publisher Studio. Likewise, you can use WebSphere Studio tools to add new business logic to the applications Host Publisher creates. The Studio also generates fully customizable HTML output with embedded JavaServer Pages tags. You can use any HTML editor to enhance and customize the HTML or JSP tags to meet your design guidelines and personal preferences.

Host Publisher Server provides enterprise-class performance, scalability, and availability through several key functions, such as chaining, connection pooling, load balancing, hot standby, and cross-platform portability. Because Host Publisher Server runs on AIX®, Windows® 2000, Windows NT®, Sun Solaris, and iSeries, applications created with the common Host Publisher Studio will run unchanged in all environments.

Host Publisher can be used with the load balancing and workload management functions in WebSphere to balance workload across a group of Host Publisher Servers. This provides predictable performance, easy scalability, and hot backup. The ability to move from one operating system platform to another will enable you to move your workload to a higher capacity platform as demands increase.

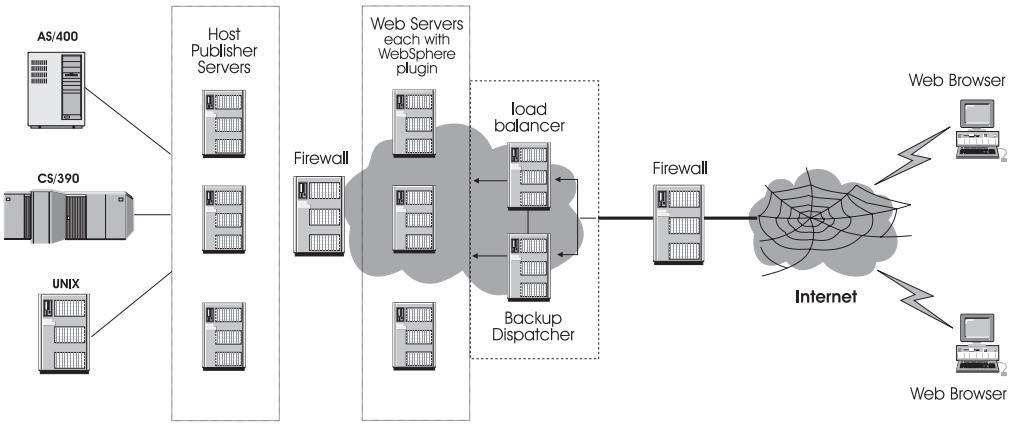


Figure 1. Host Publisher working in a typical network

What is Host Publisher Studio?

Host Publisher Studio is a collection of task-oriented, easy-to-use graphical user interfaces that assist the application developer in managing and creating Web-to-host publishing applications. It uses task-oriented prompts to guide the user through the creation process—including recording host and database interactions, identifying desired data, and labeling that data for retrieval.

Host Publisher Studio automatically generates Java beans called Integration Objects, which encapsulate the interactions and data retrieval logic. You can use Host Publisher Studio to generate fully customizable Web pages for modeling interactions with the Integration Objects and rendering the resulting data. You can enhance the generated Web pages with your favorite Web authoring tool, such as

WebSphere's page designer, to meet corporate guidelines on style and image. When the Web pages are completed, you transfer them to a Host Publisher Server for production access by end users.

Host Publisher Studio runs on the Windows 98, Windows ME, Windows NT, Windows 2000, and Windows XP operating systems.

What is Host Publisher Server?

Host Publisher Server provides the runtime environment for supporting J2EE applications created with Host Publisher Studio. Running on the application server with the IBM WebSphere Application Server product (WebSphere), it includes components such as connection management, license monitoring, runtime administration, express logon, XML Gateway, and log and trace management.

Host Publisher Server is supported on OS/400, AIX, Windows 2000, Windows NT, and Solaris operating environments.

What's new in Host Publisher Version 4.0?

This section describes the major new functions in Host Publisher Version 4.0.

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 Single 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, an 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 takes the form of an *.ear (Enterprise Archive) file* into which all the application's pages, Java objects, and resources are *assembled*.

If you have Host Publisher applications on the server that were developed with an earlier level of the product, they must be migrated to Version 4.0. A migration tool is provided for this purpose.

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.

Where can I find information online?

You can find the following forms of documentation for Host Publisher online.

Manuals

To access online documentation in either Host Publisher Studio or Host Publisher Server, you can use a Web browser to open HTML files and book (PDF) files on your local system.

In Host Publisher Studio

Listed below are the locations for the files for each document, where *install_dir* is the directory in which Host Publisher is installed.

IBM WebSphere Host Publisher Administrator's and User's Guide
install_dir\Common\doc\guide\guide.htm

To open the book (PDF) version, use the file name guide.pdf instead of guide.htm.

IBM WebSphere Host Publisher Planning and Installation Guide
install_dir\Common\doc\install\instgd.htm

To open the book (PDF) version, use the file name instgd.pdf instead of instgd.htm.

IBM WebSphere Host Publisher Programmer's Guide and Reference
install_dir\Common\doc\proggd\proggd.htm

To open the book (PDF) version, use the file name progguid.pdf instead of proggd.htm.

IBM WebSphere Host Publisher Messages Reference
install_dir\Common\doc\msgref\msgref.htm

To open the book (PDF) version, use the file name msgref.pdf instead of msgref.htm.

IBM Host Publisher *Readme*
install_dir\Common\doc\readme.htm

In Host Publisher Server

In Host Publisher Server, the documentation is available in any language you choose, when WebSphere is running on the server. Open a browser to http://ServerName/HPDoc/HPDocServlet to choose the desired language and view the documentation in that language.

Listed below are the file names for each document.

IBM WebSphere Host Publisher Administrator's and User's Guide
guide.htm and guide.pdf in \guide subdirectory

IBM WebSphere Host Publisher Planning and Installation Guide
instgd.htm and instgd.pdf in \install subdirectory

IBM WebSphere Host Publisher Programmer's Guide and Reference
proggd.htm and proggd.pdf in \proggd subdirectory

IBM WebSphere Host Publisher Messages Reference
msgref.htm and msgref.pdf in \msgref subdirectory

IBM Host Publisher *Readme*
readme.htm

Online help

Online help, including the HTML version of this book, is available from the product's graphical user interface. To access online help in Host Publisher Studio, click the Help button on a Host Publisher product window, click Help from the menu bar, or press the F1 key. In Host Publisher Server Administration, click the help icon (a question mark in the upper right corner of the window).

Information on the Web

Find the most up-to-date versions of this document, frequently asked questions (FAQs), white papers, and additional information at the product Web site: http://www.ibm.com/software/webservers/hostpublisher.

Chapter 2. Using Host Publisher Studio to develop J2EE applications

You want to extend applications to the Web. Where do you start? This chapter helps you understand the concepts and the steps involved in building Host Publisher applications.

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*

"Appendix C. Example of developing an application in Host Publisher Studio" on page 129, contains a step-by-step example of the basic steps: creating an Integration Object (using the Host Access component), building an application, and transferring the application to a server. If you are new to Host Publisher, you might want to use the example to introduce you to the product, and then use this chapter to increase your understanding of the tasks involved.

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 clients. 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" on page 10.
2. Build Web pages that use one or more Integration Objects to form a Web application. See "Using Application Integrator to build J2EE applications that use Integration Objects" on page 33.
3. Transfer the Web application to one or more Host Publisher Servers. See "Transferring applications to a Host Publisher Server" on page 45.
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. See "Creating applications that use Enterprise JavaBeans (EJB) technology" on page 42 for details.
- 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. See "Chapter 4. Using Web Services" on page 71 for details.

Integration Objects

Every Integration Object, whether you create it using the *Host Access* component or the Database Access component of Host Publisher Studio, has two characteristics: it interacts with a data source, and it is associated with a connection pool. The following sections cover these characteristics that are common to all Integration Objects.

Later in this chapter, you will find more detailed information about creating Integration Objects using either the Host Access component or the Database Access component:

- "Creating a Host Access Integration Object" on page 12, to create Integration Objects that access data from a 3270, 5250, or VT application
- "Creating a Database Access Integration Object" on page 29, to create Integration Objects that encapsulate a database statement

Interacting with a data source

Host Access Integration Objects interact with the host application by means of *macros*. Macros connect to the host, interact with applications on the host, and disconnect from the host. See "Building an Integration Object using macros" on page 12 for details.

Database Access Integration Objects contain SQL statements to access data in the database. See "Retrieving information from a database" on page 30 for details.

Defining connection pools

Every Integration Object is associated with a *connection pool*. A connection pool is a collection of communication links to back-end data sources, such as 3270 applications or databases. When an Integration Object is run on behalf of a client request and pooling is enabled, the Integration Object obtains an available connection from a pool, uses it for access to the data source, then returns the connection to the pool.

For each Integration Object you create, you can create a new connection pool, share an existing connection pool, or (in Host Access) configure a default connection pool.

After you have defined a connection pool, it can be associated with many different Integration Objects. A copy of the connection pool becomes part of each application that includes one of these Integration Objects, after the application is assembled in Application Integrator.

The information specified in a connection pool

A connection pool specifies several kinds of information, including:

- Data that allows your Host Publisher application to connect to a host. For Host Access Integration Objects, this data includes the host name. For Database Access Integration Objects, it includes the URL of the database.
- Whether connection pooling is enabled. Connection pooling is a method of keeping a connection to a host open for many data transactions, instead of opening and closing a connection with each new request. Connection pooling is designed to reduce the response time between a request from a client browser and the display of the requested information on a Web page.

  See "Enabling and disabling connection pooling" for more information.
- Optionally, a *user list*. User lists contain information about accounts (user IDs) that a Host Publisher application can use to access a host or database. Each entry in a user list includes a user ID, password, and description of the account.
- Connect and disconnect macros (in Host Access only).

See "Creating connection pools for Host Access" on page 14 and "Creating connection pools for Database Access" on page 31 for more component-specific information about connection pools, connection pooling, and user lists.

Enabling and disabling connection pooling

When connections are pooled, the overhead of establishing a connection is absorbed in its first use. Each Integration Object that reuses this connection benefits from the prior establishment of the connection and can run faster.

To illustrate, suppose a database Integration Object requests a connection. If connection pooling is enabled and a connection is available, the connection has already been made and is ready to use. If connection pooling is enabled, but a connection is not available, the connection is created. If connection pooling is disabled, a connection is always created.

When the Integration Object has completed, the connection is returned to the pool. If connection pooling is enabled, the connection remains available and can be used for the next requested connection. If connection pooling is disabled, the connection is terminated.

For Host Access Integration Objects, when a connection is made to the host the associated connect macro runs. When the host connection is ended, the disconnect macro runs. Enabling connection pooling means that the connect macro is run only when necessary to make a connection available. After a connection is available, it can be used again and again without the overhead of running the connect and disconnect macros for each connection request.

User lists

In Database Access, a user list is created by default with the user ID and password you enter to connect to the database. Because multiple users can access the database at the same time with the same user ID and password, user lists for Database Access always contain only one entry.

In Host Access, a user list is not created automatically; you must define it explicitly, for example, by clicking **User List** in the Host Configuration window. If your application will access a host that allows multiple users to logon with the same user ID, the user list might have only one entry. However, for a single-logon host, which allows only one connection per user ID, the number of users who can access the host at the same time using your application is limited to the number of entries in the user list.

See "Defining user lists" on page 26 for more information about user lists in Host Access.

Creating a Host Access Integration Object

To create Integration Objects that collect data from applications on the terminal-oriented host, use the Host Access application. To create the Integration Objects, you navigate to the information you want using a 3270, 5250, or VT connection. Host Publisher records the keystrokes you use and lets you define the host application screens that contain information.

This section provides a general description of the major function of Host Access: using macros to build Integration Objects. Then it provides information to help you with the following tasks:

Building an Integration Object using macros

The Host Access application is used to build Integration Objects that access data from a 3270, 5250, or VT application. To do this, Host Access records *macros* that contain information about the way you connect to the host, how you navigate to the information you want to make available to your end users, and how you disconnect from the host. These macros become part of the Integration Object you create.

Host Publisher uses IBM Host On-Demand's macro facility to provide an interface where you can interact with a host and record the actions you take. The actions are recorded as a macro.

A macro is an *XML* script that defines a set of screens. Each screen includes a description of the screen, the actions to perform for that screen, and the screen or screens that can be presented after the actions are performed.

Note: Macros are not used by Integration Objects created using the Database Access application.

Macros work by following a sequence of host screens that you define. As you navigate through your application using a terminal screen, you define:

- The screens
- The actions to take on each screen (that is, keystrokes)
- Which screens can appear next after the action has completed for a given screen

In Host Access, there are three types of macros:

Connect
Includes the information Host Publisher needs to connect to the host. The connect macro should contain the steps necessary for logging on to a system from as many initial states as possible. It should take into account different paths that might occur because the previous connection failed or was left in an unknown state. Host Publisher Server uses the connect macro to attempt to connect to a system and to recover from a previously-failed connection. See "Recording conditionals" on page 23 for more information on recording alternate paths.

Data Includes information about navigating to, extracting, and organizing the data you want to publish.

Disconnect
Includes information about how and when to disconnect from the host. Typically, this means tearing down the network connection. Disconnect macros prepare the host connection to cleanly disconnect.

In Host Access, the connect and disconnect macros are part of the connection pool associated with the Integration Object. Only the data macro is part of the Integration Object itself.

You can also define loops within the macro and alternate paths (for example, more than one "next screen" for a given screen) to follow. Host Access includes a wizard that guides you through recording macros, but you can use the **Macro** menu and the toolbar to fine-tune them.

When you run your macro on Host Publisher Server by invoking your Integration Object, your macro looks for the screens you defined to appear on the host terminal and will execute the actions you defined for each screen.

Integration Object *chaining* might help improve your application's response time or decrease the amount of macro recording you must do. See "Integration Object chaining" on page 73 for more information.

Experienced users can edit macro scripts after they have been created using Host Access. See "Editing a macro" on page 19 for more information.

Using the Host Access wizard

When you use Host Access to create an Integration Object, it launches a wizard. To start Host Access in Windows, click **Start > Programs > Host Publisher Studio > Host Access**.

The wizard starts automatically. To use the wizard, provide the information it requests and click **Next** to continue until the wizard indicates that the Integration Object has been created.

The wizard prompts you for the information it needs to define the connection to the host. It will then connect you to the host server you specify and guide you as you log on to the host, navigate to the data you want to publish, select and organize the data, and disconnect from the host.

If you prefer to create your Integration Object manually, you can bypass the wizard and use the toolbar or menus.

Creating connection pools for Host Access

For each Integration Object, Host Access enables you to create a new connection pool, configure a default connection pool, or share a connection pool that you have already defined.

Applications do not share connection pools when they run on the server. But you can define multiple applications that use the same connection pool; a copy of the connection pool is included in each application. See "How applications are assembled and packaged" on page 45 for details.

This section describes the tasks involved in working with connection pools. For a general description of connection pools in Host Publisher Studio, see "Defining connection pools" on page 10.)

Defining the connection pool

You can use the **Connection Pools** tab in Host Access to define connection pools or follow the Host Access wizard to specify the connection pool. You can also use the Connection Pools tab to modify attributes of the pools, including connection configurations and user lists. You cannot use this tab to select the pool an Integration Object will use; use the Macros tab to select a pool for the Integration Object.

When you create a new connection pool in Host Access, connection pooling is enabled by default. However, when you configure a default connection pool, connection pooling is disabled by default.

To create a new pool, open the Host Access wizard and then:

1. Select **Create a new Integration Object**, then click **Next**.
2. Host Configuration appears as the title of the wizard pane. Select **Create a new pool** and then specify a pool name.
3. Click **Advanced** to open the Connection Pool Configuration window where you can enable connection pooling, specify how many connections you want to remain active and ready for use, and specify when to release connections. Click **OK** when you are done.

   Click **User List** if you want to configure a user list (see "Defining user lists" on page 26). When you are done configuring a user list, click **OK**.
4. In the Connection Information section in the wizard pane, define the connection configuration to use for the pool. Often, this will mean connecting to a host machine. You need to choose the type of connection you want, provide the name of the host, either as a TCP/IP hostname or as an IP address, and provide a specific logical unit (LU) or LU pool name, if any, to use. To save

time, you can define a connection to a particular host once, then share the configuration between pools that connect to the same host.

- To create a new connection configuration, select **Create a new connection configuration** and provide the information to establish a connection to the specified host.
- To use a previously-defined connection configuration, select **Share an existing connection configuration**. Then select a name from the list and skip to the end of the procedure.

5. Click **Advanced** to open the Connection Configuration window where you can define connection and security information for new connections. Click **OK** when you are done.
6. Specify the host server information.
7. If you enable express logon, when you choose to insert a user ID or password, you must define an application ID and a user ID and password. (See "Express logon considerations when using Host Publisher Studio" on page 83 for a detailed description of configuring for express logon.)

Customizing the connection pool

If you want to edit your connection pool information at any time, you can access all connection pools created in Host Publisher Studio from the **Connection Pools** tab. The pool used by the current Integration Object is highlighted.

Use the tabbed panes on the right to configure connection pooling options. You can:

- Configure time-outs and connection limits
- Change connection configuration information
- Change the user list associated with an Integration Object
- Edit the properties of a user list member
- Create a new user list

Recording interactions with a host

When you use the Host Access wizard, it guides you as you interact with the host to navigate to the screen that contains the data you want to publish, indicate the specific data, and define the way it should be presented. You use a terminal screen just as you normally would, and Host Publisher records your interactions as a macro. As you record, macro screens and actions are added to the macro tree view, which is displayed in the left pane.

In the data macro, you can use the **Gather Data** and **Do not Gather Data** choices to create an Integration Object that extracts data, or to create an Integration Object that only navigates through selected terminal screens without gathering any data.

Note: The connect and disconnect macros are part of the connection pool, not the Integration Object itself; however, the data macro is part of the Integration Object.

As you record your macro, you can stop and start recording at any time. If you select Connect, Data, or Disconnect macro in the tree diagram and then click **Record**, you begin recording at the end of the selected macro if the terminal screen matches the last screen in the macro selected. If the terminal screen does not match the last screen in the macro you selected, you cannot begin recording here.

If you select a screen with a definition matching the currently-displayed screen and then click **Record**, you begin recording at the selected screen prior to any previously recorded actions or keystrokes for this screen. If you click **Record** on any action or keystroke for a screen with a matching definition, you begin recording after the selected action or keystroke.

If you select a screen with a definition that does not match the currently-displayed screen and then click **Record**, a *jump screen* is inserted after the currently-selected screen and you begin recording after the jump screen. (A jump screen is a specific screen that follows a step in a macro; in effect, the macro jumps to that screen.)

You can use the toolbar and menus to perform many tasks, including:

- Recording interactions without using the wizard
- Defining an entry field
- Adding conditional paths or looping to the macros you create (see "Using input variables, conditionals, and looping" on page 22)
- Defining a screen

After you stop recording, you can use the shortcut menu to modify a specific step in the macro by selecting a step in the macro tree and right-clicking on it.

Later, when you execute your Integration Object on the Host Publisher Server, your recorded macros will run so that the steps you followed to access the data during creation of the Integration Object are the same steps used to return the data to the Web browser.

Defining a screen

Defining a screen provides a way for the macro to identify that it has reached the desired host screen. When the macro plays, it waits until the screen is recognized as defined before playing more keystrokes. (Each step in a macro has at least one screen defined as the possible next screen; Host Publisher waits for an expected screen to appear before playing the actions associated with that screen.)

When you are prompted to define a screen, before you can enter more information on the host terminal screen, you must either define the screen or respond that you do not want to define the screen. If you choose not to define the screen, the screen is unrecognized. When the completed macro plays an unrecognized screen, instead of waiting for the expected screen, the recorded keystrokes play regardless of which terminal screen appears.

In addition to the ability to recognize screens by text area, cursor position, and number of fields, the following recognition capabilities are provided.

- Comparison of text case: Case sensitive is selected by default.
- Comparison of text color to a specified color: You can change the coordinates of the color region and select different background and foreground colors.
- Comparison of a region to a value: You can change the start and end position coordinates of the defined region on the terminal screen.
- Comparison of two region values: You can change the start and end position coordinates of the first and second regions you defined on the terminal screen. The screen will be recognized whenever the two regions are the same—even if they are both blank.

- Multiple screen recognition criteria: You can combine more than one criteria to form a logical expression for a single screen, and you can choose whether each criterion is optional or non-optional.
- Logical NOT: You can match a specified criterion or not match a specified criterion.

Some tips for defining a screen in a macro are:

- Never define a screen by text that could change over time, such as a connection identifier, date, time stamp, or user ID.
- Define a screen using specific criteria to ensure that only one screen matches any set of next screens.
- Define a unique string that can be located anywhere on the screen without specifying its location; this eliminates the possibility of the string being in a different position and your screen not being recognized.
- Some screens arrive in segments, depending on the complexity of the screen.

  If you specify a screen description that identifies a part of the screen that arrives in one of the first segments, Host Publisher might recognize the screen and start performing the actions before the complete presentation space on the terminal has been updated. Therefore, make sure you always specify a detailed screen description to ensure that the last screen segment arrives before the actions in the tag are executed.

  To find out whether your host application screens are segmented, turn on tracing in your host emulator or on the TN3270 server.

  Note: Screen segmentation is controlled by the host application and not by Systems Network Architecture (SNA) parameters such as RU sizes or DLC MTU sizes. In SNA, a screen segment corresponds to a chain.
- If you cannot determine whether a screen is segmented, and you want to reliably recognize the screen in a macro, you can edit the macro and add a *pause* attribute to the screen that is displayed before the screen in question. Pause is the time delay, in milliseconds, between when actions of a screen description are performed and when valid next screens of the screen description are registered to the Host On-Demand screen recognition logic. The pause default is 200 milliseconds. This attribute must be modified using a text editor. For example:

  ```
  <screen name="ready.2" entryscreen="true" exitscreen="false"
   transient="false" pause="3000">
  ```

  Notes:

  1. If the host application sends various screen segments in response to inbound keystrokes sent during actions, this delay can be used to ensure that all segments of the screen update arrive before the next screen match is attempted. Pause introduces a delay in the macro execution, which affects performance and response time. If no application screens seen by a macro are segmented, setting pause to zero might result in improved Host Publisher runtime performance and scalability.
  2. There is a global pausetime attribute in the first tag in the macro that sets the default pause time for all screens in that macro. You can override this attribute by modifying the pause parameter.
- Define global screens and associated actions for those screens that might appear at any time, such as monthly reminders or messages from colleagues. Global screens enable you to define a generic action to take (such as a clear screen command) if such a screen is encountered, enabling the macro to return to the normal flow. See “Defining global screens” on page 18 below.

Note: If you use Host Access to record your macro, you cannot navigate back through a screen definition. To change the definition, you must right-click on the screen in the macro tree and click **Modify**.

Defining global screens

Global screens handle application screens that might appear at any time during the execution of your macro. When the same action is required each time such a screen is encountered, you can use global screens. During the execution of a macro, each *next* screen is checked sequentially in the order it is listed. If a next screen does not match, the set of defined global screens are checked for a match. If a match is found, the action associated with the global screen is performed and the macro continues. After the actions associated with the global screen are executed, the next screens are examined again for matches in the order they are listed.

For example, if you are recording a 3270 VM application logon sequence as a macro, you might notice that messages sometimes display after you type the user ID and password. Each time you see one of these messages, you press Clear to remove the message and continue with the macro. When you record the macro that defines these message screens, define the screens as global screens and then press Clear. In this way, you do not need to anticipate when the message screens will occur. The macro player automatically executes a clear when the screens are encountered in the connect, data, or disconnect macro.

Identifying data to extract

Extracting data from the host application is one of the major functions of an Integration Object. This is the data that will be displayed on a Web page in your Host Publisher application. In Host Access, data is extracted only in the data macro, and every host screen from which data is extracted must be defined.

If you selected **Gather Data** in the first window of the data macro, navigate in the terminal *session* to the desired data and then select the data for extraction. You can:

- Select a table of data
- Select a region of text data
- Select multiple regions on a screen by choosing **Extract More Data**

Assign a unique name to each region or box of text you select.

For example, if your Integration Object provides a connection to a telephone directory application, your Web user might enter a name to search for. The application displays a table of information about the people whose names match. You can publish all the columns in the table, or you can specify only some columns. You might want to publish only the names and phone numbers and not the office address or job description. You might also want to provide the name of the person's manager, which is displayed on another screen. Using Application Integrator, you can select all the information you want, arrange it, and display it to the user as though it were one piece of data.

You can also extract data by using an icon on the toolbar or by clicking **Insert > Data Extract** from the **Macro** menu.

Note: If you want an application to trace the x and y screen coordinates of the data that is being extracted, you can use a customized Integration Object template. Refer to the *IBM WebSphere Host Publisher Programmer's Guide and Reference* for details about modifying Java coding templates.

Generating an Integration Object

After you record your disconnect macro, you create an Integration Object by clicking **File > Save**. If you have deselected **Automatically Generate Integration Object on Save**, you create an Integration Object by clicking **File > Create Integration Object**.

You are prompted to name your Integration Object. Use a unique name for each Integration Object to avoid overwriting existing objects.

Verifying a macro

After you record a macro, verify that it works correctly. To verify a macro in Host Access:

1. Open the Integration Object that contains the macro you want to verify. You will see the steps of your macros displayed in the tree under the Macros tab. The connect macro is highlighted.
2. On the toolbar, click **Play** to play the connect macro.
3. If there are no errors, click **Play** again to play the data macro and inspect the captured data.
4. Click **Play** again to play the disconnect macro.

You will get a message if there are errors that prevent the macros from completing. If a macro does not complete correctly, it might be because there are some screens that appear that you did not define. You might need to define those screens (often these are global screens, such as screens that appear at different places and require you to press Clear or another key to continue).

Editing a macro

Advanced users can edit recorded macros in a text editor. To do this:

1. In the Host Access macro tree, stop recording. Then either right-click the Connect Macro, Data Macro, or Disconnect Macro and click **Modify**, or click **Modify** from the **Edit** menu.
2. Click **Edit** in the Global Macro Information window.
3. Make the necessary changes to the text in the Edit Macro window, then click **OK**.
4. Click **OK** on the Global Macro Information window to update the macro tree.
5. To save your changes, save the Integration Object associated with this macro.

For detailed information about the syntax of macro scripts, refer to the *IBM WebSphere Host Publisher Programmer's Guide and Reference*.

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. See "Integration Object chaining" on page 73 for a complete description of chaining.

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. See "Creating applications that

use Enterprise JavaBeans (EJB) technology" on page 42 for more information about incorporating EJB support into your Integration Objects.

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

For more information about changing these properties, see "Specifying default properties for EJB Integration Objects" on page 44.

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.

For more information about creating Web Services, see "Chapter 4. Using Web Services" on page 71.

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.

For more information about changing these properties, see "Specifying properties for Web Services Integration Objects" on page 72.

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 Host Publisher from a remote machine using Web Services" on page 72 for more information. For information about creating Remote Integration Objects, see "Accessing a remote machine using Remote Integration Objects" on page 50.

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. See "Accessing a remote machine using Remote Integration Objects" on page 50 for more information.

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.

Using advanced functions in Host Access

This section describes advanced functions you can use as you develop Integration Objects in Host Access:

- "Using input variables, conditionals, and looping"
- "Securing passwords and other sensitive data in macros" on page 26

Using input variables, conditionals, and looping

The macros you record using Host Access can be simple or complex. Simple macros follow a straightforward path; each step leads to only one next step. Most often, the macros you record will be complex; they will include steps that lead to a choice of several steps, or they will include sequences of steps that repeat. When you have an input variable, for example, the user could enter information that leads to another prompt or to a loop of repeated actions. "Using input variables" on page 23

on page 23 below, “Recording conditionals”, and “Recording loops” on page 24 , describe how you can use commands on the toolbar to add complexity to a simple macro.

Using input variables: You use input variables for information you want the user (or another Integration Object) to provide. This information is not coded into your macro and is provided when the user makes a page request. For example, if your application enables the user to search for information about a person, you could use an input variable to contain the name to search on.

To insert an input variable:

1. Start recording a macro.
2. When you reach the point where the user will enter information, click the **Insert input variable** button on the toolbar.
3. On the window that appears, type a name for the variable and the value for the field. You use the name when you design a Web page that uses this Integration Object. For example, you might type `person` for the input variable and `Smith, Mary` for the value of the input variable to be used during macro recording session. The value you provide is used only in the current recording and does not become part of the macro. The value can be null.

Recording conditionals: Conditionals enable you to handle the situation where a host application could respond to a command (or keystrokes) with more than one screen. Conditionals also enable you to record an alternate path for your macro to follow. When you record a conditional, your screen will have more than one next screen defined in the macro tree.

For example, Figure 2 shows an example of a conditional in a connect macro. When the connection is established, the terminal screen can display either a logged-in screen or a welcome screen that must be cleared before the logged-in screen is displayed. You can use **Insert Conditional** to define how Host Publisher handles either screen.

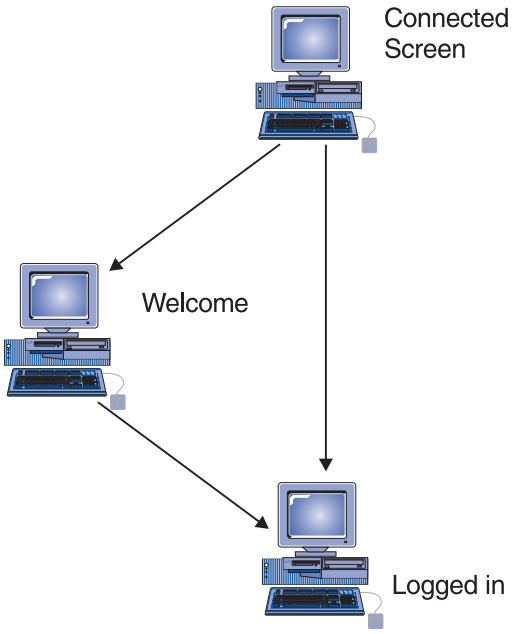


Figure 2. Host Publisher conditionals

To insert a conditional in a macro:

1. Record one path through the macro. For example, record the macro with the connection in a state so that the logged-in screen appears.
2. Put the connection in a different state, then replay the macro and cause the alternate condition to occur. Another way to cause an alternate condition is to specify a different value for an input prompt.
3. Host Access will report that a different-than-expected screen was encountered and display the Macro Play Error window. Click **Record an alternate path** in the Macro Play Error window.
4. Record the keystrokes you take to move from the new screen back to the main path of the macro, then stop recording. You might have several screens to define before you return to the macro. One way to get back to the main path is to highlight the last step in the conditional path and then click **Jump to defined screen** on the toolbar. On the window that appears, select the name of the screen in the main path you want to use as the destination for the jump. You must have already defined the screen before you can select it to be the destination of a jump.

   You can have several conditions for one step of a macro; for example, one screen might have several next screens associated with it in the macro tree. Each screen that you define has actions associated with it. When you play your macro and Host Access encounters one of the possible screens you specify, it performs the actions associated with that screen.

In the macro tree, the actions and the next screens associated with a screen are indented under the screen's entry.

Recording loops: Looping enables you to define an action that should be repeated.

For example, an application might return a list of data that requires more than one screen to display. You want your application to retrieve all of the data, so you need to define the macro in such a way that it displays each screen one at a time, retrieves data from each screen, and recognizes the last screen. The **Start loop** control on the toolbar enables you to do this. Figure 3 on page 25 illustrates the process.

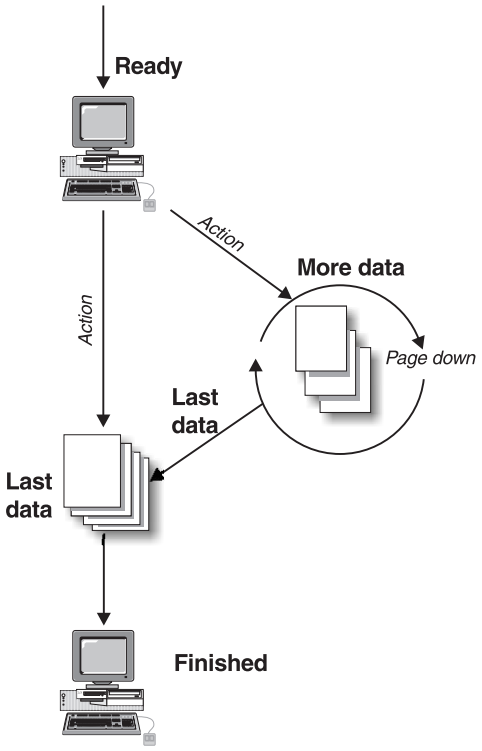


Figure 3. Host Publisher looping

To insert a loop in a macro:

1. Start recording your macro.
2. When you have reached the point where the loop will occur, click **Start loop** on the toolbar.
3. The Recording Loop wizard starts and provides the steps you need to perform to record the loop. Click **Next** to begin.
4. Follow the wizard instructions about gathering data.
5. Click **Next** to continue.
6. Type the keystrokes that will cause the terminal to advance to the second screen in the loop. Click **Next**.
7. Host Access stops recording your actions. You can now choose to stop the loop after a fixed number of iterations, or define a unique screen where the loop will stop. If you define a screen, be sure to supply criteria that uniquely differentiates that screen from other screens the macro is likely to encounter.
8. If you are defining a unique screen, Host Publisher stops recording while you navigate to the final screen of the loop. Host Access does not record the steps you take to navigate to the last screen because when the macro is running, this screen appears after the loop repeats. After you reach the screen that indicates the end of the loop, click **Next** to define the screen. In your definition, identify something on the screen that is displayed only when the loop is complete; for example, use words like *Bottom* or *Finished*, or use empty screen lines that do not appear until the last screen of the loop.

   If you choose to stop the loop after a fixed number of iterations, decide how many times you want the loop to repeat. Host Publisher stops recording and you can navigate to the final screen of the loop.
9. Click **Next**. The Recording Loop wizard ends and the loop is complete.

Note: If your loop contains a data extraction, when you play the macro in Host Publisher Studio an extraction occurs every time the loop executes. However, the macro behaves differently when it is played on the server. On the server, the extracted data is accumulated and then is returned to the application as one value at the completion of the loop.

Securing passwords and other sensitive data in macros

When you record a macro using Host Access, you can specify that user ID and password values be provided to a host application in a variety of ways when your macro runs on the server.

- They can be hard-coded in the macro; for example, for an iSeries or VT host where the same user ID and password can be used several times to log on to an application simultaneously. When you record or play a macro using Host Access, you might notice that hidden fields on the Host On-Demand terminal are placed in the macro tree as asterisks; however, when the macro is written out to the file system on the server or on the Host Publisher Studio machine, these fields are plain text and your sensitive data is visible.
- The user ID and password parameters can be provided externally (for example, by using an HTML form in the Web environment), and used to set the Integration Object's input properties that represent those macro parameters. To set up this option, click **Insert Input Variable** in Host Access.
- One or more user IDs and passwords can be defined in a user list. To define a user list in Host Access, see the following section, "Defining user lists".

  Application Integrator provides two levels of optional encryption to protect this information on the server. The user list is not encrypted in Host Publisher Studio. See "Setting security options" on page 49 for details.
- The user ID and password parameters can be provided using express logon. See "Express logon" on page 82 for more information.

Defining user lists

User lists contain information about accounts (user IDs) that an application can use to access a host. Each entry in a user list contains the user ID, password, and descriptions for an account. This list of user ID and password pairs provides values for macros to use at runtime when connecting to the data source.

Defining a user list to access a single-logon host

For an application that accesses a single-logon host, it is mandatory that the user IDs in a user list be unique to the application. This is because a connection cannot be created when a user ID is already in use—whether by the same application, another application, or a *clone* of the application. As a result, an application requesting such a connection would fail to run.

When you create a user list in Host Access, you can only create one entry (user ID and password) in the user list. You use this entry to record the connect macro so that it uses this user list to connect to the host. You can also use this entry to play the connect macro to test that it is correct. The user list is associated with a connection pool, which can be used with other Integration Objects that access the same host.

To help ensure that you have unique user IDs for applications that access a single-logon host, entries can be added to a user list only in Application Integrator. When you import an Integration Object associated with a user list into an application, a copy of the user list you created in Host Access is made in the application's directory. Using Application Integrator, you can customize the application-specific user list by adding users, deleting the entry you used to record

the connect macro, and making other changes. This is the user list that eventually is packaged into your application's .ear file.

Creating a new user list

To define a new user list, do one of the following:

- Select **Create a new pool** in the Host Configuration window. When you click the **User List** button, you are prompted to name the user list and put one user into it.
- Click the **Connection Pools** tab in the Object Configurations pane and select a connection pool. Then click the **User List Configuration** tab in the Pool Configuration window. When you click the **New** button, you are prompted to name the user list and put one user into it.

To create a connect macro that will utilize a user list, click the **Macro** tab and begin recording your macro. The Integration Object you are recording must use a connection pool that has a user list. Navigate to the place where you want to enter the user ID, select **Insert User ID** from the icon bar or menu bar, and click **OK**. The macro tree shows the entry as _userid.

To include the password, navigate to the place where you want to enter the password, then select **Insert Password**. The password associated with the user ID is inserted automatically. The macro tree shows the entry as _password.

If the Integration Object performs several logons to obtain data, and if you construct your user list so that all the user IDs and passwords are valid for all of the logons, then you can record your macro using a user ID and password from the Host Access toolbar for each logon attempt. The same user ID and password will be used for each logon attempt.

When you play the connect macro to test that it is correct, it uses the user ID and password in the user list to connect to the host.

Note: Data in user lists can be accessed only from a connect macro. If you write a data macro that logs on to a host application, you cannot obtain the user ID and password from a user list; you must use one of the other means defined in "Securing passwords and other sensitive data in macros" on page 26.

Working with user lists in Host Access and Application Integrator

Here is the process by which you customize a user list for an application that uses Host Access Integration Objects:

1. Create an Integration Object that utilizes a user list, as described in "Creating a new user list". The list contains one user ID, along with its password and a description. The list is stored in the SessionDefs directory as a file with the .userpool extension. (See *IBM WebSphere Host Publisher Programmer's Guide and Reference* for information about Host Publisher file formats.)
2. Using Application Integrator, import the Integration Object with which the user list is associated into an application. An application-specific copy of the user list is made in the application's directory. The application now has its own user list, which is not shared by other applications.
3. Customize the application-specific user list in Application Integrator. (See "Connection Pools" on page 38.) Customization tasks include:
   - Adding user IDs and passwords for the application to use when it connects to the host. For single-logon hosts, these user IDs and passwords must be unique: they cannot be in the user lists for any other applications.

- Deleting the original entry, with which you recorded and tested your connect macro in Host Access.
- Making other changes if desired.

Note that when you customize an application-specific user list in Application Integrator, the changes do not affect the original user list you defined in Host Access. Conversely, if you change a user list in Host Access, the changes do not affect copies of the user list in applications you have already developed in Application Integrator.

4. Assemble the application .ear file in Application Integrator by using the Transfer to Server wizard or by clicking **File > Create J2EE Archives**. The application-specific user list is included in the .ear file

If another application imports an Integration Object that uses the same connection pool (and the same user list), a new application-specific copy of the user list is created for that application.

Note: There are special considerations for defining user lists in a cloned environment. See "Cloning and user lists" on page 82 for more information.

Runtime processing of user lists

To illustrate how user lists function when applications are executed on the server, imagine that you have created an application that connects to a host and has various user accounts it can access to connect. The accounts have associated user IDs and passwords.

If you have a user list associated with the connection pool, and you have recorded your connect macro to use this list, Host Publisher uses the list you defined to supply the values that it passes on to the host when you run your Integration Object on Host Publisher Server. If the first user ID in the list is in use, Host Publisher uses the next user ID (unless you have specified that user IDs can be connected more than once).

Host Publisher tracks the IDs that are being used and updates the list as IDs become available for use; for example, imagine your application has three IDs and passwords as follows:

ID	Password
pam	pampw
sarkar	sarkarpw
villari	villaripw

At runtime, if **pam** is in use, Host Publisher Server uses **sarkar**. If **pam** then becomes available, Host Publisher uses **pam** for the next connection. If **pam** and **sarkar** are both in use, Host Publisher uses **villari**. If all three IDs are in use, Host Publisher rejects the request for a new connection, and your application must wait for the next available ID.

Customizing a user list created with an earlier version of Host Publisher

A user list defined in Host Access in Host Publisher Version 4.0 contains only one entry. A user list defined in a previous version of Host Publisher can contain more than one entry, but you cannot use Host Access to add additional entries. You must use Application Integrator to add entries.

To modify or delete entries in a user list that was defined in an earlier version of Host Publisher, click the **Connection Pools** tab in the left pane, select a connection pool, and click the **User List Configuration** tab in the right pane.

When you create an application using Application Integrator, copies of user lists become application specific. In Application Integrator, the **Modify User List** button in the Select Connection Pools window enables you to customize the user list associated with an application. You can add, edit, or remove users.

For information on adding additional properties to a user list, refer to the *IBM WebSphere Host Publisher Programmer's Guide and Reference*.

Creating a Database Access Integration Object

Use the Database Access application to create Integration Objects that encapsulate a database statement. To create the Integration Objects, you specify your structured query language (SQL) statement. If you are querying a database, you specify the data you want to retrieve from the database table.

This section provides information to help you with the following tasks:

Using the Database Access wizard

When you use Database Access to create an Integration Object, it launches the Database Access wizard. To start Database Access in Windows, click **Start > Programs > Host Publisher Studio > Database Access**.

The Database Access wizard starts automatically. Tabs in the wizard guide you through the process of building and executing a valid SQL statement, as you provide the requested information for each tab. You can navigate by clicking **Back** or **Next** at the bottom of the window. When you complete the wizard, the Integration Object is created when you save the file or when you click **Finish**.

Defining database connections

Before you can build your SQL statement to access the data you want to publish, you need to connect to your database. Select a database driver. Then replace [database] in the **Database URL** field with the name of the database or the host connection information for the database. Type any required user ID and password. Click **Connect** to connect to your database.

Notes:

1. Host Publisher does not install or set up Java Database Connectivity (JDBC) drivers. You must ensure that the drivers are installed on the machine running Host Publisher Studio and that they are at the same FixPack level as the drivers on the server where your applications will run.

2. So that you will be able to connect to your database, you should define the paths to the JDBC drivers in your machine's system environment classpath. On Windows platforms, you access the System Environment window from the Control Panel.

   (You can also define the path on the -classpath statement in the *install_dir*\Studio\dbaccess.bat file, and—if you launch Database Access from the Application Integrator—in the *install_dir*\Studio\webbridge.bat file, where *install_dir* is the directory in which Host Publisher is installed. However, we recommend that you define the path in your machine's system environment classpath instead of in the .bat files.)

Retrieving information from a database

The Database Access wizard helps you navigate to the tables that contain the data you want, specify conditions to identify the data, and sort the data you want to publish.

You specify the SQL statement type and select the tables in the database to access. You determine which columns of the tables to include, applying conditions to the data in the columns. You can also sort the order in which the data is displayed.

The SQL statement types are:

- Select
- Select Unique
- Insert
- Update
- Delete

After Database Access guides you through creating an Integration Object, you can use the Application Integrator application to create a Web page where the data will appear.

Note: When a database query (such as select) is made and the field is empty, the database returns a value of null. As a result, the Host Publisher database Integration Object returns a value of null to the invoker. By default, JSP pages generated in Application Integrator display null values as ″null″—not as a string of blanks. If you prefer that null values appear as blanks, you need to modify the JSP pages so that the null values are displayed as blanks.

Generating an Integration Object

After you create your SQL statement, click **Finish** (or click **File > Save**) to create an Integration Object.

After you import a completed Integration Object into an application and publish it, Host Publisher will use the SQL statement you created to access the data that will be returned to the Web browser.

Verifying an SQL statement

To verify your generated SQL statement, click **Run SQL** on the **SQL** tab. If successful, a result window displays your results with a maximum number of records. You can specify the maximum number of records to display by clicking **Run SQL Settings** This maximum is only used by Database Access to run a sample of your SQL statement. The maximum will not be used when your Integration Object executes on the server; all of your records will be returned when

executing on the server. Setting a maximum number of records to display using Database Access will avoid some out-of-memory conditions because your server is likely to have more memory than your Studio machine.

If an error occurs when running the generated SQL statement, the Database Access Exception Window opens. When this window opens, do the following:

1. Note the cause of the error.
2. Click **OK** to exit the window and return to the **SQL** tab.
3. Make corrections to the data on the appropriate tabs.
4. Return to the **SQL** tab.
5. Click **Run SQL** again.

Creating connection pools for Database Access

You can use the **Connection Pools** tab in Database Access to define connection pools. (For a general description of connection pools in Host Publisher Studio, see "Defining connection pools" on page 10.) You can also use the **Connection Pools** tab to modify attributes of the pools, including connection configurations and user lists.

For each Integration Object, Database Access enables you to create a new connection pool or share a connection pool that you have already defined.

If you have followed the tabs of the Database Access wizard in order, you have already connected to a database. The data on the **Connection Pools** tab will be primed with the database driver, database URL, user ID, and password you used when you connected to the database.

The default for connection pooling is disabled. Click **Pool Properties** if you want to enable connection pooling for this connection pool. If you do not use connection pooling, your Integration Object connects to the database each time you request a connection. Connection pooling keeps one or more connections to the database initialized, which might reduce the response time between when a client with a browser requests information and when it is displayed on the page, especially if you are using a remote database. You specify how many connections you want to remain active in the pool and ready for use, and when to remove connections from the pool.

You can create a user list, which contains a user ID and password associated with a particular connection pool. User lists contain information about accounts (user IDs, passwords, and descriptions) that a Host Publisher application can use to access a database. For more information about user lists, see "Defining user lists" on page 26.

To create a user list in the main Database Access window:

1. Click the **Connection Pools** tab.
2. Use the fields in the User List Configuration section to supply the user ID and password your application will use when connecting to the data.
   - The user ID and password default to the values used on the **Connect** tab to connect to the database.
   - The name of the user list defaults to the name given to the Integration Object when it is saved.

A user list is created by default, unless you select **Prompt for connection values at runtime**.

In Database Access, you can update the user list by opening it, clicking the **Connection Pools** tab, and modifying the information in the User List Configuration section.

In Database Access you can specify only one user ID and password for each connection pool. This is because user IDs and passwords for databases are always reusable; that is, they can be used by more than one active connection to the database.

User lists are created in the SessionDefs directory by Database Access. When you import an Integration Object into an application using the Application Integrator, the user list is copied to the application directory. This creates an application-specific copy of the user list, which you can customize. This copy of the user list becomes part of the application's .ear file before the application is transferred to the server.

Using the Options menu in Database Access

This section describes the selections you can make from the **Options** menu in Database Access.

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

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. See "Creating applications that use Enterprise JavaBeans (EJB) technology" on page 42 for more information about incorporating EJB support into your Integration Objects.

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

For more information about changing these properties, see "Specifying default properties for EJB Integration Objects" on page 44.

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.

For more information about creating Web Services, see “Chapter 4. Using Web Services” on page 71.

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.

For more information about changing these properties, see “Specifying properties for Web Services Integration Objects” on page 72.

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 Host Publisher from a remote machine using Web Services” on page 72 for more information. For information about creating Remote Integration Objects, see “Accessing a remote machine using Remote Integration Objects” on page 50.

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. See “Accessing a remote machine using Remote Integration Objects” on page 50 for more information.

The default prefix is *Remote*.

Using Application Integrator to build J2EE applications that use Integration Objects

Use the Application Integrator component of Host Publisher Studio to create Web pages that use the Integration Objects you created using Host Access or Database Access. Application Integrator enables you to import the Integration Objects, EJB Access Beans, and other Java beans or objects. (See “Creating applications that use Enterprise JavaBeans (EJB) technology” on page 42.)

This section provides information to help you with the following tasks:

- "Archiving application files" on page 37
- "Using the Options menu in Application Integrator" on page 38

For information on migrating applications, see "Migrating from previous versions of Host Publisher Studio" on page 51.

Specifying Integration Objects to publish to the application server

A Host Publisher application is a collection of Web pages, Integration Objects, and other Java objects that enable the end user to interact with multiple data sources.

Application Integrator enables you to build a series of Web pages using standard HTML tags in conjunction with special JavaServer Page (JSP) tags and Java code. These standard tags manipulate Java objects (such as an Integration Object) on the Web page. These tags also provide the ability to specify Java object output directly on the page.

JSP tags are designed for any Java object or bean. You can use Application Integrator to import other Java classes or beans, in addition to Host Publisher Integration Objects, into your application and publish them into Web pages.

Application Integrator accepts as input any Integration Objects, other Java components, or any prebuilt HTML pages to which you want to add data interactions. The output is a collection of Web pages (JSPs), that have been generated or modified to interact with Integration Objects.

The next section, Using the Application Integrator wizards, describes the wizards you use to build your Host Publisher application.

Using the Application Integrator wizards

When you use Application Integrator, you can build your application using wizards. The wizards guide you through naming the new application, importing Integration Objects and other Java objects into the application, and creating pages for publishing data.

To start Application Integrator in Windows, click **Start > Programs > Host Publisher Studio > Application Integrator**. Then click **Create Application**.

You can choose to create a new application or modify an existing application. To create a new application, specify an application name and follow the wizards using the **Next** and **Back** buttons at the bottom of the wizard windows. You can create an entire working application using the New Application wizard. Use the buttons at the bottom of the windows to move from one wizard to another.

During creation of Web pages in the Application Integrator, some of the windows contain multiple layers of buttons. Carefully review the instructions presented at the top of each Studio Wizard window. The required steps are listed in chronological order. Click **Next** only after you complete all of the actions required on the current window and are ready to move to the next step in the Web page creation process. Click **Back** when you want to return to the previous step in the Web page creation process. Click **Finish** only when you are ready to leave the Application Integrator wizard.

If you prefer to create your application manually, you can bypass the wizard and use the menus. To exit the wizard, click **Finish** at any time. Some of the items on the menus launch additional wizards to help you accomplish tasks.

The Application Integrator wizards are:

New Application wizard
This wizard guides you through naming the new application and selecting the pages and Integration Objects to add to the application. You can use this wizard to create an entire functioning application, or click the **Finish** button at any time to exit the wizard. If you exit before completing the wizard, use the pull-down menus on the Main window to complete your application. Launch this wizard by clicking **New Application** from the **File** menu.

New Integration Object wizard
This wizard guides you through adding an Integration Object to a Web page, defining the Integration Object inputs, and rendering the outputs. This wizard is launched by the New Application wizard or by clicking **Insert Integration Object**from the **Insert** menu.

New Page wizard
This wizard guides you through creating a new page, naming the page, and specifying its function. You identify the resources to add to the page, and whether the resources provide input to an Integration Object or render the output of an Integration Object. This wizard is launched by the New Application wizard or by clicking **HTML Page** from the **Create** menu.

Insert Output Control wizard
This wizard guides you through adding an output control to the current page. As part of creating the output control, you specify the Integration Object output to be displayed with the control. Host Publisher formats table, list box, text entry, and text area controls for you.

This wizard is launched by the New Page wizard, the New Integration Object wizard, and by various options on the **Insert** menu.

Insert an Input wizard
This wizard guides you through adding an input control to the current page. When creating the input control, identify the form that contains the new input control and the destination of the submit action for that form (if you are creating a new form).

Before an Integration Object runs, it might have inputs that require data. This data can come from an HTML form on another page, or from other Integration Object output. The wizard guides you through deciding how to satisfy Integration Object inputs.

Every Integration Object has an associated output method (*getter*) for each input variable specified. This enables the page to echo a value from an Integration Object as output, based on a value sent from another page. For example, on one page, a user requests the name of a person to use as input to an Integration Object on another page. The second page searches for a telephone number for the person and uses the output method for the name provided as input. You could construct a sentence on the page like this:

```
The phone number for (name) is (number)
```

where (name) is the value derived from the input and (number) is the phone number found by running the Integration Object.

This wizard is launched by the New Page wizard, by the New Integration Object wizard, and by various options on the **Insert** menu.

New Error Page wizard

This wizard guides you through defining an error page to display to the client when an error is encountered during processing of a JSP page in a Host Publisher application, including Integration Objects and EJB Access Beans used by the JSP page. Error pages contain code to handle exceptions and give available information on the type of error found. Use the error page to inform the client that an error occurred and how to address the problem, such as calling a service telephone number.

A default error page, DefaultErrorPage.jsp, is provided for you. When you define a new error page, all the Web pages in your application containing Integration Objects are updated with the name of the new error page as an input parameter to their Integration Objects. If you add Integration Objects to new or existing pages, the new error page name is added as an input parameter to those Integration Objects as well. (The previous error page remains in the application unless you remove it.)

By default, the most recent error page you define for an application is referenced by all the Web pages in the application. However, you can have some pages refer to a different error page by changing the **errorPage** attribute (of the **page** directive) in those pages.

This wizard is launched by the New Application wizard or by clicking **Error page** from the **Create** menu.

Notes:

1. The default error page, DefaultErrorPage.jsp, is in the language in effect for the workstation when Application Integrator was started. If you change the language settings for the workstation before assembling the application for transfer to the server, DefaultErrorPage.jsp will change to the language in effect at the time the application is assembled.
2. If you migrate applications from previous versions of Host Publisher, the error pages they contain are updated to comply with the JSP 1.1 and Java Servlet 2.2 specifications.

Transfer to Server wizard

This wizard guides you through the process of assembling applications and transferring them to Host Publisher Servers. To launch this wizard, click **File > Transfer to Server**.

All Host Publisher Servers to which you want to transfer your application must be configured using the Host Publisher Server Definition window. To open this window, click **Options > Preferences**, then click the **Servers** tab; or click **Server Info** on the first page of the Transfer to Server wizard.

For a complete description of the process for transferring applications to the server, see "Transferring applications to a Host Publisher Server" on page 45.

As you use the Transfer to Server wizard, you will encounter two key windows:

Select Connection Pools

When you build your Host Publisher application, each Integration Object you import has a connection pool associated with it. A copy of the connection pools is automatically included in each application.

Be sure to select all other connection pools the application could use. For example, if you plan to use different connection pools for production than you use for developing and testing the application, you might want to select several connection pools.

The Select Connection Pools window enables you to add and remove connection pools, and to modify the user lists associated with each connection pool. It is opened by the Transfer to Server wizard, by clicking **Select Connection Pools** from the **Options** menu, or by clicking **Create J2EE Archives** from the **File** menu.

Modify User List

When you import an Integration Object with a connection pool that includes a user list, an application-specific copy of the user list is created in the application directory. Use the Modify User List window to customize this copy of the user list. When you created the user list, it had only one user in it. Now, by adding users, deleting users, or editing information about users, you can customize the application-specific copy of the user list.

After the application-specific copy of the user list is assembled into an application's .ear (Enterprise Archive) file during the transfer-to-server process, any changes you make in Host Publisher Studio will not be reflected in the application on the server until you transfer it again.

For more information see "Defining user lists" on page 26.

Previewing a page

When you are creating an HTML or JavaServer page (JSP) in the Application Integrator, you can preview the page with a Web browser to verify the layout. The layout of the page is displayed, but the results of the JSP tags, Java code, and associated Java objects are not shown.

Sharing application files with other users of Host Publisher Studio

Application Integrator provides the *application bundler*: a way for you to *bundle* application files so you can share them with another user.

Note: Only users who are running the same version level of Host Publisher Studio can share files.

Click **File > Bundle Application Source** to create a .zip file of all source files associated with the current application—including Web pages, Java objects, and other resources. You can then send the .zip file to another user, who can unzip it and modify the files in Application Integrator.

Archiving application files

When an application is transferred to the server, all of its associated files—including Integration Objects, connection pools, and user lists—are packaged into an .ear (Enterprise Archive) file. For details about how this is done, see "How applications are assembled and packaged" on page 45.

You can preserve an archive of the application's .war, .jar, and .ear files without performing the file transfer. To do so, click **Create J2EE Archives** from the **File** menu. The resulting files are saved in the application directory.

A common use for this function is when you use a WebSphere Studio tool, such as WebSphere Studio Application Developer, for some of the work of building an application. See "Using WebSphere Studio tools with Host Publisher Studio" on page 51 for more information.

Using the Options menu in Application Integrator

This section describes the selections you can make from the **Options** menu in Application Integrator.

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

Preview Page

Select this option to preview the currently selected page in the Application Pages pane in a Web browser. The layout of the page is displayed, but the results of the JSP tags and associated Java objects are not shown.

Use the Preferences window to specify the Web browser for previewing the page. If you do not specify a browser, Application Integrator uses the system default browser.

Launch Personal Page Editor

Select this option to launch an editor application that can be used to customize a Web page (HTML or JSP page). This can be a Web editor or any other text editor that exists on your system.

When you modify a page, the page must be refreshed from the file system to display the changes. Click **File > Refresh Page** to refresh the page.

There is no default page editor. You can specify a default page editor using the Preferences window.

Preferences

Select this option to display the Preferences window, where you can specify the directories and paths for components of Host Publisher applications and the servers to which your applications are transferred.

Connection Pools

Select this option to display the Select Connection Pools window (see page 36), which lists connection pools associated with your application. Unless you modify this list, copies of these connection pools are included in the .ear (Enterprise Archive) file for this application when the application is assembled and transferred to the server.

See "Defining connection pools" on page 10 for more information about connection pools.

From the Select Connection Pools window, you can access the Modify User List window (see page 37).

Display Welcome Screen

When this option is selected, a welcome window opens whenever Application Integrator is started. Deselect this option if you do not want the welcome window to open.

The default is to display the welcome window.

Application Migration

Some applications created with previous versions of Host Publisher Studio need to be migrated or they cannot run on a Host Publisher Version 4.0 server.

When this option is selected, Application Integrator checks every application you open and notifies you when an application needs to be migrated. You are then given the opportunity to migrate the application. For more about migrating applications in Host Publisher Studio, see "Migrating applications in Host Publisher Studio" on page 52.

The default is to perform migration checking. For best performance we recommend that you migrate all of your applications immediately after you install Host Publisher Studio, and then use Application Integrator with this option deselected.

Creating composite applications

Composite applications combine multiple Integration Objects to produce a single stream of output information to the user. Composite applications enable you to use the output of one Integration Object as input to another or fill a table with data from several different Integration Objects that access different data sources. The four ways to produce a composite application using Host Publisher Studio are:

- "Combining Integration Object output"
- "Sequencing Integration Objects on a single page" on page 40
- "Sequencing Integration Objects on multiple pages" on page 40
- "Sequencing Integration Objects between non-adjacent pages" on page 41

Combining Integration Object output

This type of composite application contains multiple Integration Objects on a page and displays their combined output on the page. For example, a user could use an application to query his or her own employee information from a central human resources application and database. The application maintains departmental and contact information. The database maintains payroll and insurance information. With two Integration Objects, one that accesses the application and one that accesses the database, a table can be displayed on an output page containing the output from both Integration Objects. The information in the table appears to originate from a single source instead of two different applications.

To create this type of composite application, use the New Application wizard in Application Integrator:

1. Create a new application.
2. Import the Integration Objects into the application using **Import > Integration Object** from the **File** menu.
3. When defining how the Integration Objects are used, specify that they are to be added to the same output page.
4. If any of the Integration Objects require input information, specify that the input controls requesting the input data are all added to the same form on the same input page.
5. When defining how the output is to be rendered, you work with one Integration Object at a time. If you want to create an output table combining data from multiple Integration Objects, close the New Application wizard and click **Insert > Output Control > Table** from the menu bar. To select outputs

from multiple Integration Objects, you must insert the output control after the Integration Objects have executed and gathered data. Therefore, you must position your cursor after the last Integration Object that you want to include in the combined output, *or* click **No** when prompted Do you want to insert at the cursor on the current page? to create the table at the logical end of the page. The wizard guides you through the process of creating a table and selecting the Integration Object outputs to use to fill the table.

You should have two pages in your application. One page requests input in an input form and delivers it to the other page. The second page executes the Integration Objects and displays their data. If the Integration Objects require no input data, you have no input page.

Sequencing Integration Objects on a single page

You can use one Integration Object to gather data from a data source and send the data to another Integration Object as input. For example, one Integration Object takes a user's name as input and provides as output that user's employee department number. This department number is then sent as input to another Integration Object, which takes the department number and locates the members of the department using another application. The department list is displayed to the user on the page.

The easiest way to accomplish this is to have both Integration Objects on the same output page. Be sure that the Integration Objects appear on the page in the correct order.

To create this type of composite application, use the New Application wizard in Application Integrator:

1. Create a new application.
2. Import the Integration Objects into the application using **Import > Integration Object** from the **File** menu.
3. Define the first Integration Object in the logical sequence first. Specify how to provide input to this Integration Object, if applicable (using another form page, for example). Do not render any output from this Integration Object on the page. The output is used by the next Integration Object.
4. Define the second Integration Object. To satisfy its inputs, specify that they are to be satisfied using other Integration Object output.
5. Select the other Integration Object from that page as providing that output. Note that Integration Objects accept only single-valued inputs, so you can use only single-valued outputs from other Integration Objects as input to another Integration Object.
6. Render the output of the second Integration Object on the page.

You should have two pages in your application. One page requests input in an input form and delivers it to the other page. The second page executes the first Integration Object using the input, executes the second Integration Object using the first Integration Object's output as input, and displays its own output on the page.

More Integration Objects can be added to the page to lengthen the sequence.

Sequencing Integration Objects on multiple pages

You can use multiple Integration Objects to return data to the user in steps, enabling the user to act upon the data and return selected data for the next

Integration Object to use. For example, an application contains three pages. The first page uses a phone number-lookup application to request a name. The second page displays all matches found, enabling the user to select one. The third page displays the phone number of the selected individual.

This type of composite application consists of an initial form page followed by a series of pages. Each page contains an Integration Object and an input form filled with the output from that Integration Object. Each Integration Object executes, using the selections from the previous page as input, and fills the new input form on the current page with output. The user can make a selection and continue to the next Integration Object.

To create this type of composite application, use the New Application wizard in Application Integrator:

1. Create a new application.
2. Import the Integration Objects into the application using **Import > Integration Object** from the **File** menu.
3. For the first Integration Object, specify that it should be placed on output page output1. Satisfy the first Integration Object's inputs using an input form on a page called inputForm. This inputForm page will submit results to the output1 page.
4. Render the first Integration Object's output to a list box control. Because a list box also serves as an input control (you can select an item out of the list), a form is defined for it on the output1 page. Do not specify a page for this form's destination.
5. For the second Integration Object, specify that it should be placed on output page output2. Satisfy the second Integration Object's inputs using the input controls created on page output1. The input form on output1 is modified to point to output2 as the destination for the data.
6. Specify that the second Integration Object's output should go into a list box, creating a new form without a destination page.
7. Repeat steps 5 and 6 for the third Integration Object. Specify output3 as the output page and output2 as containing the form providing input.

The result is an interactive composite application that enables a user to interact with data before it is passed to the next Integration Object. The application consists of four pages: inputForm, output1, output2, and output3. Pages inputForm, output1, and output2 each have input forms that provide data to the Integration Object on the next page. Pages output1, output2, and output3 all show the data resulting from executing the Integration Objects on that page.

Sequencing Integration Objects between non-adjacent pages

You can create a composite application similar to the one described in "Sequencing Integration Objects on a single page" on page 40, but with Integration Objects that reside on different pages and do not require user input. The output of one Integration Object is stored within the Web session and retrieved by another Integration Object.

You can use this type of composite application to send an output value to an Integration Object as input on another page where an input form is not involved. For example, if there is no form between a page containing an Integration Object that returns an employee's department number and another page that takes the department number and returns all of the employees in that department, there is no way to send data from the first page to the next. Using Host Publisher Studio,

you can cause the first Integration Object to save the department number within the Web session to be retrieved by the second Integration Object. This method requires no interaction with the end user and does not demand that the two pages be adjacent to each other in the logical page hierarchy in your Web site.

To create this type of composite application, use the New Application wizard in Application Integrator:

1. Create a new application.
2. Import the Integration Objects into the application using **Import > Integration Object** from the **File** menu.
3. For the first Integration Object, specify that it should be executed on page execute1. If it has inputs to satisfy, create an input form on page input1. To create a <FORM> tag and to enable session transfer to take place, you must render at least one variable as a control that will serve as both output and input; for example, list box, password field, and so forth. This excludes tables and normal text, for which selection is not possible. The created form must specify that the expected page of the second Integration Object (execute2) will appear when the form is submitted. This links the pages as parent/child, makes the data from the first Integration Object available to that page, and enables the expected button for the next step.
4. For the second Integration Object, create a new page execute2, then define an input of this Integration Object that will be satisfied by an Integration Object output. Select an output variable of the first Integration Object.

When you complete the wizards, a statement will be added to the execute1 page to insert the Integration Object's output into the HTTP connection. On execute2, the value is extracted from the HTTP connection and set as input to the second Integration Object. You can change the destination of the form to a different page, or you can remove the form completely if you do not need it for other purposes. You must ensure that the normal page flow of your application causes execute1 to always occur before execute2.

Creating applications that use Enterprise JavaBeans (EJB) technology

Enterprise JavaBeans (EJB) is a server-side component architecture that enables rapid development of versatile, reusable, portable applications. This section describes common tasks associated with developing Host Publisher EJB applications.

EJB support in Host Publisher is intended for use by enterprises that run their back-end servers using EJB technology. With EJB support in Host Publisher, you can logically (or physically) partition the Integration Object's function of navigating and retrieving back-end enterprise data from the presentation and consumption of that data. This processing model is aligned with the J2EE three-tier application model.

For a more thorough description of EJB support in Host Publisher, and for information about how to perform advanced tasks, see "Performing advanced tasks with Enterprise JavaBeans (EJB)" on page 78.

Understanding EJB support in Host Publisher

Host Publisher provides support for executing Integration Objects in EJB containers to take advantage of the server-side characteristics provided by the EJB architecture. This support consists of the following parts:

- The Host Publisher EJB, which is a stateful session EJB capable of running Integration Objects in an EJB environment. (Stateful means that the EJB maintains a conversational state with the client for the duration of an application.) This EJB is contained in the .ear file for each application.
- Host Publisher Studio support for the generation of EJB support files for running Integration Objects. This includes, for each Integration Object, the generation of an EJB Access Bean that provides the same signature as the real Integration Object.
- Application Integrator support for building applications using the generated EJB support files and the Host Publisher EJB.

Because the EJB Access Bean has the same signature as the Integration Object, the EJB Access Bean can be used in client-side code exactly as the real Integration Object would have been used. Therefore, the client can be:

- A JSP page or a servlet that uses one or more EJB Access Beans in a J2EE application where the Integration Objects execute in an EJB container.
- A Java application client that uses one or more EJB Access Beans to execute the Integration Objects in an EJB container.
- A Java application thin client that uses one or more EJB Access Beans to execute the Integration Objects in an EJB container.

Refer to the WebSphere documentation for more information about Java application clients and Java application thin clients.

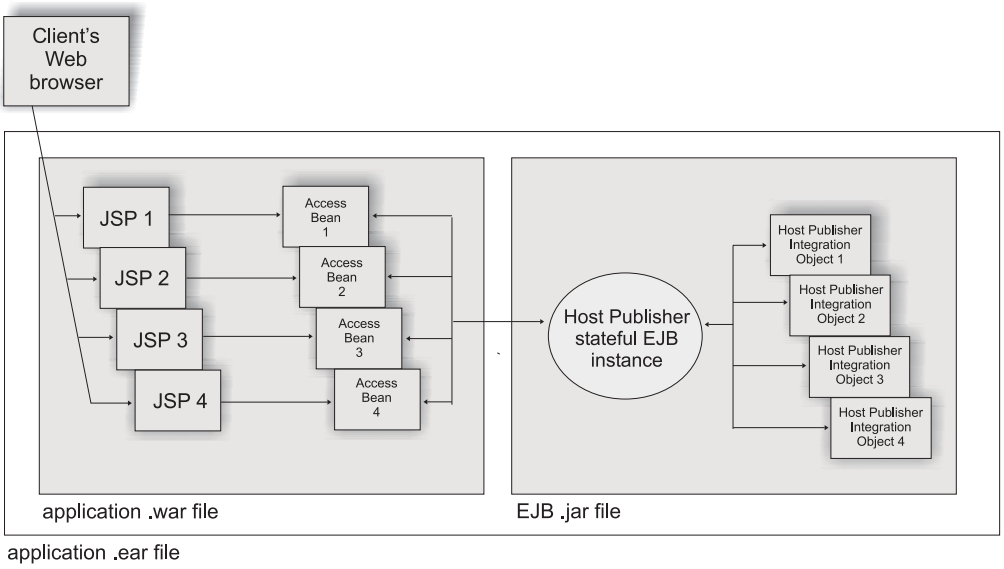


Figure 4. Components of a typical EJB application

Figure 4 illustrates the case where:

- The JSPs, packaged together with the EJB Access Beans in the application .war file, execute in a *Web container*.
- The Host Publisher EJB, packaged as an EJB .jar file, executes in a WebSphere EJB container.

Creating EJB support files for Integration Objects

If you intend to generate EJB support files for an Integration Object, you must so indicate before you create the Integration Object. Using Host Access or Database Access, click **Options > Create EJB 1.1 Integration Object Support**.

When this option is selected, EJB support files are generated during creation of the Integration Object. An EJB Access Bean .jar file is created in the \IntegrationObjects directory, and additional EJB support class files are created to be bundled in the EJB .jar file of the application. The default name for the EJB Access Bean .jar file is *IOName*Access1.jar, where *IOName* is the name of the Integration Object.

Specifying default properties for EJB Integration Objects

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

The default suffixes are:

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

To change the default suffixes for this option, edit the following properties in Studio.ini:

- EJB_PROPERTIES_SUFFIX
- EJB_HELPER_SUFFIX
- EJB11_ACCESS_SUFFIX

Creating a Host Publisher application using EJB Access Beans

After EJB Access Bean .jar files are created for Integration Objects, you can use them with Application Integrator in the same way that you use Integration Object .jar files to create Host Publisher applications.

You can create JSPs to instantiate, drive, and render the EJB Access Bean output properties in the same way that you use the original Integration Objects, including support for EJB Access Beans created for chained Integration Objects.

Before the application is transferred to the server, it is packaged into an .ear (Enterprise Archive) file that contains a Web module with all of its associated JSPs, EJB Access Beans, and an EJB module that contains the Host Publisher EJB, helper classes, and Integration Objects for the application.

Importing Java objects

You can import Java objects into a Host Publisher Studio application. The Java objects can be Java classes, beans, Host Publisher Integration Objects, or Host Publisher EJB Access Beans. You can import Java .class files, .zip files, or .jar files. These files can contain many Java objects. If you select a .zip or .jar file to import, you choose which Java object to import from the file.

If you import an Integration Object generated by one of the Host Publisher applications, such as the Database Access application or Host Access application, Application Integrator knows the inputs, outputs, and execution method.

You must ensure that both the class you are importing and all dependencies of that class can be located in the Java CLASSPATH in effect when the Studio started.

Note: All Host Publisher applications must use *UTF-8* encoding. If you import a Java object that was created outside Host Access or Database Access and that does not have UTF-8 encoding, you are prompted to migrate your application when you open it in Application Integrator.

For information about sharing Integration Objects among different users of Host Publisher Studio, see "Sharing application files with other users of Host Publisher Studio" on page 37.

Transferring applications to a Host Publisher Server

Application Integrator gives you the ability to transfer your J2EE application to one or more Host Publisher Servers, making the application ready to *install* as an Enterprise Application. Two things happen when you transfer an application to the server:

- The application and associated files are assembled and packaged into an .ear (Enterprise Archive) file.
- The .ear file is moved to the specified server or servers using file transfer protocol (FTP).

When you use the Transfer to Server wizard, you are prompted to select one or more servers, you can choose security options such as encryption, and you can make changes to resources such as connection pools and user lists.

How applications are assembled and packaged

Applications produced by Host Publisher Studio comply with J2EE standards. J2EE applications can be deployed rapidly and enhanced easily as the enterprise responds to competitive pressures.

In Application Integrator, the Transfer to Server wizard collects the application's Web pages, Java objects, and other resources and assembles them into a J2EE-compliant .ear (Enterprise Archive) file. By default, the .ear file is assembled in the *install_dir*\Studio\Publish directory. Then it is copied, along with its associated .war and .jar files, to the *install_dir*\Studio\Applications*appname* directory. (*install_dir* is the directory in which Host Publisher is installed and *appname* is the name of the application. Using the Preferences window on the **Options** menu, you can modify the directory where assembly takes place.)

Here is the process Application Integrator uses to assemble and package applications.

1. Application Integrator collects all Java objects and dependent files into temporary subdirectories for inclusion in the *appname*.ear, *appname*.war, and *appname*EJB.jar files it will generate.
   - For Host Publisher Integration Objects that are imported into Application Integrator, the dependent files include the .jar file (which contains the

Integration Object); the connection pool files; the connect, data, and disconnect macro files; the checkin screen file; and user ID and password list configuration files.

- For Host Publisher EJB Access Beans that are imported into Application Integrator, the dependent files include the EJB .jar file as well as all of the files required for the associated Host Publisher Integration Objects. The EJB .jar file contains the .class files for the Integration Object—including the BeanInfo, Helper, Properties, and IOProperties classes. The EJB .jar file also contains all of the HPubEJB2 interface classes and .properties files.
- For Java objects that are imported into Application Integrator but that are not Integration Objects, such as Java beans or classes, no dependent files are included.
- For all Java objects, Host Publisher expects prerequisite Java classes or Java beans which are not part of the application's .ear file to be defined in the CLASSPATH of the destination servers.

2. Application Integrator attempts to locate all dependent files that are needed by the application, such as image files, multimedia files, and other referenced pages. Application Integrator locates these dependent files if they are referenced by relative path names from within a page rather than by Web addresses (URLs). A relative path name describes the location of the file relative to the location of the owning file within the same file system.

   For example, when an image file is referenced as `IMG="images\mailbox.gif"`, this indicates that the `mailbox.gif` image file resides in a subdirectory called `images` within the application's directory.

   Only relative path names within the application's directory structure are allowed. For example, a relative path such as `..\images\mailbox.gif` would be within a parallel or peer directory to the application's directory space and would not be permitted in a J2EE-compliant server environment.

   If any files cannot be located based on the location of the owning page, Application Integrator notifies you that parts are missing. To complete the page, you need to locate the missing parts and add them to the application's directory structure in order for them to be included in the .ear file.

   If the resource references a Web address (URL), the page must rely on the application server, not Application Integrator, to locate the image.

3. Application Integrator generates the following application-specific files. (In each file name, *appname* is the name of the application.)

   For applications into which Integration Objects are imported:

   - *appname.war*: the .war (Web Archive) file containing the Web pages and default error pages for the application, along with Integration Objects and other Java classes or dependent files required by the application.

     Note: The default error page, DefaultErrorPage.jsp, is in the language in effect for the workstation at the time the application is assembled, even if a different language was in effect when the application was developed.

   - *appname.ear*: the .ear file, into which all of the application's pages, Java objects, and resources are assembled.

   For applications into which EJB Access Beans are imported:

   - *appnameEJB.jar*: the .jar file for a Host Publisher EJB-based application. It contains EJB Integration Objects, the HPubEJB2 class and properties files, and other dependent files required by the application.

- *appname.war*: the .war (Web Archive) file containing the Web pages and default error pages for the application.

  Note: The default error page, DefaultErrorPage.jsp, is in the language in effect for the workstation at the time the application is assembled, even if a different language was in effect when the application was developed.
- *appname.ear*: the .ear file, into which all of the application's pages, Java objects, and resources, including the *appnameEJB.jar* file, are assembled.

The files are saved in the application-specific subdirectory, *install_dir*\Studio\Applications*appname*. The .ear file is also kept temporarily in the *install_dir*\Studio\Publish subdirectory until the next application transfer.

After the assembly and packaging are complete, Application Integrator transfers the .ear file to the specified Host Publisher Server, where it is stored in a staging subdirectory (*WebSphere_install_dir*\installableApps\HostPublisher) until it is deployed. See "Selecting a server" for information about specifying the server.

After your Host Publisher applications are assembled into J2EE applications, they are self-contained. As such, they no longer share resources such as connection pools and user lists. So, for example, every Host Publisher application that is associated with a particular connection pool must include the connection pool definition in their .ear files. If you subsequently modify the connection pool, you need to reassemble every application that uses it and then transfer the reassembled applications to the server. See "How applications are assembled and packaged" on page 45 for details.

Before you transfer an application to the server, therefore, it is important to insure that it has all the objects and resources it needs. You can create an .ear file for an existing application at any time using the Create J2EE Archives wizard. (See "Archiving application files" on page 37.)

Also, the Transfer to Server wizard gives you the opportunity to update the configuration information before the transfer takes place. This wizard is described on page 36.

The J2EE specification provides details about the contents and layout of .ear files, .war files, and EJB .jar files.

Selecting a server

When the application has been packaged into an .ear file on the Host Publisher Studio machine, the file transfer protocol (FTP) is used to transfer the .ear file to the specified servers.

During the transfer process, you are prompted for the servers you want to transfer the application to. Each server must have a server information definition, created by clicking **Options > Preferences > Servers** or by clicking **Server Info** in the Transfer to Server wizard. When defining the server information, specify the target directory for the transfer and the type of platform. This target directory must correspond to an FTP alias that matches the WebSphere installation directory on that machine. Therefore, before a transfer can take place, the FTP service on that server must be configured to allow access to the WebSphere installation directory.

Table 1 lists some examples of target directories by platform:

Table 1. Examples of target directories

AIX	/usr/WebSphere/AppServer
Windows	C:\WebSphere\AppServer Note: FTP does not permit the use of a drive letter as a destination. You must configure the FTP service you are using on Windows to have an alias (such as \WebSphere) that points to C:*YourInstallDirectory*. You then specify \WebSphere as your target directory for this server.
Solaris	/opt/WebSphere/AppServer
iSeries with WebSphere AE	/QIBM/ProdData/WebASAdv4/AppServer
iSeries with WebSphere AEs	/QIBM/ProdData/WebASAEs4/AppServer

If Host Publisher Studio and Host Publisher Server are installed on the same system, you still must create a server information definition for the local system. Specify **localhost** as the host name so Host Publisher Studio knows to perform a local file copy instead of using FTP.

You can also use this method to transfer applications to remote Windows NT servers by sharing the target drive on the local system and performing a local transfer to that shared drive, again using **localhost** as the host name. In these cases, you must specify the drive letter and destination directory; for example: F:\WebSphere.

If a server was defined in a previous version of Host Publisher

If you defined servers using previous versions of Host Publisher, you must update those definitions for Host Publisher Version 4.0. If this is the case, the Transfer to Server wizard issues a message prompting you to update your server definitions. Previously, applications were transferred to the server directory in which Host Publisher Server was installed; now, they are transferred to the directory in which WebSphere is installed (for example c:\WebSphere\AppServer).

Note: The server name displayed in the Server Definition window is a default value, but it is not necessarily the currently-selected server. To accept the default value, therefore, you must click **OK**. Clicking **Cancel** does not update the server definition.

If an application already exists on the server

If you transfer an application that has the same name as an existing application, a warning message displays. You then have the options to:

- Continue the file transfer and overwrite the existing file.
- Stop the file transfer and keep the existing file.

Application transfer places the application files in a staging subdirectory (*WebSphere_install_dir*\installableApps\HostPublisher). The situation occurs only when two applications with the same name are transferred before one of them is deployed.

Setting security options

The Transfer to Server wizard asks you how to secure potentially sensitive user data contained in user lists, such as passwords. Any data in a user list except the user ID can be encrypted. You can choose to:

- Not secure the data, leaving the values readable from the configuration files
- Scramble the data (weak encryption)
- Encrypt the data (strong encryption)

Strong encryption requires that you specify an encryption password, which the server *administrator* must supply before applications can connect to the host. (See "Providing application passwords" on page 59.) When a J2EE application containing a strongly-encrypted user list is deployed on the server, the password must be specified before the application can service requests. If another user list is created with a property that requires strong encryption, the same password must be specified to encrypt that user list. Host Publisher Server can have only one encryption password, so the same password must be used by all strongly-encrypted user lists. If weak encryption is used, no password is required.

Modifying applications on the server

We recommend that you use Host Publisher Studio to modify your applications, but if you want to modify an application on the server you must ensure that the application's .ear (Enterprise Archive) file has been expanded before you make your modifications. You must also use a text editor compliant with UTF-8. (Because not all operating environments support UTF-8 editors, this might involve copying the application to a different machine, editing it, and then copying it back to the server.)

All Host Publisher Version 4.0 text files must be written with UTF-8 encoding. Whenever you attempt to modify an application on the server, you must use a text editor compliant with UTF-8. (If you use Host Publisher Studio to modify the application, Host Publisher checks for UTF-8 encoding and, if necessary, invokes a migration tool that updates the application so that it is written with UTF-8 encoding.)

Enabling tracing

You can enable tracing for the components of the Host Publisher Studio by editing the Studio.ini file located in the Studio directory. Find the entry for ENABLE_TRACE and change it to state ENABLE_TRACE=true. Tracing begins when the Host Publisher Studio components are restarted.

Trace information is written to separate files for each component:

Application Integrator	webbridge.trc
Host Access	hostaccess.trc
Database Access	dbaccess.trc

The trace files are located in the Studio directory.

Accessing a remote machine using Remote Integration Objects

You might want to write a program that does not execute in WebSphere but that requires access to an Integration Object. We recommend that you do this using Web Services—see "Accessing Host Publisher from a remote machine using Web Services" on page 72 for more information.

However, you can also create Remote Integration Objects (RIOs) for accessing Integration Object data from a Java program (applet or application) running on a remote machine. The remote machine requires lightweight RIO .jar files and network access to a Host Publisher Server; but it does not require Host Publisher Server or WebSphere Application Server. This might be advantageous if you are already using RIOs that were developed in previous versions of Host Publisher and you have not yet developed applications that use Web Services.

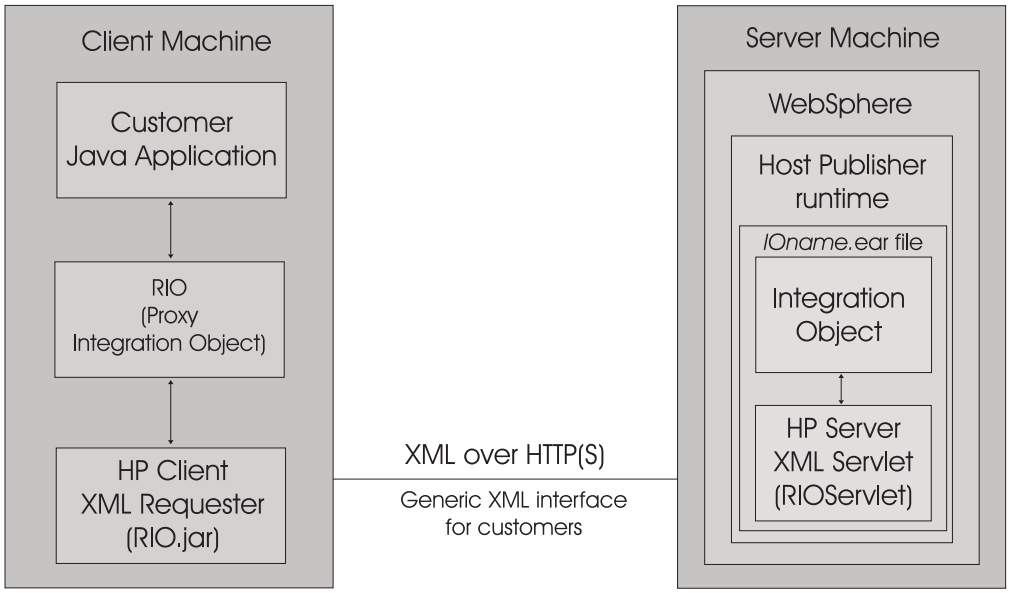


Figure 5. Using Remote Integration Objects

You can create a RIO using either Host Access or Database Access at the same time you create your Integration Object. This RIO is a proxy object with the same signature as the Integration Object. When your program invokes one of the methods of the RIO proxy object, the proxy object communicates with a RIO servlet, running on a WebSphere Web container, that can execute any Integration Object within that application server. The RIO servlet packages the results of the Integration Object invocation and passes it back to the RIO proxy object, and subsequently back to your program.

Each RIO application on the server has its own RIO servlet. When Application Integrator creates an .ear file for the application, you are given the option of including RIO support in the application archive.

The protocol used for communication between the proxy object and the servlet uses XML to encapsulate the method invocation and the input parameters and output parameters. The XML flows over an HTTP(S) connection. Because XML is used to interface with the RIO servlet, you can develop an XML application, written in Java, Perl, C++, or other languages, to access the servlet to execute your

Integration Objects. These applications do not require either the RIO proxy object or the RIO.jar file on the client, but do require HTTP access to the RIOServlet and an XML parser to process.

To create a RIO, click **Options > Create Remote Integration Object** as you are defining your Integration Object. Refer to the *IBM WebSphere Host Publisher Programmer's Guide and Reference* for more detailed information on RIOs.

In Host Access and Database Access, the **Options** menu has a **Remote Integration Objects Properties** selection. With this option you can specify a file prefix that is added to the names of the files (.java files) associated with the Integration Object. The prefix helps you easily locate the RIO-specific files that are generated by Host Publisher Studio. A default prefix is provided, but you can use the **Remote Integration Objects Properties** option to specify a different prefix for the RIO you are currently working on. To change the default value for this option, edit RIO_NAME_PREFIX in Studio.ini.

Using WebSphere Studio tools with Host Publisher Studio

Integration of Host Publisher with WebSphere Studio tools is flexible, in that you can choose how to divide work between Host Publisher Studio and the WebSphere Studio products. (In this section, the term WebSphere Studio refers to any of the WebSphere Studio tools, such as WebSphere Studio Application Developer.) In general, Host Publisher Studio is fine-tuned for building Web-to-host applications, whereas WebSphere Studio provides a more general-purpose studio for building and managing a wider variety of J2EE applications.

The Host Access and Database Access components of Host Publisher Studio create Java objects (Integration Objects, EJB Access Beans, Web Services helper files, and Remote Integration Objects) that encapsulate interactions with legacy data sources. 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 you have created with Host Publisher Studio into a new or existing WebSphere Studio project.

You can also build a J2EE application in WebSphere Studio using Host Publisher Integration Objects and other Java objects developed in Host Publisher Studio.

Application Integrator enables you to preserve an archive of the application's .war, .jar, and .ear files, all of which can be used in conjunction with WebSphere Studio. See "Archiving application files" on page 37 for more information.

For detailed information on how to perform these tasks, see *IBM WebSphere Host Publisher Programmer's Guide and Reference*.

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 67 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 FixPack 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 54 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 *IBM WebSphere 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 55 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 JSPs. 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 *IBM WebSphere 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.

Chapter 3. Using Host Publisher Server Administration

After you have published applications to Host Publisher Server, a server administrator can use an administration interface to manage connections and perform problem determination. This chapter describes how you can use these management functions, known as Host Publisher Server Administration.

Host Publisher Server Administration is a part of Host Publisher Server. Host Publisher Server provides a runtime environment—middleware needed to execute J2EE applications that access Host Publisher Integration Objects. Host Publisher Server Administration enables you to manage those middleware components.

Host Publisher Server Administration is Web based. You perform administration tasks using a Web browser, through a user interface provided by the Host Publisher Administration servlet, **HPAdminServlet**.

Using Host Publisher Server Administration, an administrator with the proper authentication can perform the following types of tasks:

- Selecting a server, and an instance of Host Publisher Server in that server, to administer
- Monitoring server status: starting and stopping Host Publisher Server
- Supplying passwords required by Host Publisher Server
- Managing licenses: changing the allowed number of licenses on a server running one or more instances of Host Publisher Server
- Administering connections and connection pools: displaying pool definitions, pool status, and the status of active connections for Host Publisher Server.
- Displaying user lists
- Performing problem determination: viewing trace and log files, and setting various options for tracing and logging
- Administering the XML Gateway: configuring the XML Gateway to access hosts and host applications, and creating a portal for accessing hosts and host applications

You can use Host Publisher Server Administration from any machine with HPAdminServlet installed, or remotely using distributed administration. (See “Administering Host Publisher from a remote machine” on page 64 for details.)

If WebSphere global security is enabled, you must have the proper authentication to use any of the functions in Host Publisher Server Administration. The administration client communicates only with administration servers that are at the same Host Publisher version level.

Other administration topics, which might be of interest to advanced users, are described in “Advanced Server topics” on page 65.

Getting started

This section describes how to start Host Publisher Server Administration and how it is assigned a name that identifies it in a WebSphere application server.

Starting Host Publisher Server Administration

To start Host Publisher Server Administration, load this URL in your browser: **http://***server***/HPAdmin/main.jsp** (where *server* is your server name)

When you start Server Administration, it uses the default language of the server. You can, however, change the language by clicking **Select Language** in the navigation frame, choosing the language you want, and then restarting the browser. The new language choice remains in effect for that browser until you invoke **Select Language** to change the language again.

Note: If you use a Netscape™ browser, we recommend that you use Netscape 6.1 or later (not 6.0) for Host Publisher Server Administration.

When Host Publisher is installed on the server, WebSphere Application Server is called *HostPubServer* and includes three J2EE applications:

- *HPAdmin.ear*: Host Publisher Server Administration. As you generate new applications with Host Publisher Studio, you deploy them into the application server using WebSphere's Install Enterprise Application wizard.
- *xmlLegacyGW.ear*: the XML Gateway function.
- *HPDoc.ear*: Host Publisher documentation.

Naming an instance of Host Publisher Server

Host Publisher Server, when it is initialized in a WebSphere application server, assigns itself a name that is unique within that machine with respect to other instances of Host Publisher Server that might be running on other WebSphere application servers.

The application server name is the same as the fully-qualified name of the application server, as defined by WebSphere. This name is *domain_node_server,* where:

- *domain* is the name of the WebSphere domain. (Note that the domain name can be null, in which case there is no underscore character (_) before the name of the node.)
- *node* is the name of the WebSphere node.
- *server* is the name of the WebSphere server.

Using the functions in Host Publisher Server Administration

This section guides you through the various selections on the Host Publisher Server Administration window.

Selecting host and application server

You can select a host to administer that is already known to the default server, or you can type the host name or IP address of a new host. You also have the option of selecting the default application server.

Monitoring server status

This window displays information about the current status of the server, including the number of licenses installed on this server.

From this window, you can start, stop, or restart the server.

Providing application passwords

If users will run applications containing strongly encrypted user lists, you must provide the Host Publisher encryption key. If users will run applications that use the express logon feature, you must provide the Host Publisher key ring password for express logon. Host Publisher Server requires these at runtime so that it can create the connection.

Supply the encryption key, the key ring password, or both after Host Publisher Server is started but before users begin making requests that require them. Using check boxes, you can specify whether the encryption key and the key ring password are saved to disk so that they are available to Host Publisher Server on subsequent restarts. (It is recommended that you have security procedures in place that mitigate the risk of saving passwords to disk.)

For more information about express logon, see "Express logon" on page 82.

Managing licenses

Host Publisher Server tracks the number of connections established to data sources and automatically logs a message when the value exceeds the number of licenses purchased. One license is equivalent to the right to use one Host Publisher connection at a time. The number of licenses applies to all Host Publisher instances running on all WebSphere application servers on that machine.

Host Publisher Server can optionally track license usage history over time. This history maintains the maximum number of licenses used (or connections established at one time) during a one-hour period, logging this information to a file each hour. By default, this option is set to 0, which means that no license tracking occurs. In the case of workload management (WLM) configuration, the license tracking is done across all application servers.

To enable the license tracking option, reset the licenseTracking property in the server.properties file from 0 to 1. The file license1.txt is created after Host Publisher is used for one hour and is located in the Log subdirectory under your Host Publisher installation directory.

For detailed information about editing the server.properties files, see "Appendix B. Server properties files" on page 123.

If you purchase more licenses and want to change your current value, you can do so using Host Publisher Server Administration.

Note: When Host Publisher is used in the Web Access environment, Client Access™ licensing is used, which does not work as previously described. Refer to WebSphere Application Server iSeries documentation for more information.

Monitoring connection pools

Connection pools are collections of communication links to back-end data sources, such as 3270 applications or databases. When an Integration Object is run on behalf of a client request, the Integration Object obtains an available connection from a pool, uses it for access to the data source, then returns the connection to the pool. When connection pooling is enabled, the overhead of establishing a

connection is absorbed in its first use. Each Integration Object reusing this connection benefits from the prior establishment of the connection and can run faster.

This window displays information about defined connection pools. A connection pool is not listed until at least one connection is allocated from the pool.

Monitoring pool definitions

A pool definition provides information about a collection of data source connections. Some related definitions are associated with a pool definition; for example, connection and macro definitions and user list name. In addition to creating associations between several other definitions, the pool definition provides configuration parameters for all connections in the pool, such as minimum and maximum pool size (in terms of number of active connections) and connection timers.

This window displays pool definition information for host and database connections. A pool definition is not listed until at least one connection is allocated from the defined pool.

Monitoring connections

Each connection shown in these displays represents a communication link to a back-end data source, such as a 3270 application or a database. The displays enable you to see details about which users are connecting to which data sources, and which connection identifiers they are using. You can also shut down sessions.

There are separate displays with which you can monitor:

- All connections
- Host connections: connections to 3270, 5250, or VT applications running on a host, iSeries server, or other server accessible via Telnet (VT)
- Database connections: connections to databases through the Java Database Connection (JDBC) interface

Monitoring user lists and user list members

User lists identify the user IDs and other connection-specific parameters associated with a particular connection pool definition. For systems that allow only a single connection per user ID, the number of user IDs determines the number of connections that can be active in a single pool.

For a description of user lists and how they are used, see “Defining user lists” on page 26.

Administering problem determination components

Host Publisher Server Administration enables you to set up and monitor logging and tracing to help you resolve problems.

Log and trace file names

The base log and trace file names in server.properties are used as templates to generate unique sets of log and trace files for each application server. The default base trace file name is *trace.txt*; the default base log file name is *messages.txt*. You can change these names in server.properties. The application server running Host

Publisher is the concatenation of the underscore (_) character, followed by the name of the Host Publisher Server instance, followed by another underscore (_) character.

The Host Publisher Server instance ID (for example *_SSS_*) is then appended to the base file name to generate the template for the log and trace files for an application server.

File naming example

Using the trace file as an example, the name becomes *trace _SSS_.txt*. Finally, an index (1, 2, 3, and so forth) is added to this name to distinguish multiple files. So, for example, if the Host Publisher Server instance ID is *domain_node_server*, the trace file for the application server is named *trace_domain_node_server_.txt*. With multiple trace files configured, the trace file names for this application server are *trace_domain_node_server_1.txt, trace_domain_node_server_2.txt*, and so forth.

When *trace_domain_node_server_1.txt* reaches maxTraceFileSize, it is closed and renamed to *trace_domain_node_server_2.txt*. A new *trace_domain_node_server_1.txt* file is opened.

When *trace_domain_node_server_1.txt* reaches maxTraceFileSize again, previous trace files are renamed—for example, *trace_domain_node_server_2.txt* is renamed to *trace_domain_node_server_3.txt*. Then *trace_domain_node_server_1.txt* is renamed to *trace_domain_node_server_2.txt*, and a new *trace_domain_node_server_1.txt* file is opened.

When the maxTraceFiles number is exceeded, the oldest file is deleted.

The same naming scheme is applied to log files. A typical log file, therefore, has a name like *messages_domain_node_server_1.txt*.

Version Information

The Version Information window displays the Host Publisher version, build level, and a list of Authorized Program Analysis Reports (APARs) that have been applied. Refer to this information when you plan to apply APARs or when you are talking with the IBM Software Support Center about a problem.

View Log

Using the View Log window, you can view the last 200 lines of the Host Publisher log file, download the entire file, or clear the file. The log file records information about three kinds of events:

Error events
: Problems that prevent an operation from completing. Error events usually require that you take some action to correct the problem.

Warning events
: Unexpected occurrences that might require action to correct the problem. Warning events are not as serious as error events.

Information events
: Normal occurrences, such as starting and stopping the Host Publisher Server. Information events do not require any action.

The log file is intended for you to read and use as a reference when troubleshooting problems. Most of the messages that appear in the log are

documented in the *IBM WebSphere Host Publisher Messages Reference*. That publication contains suggestions for actions you can take to correct problems, when necessary.

Note: In Windows NT and Windows 2000, you cannot rename or delete the standard output and standard error logs while Host Publisher Server is running. If you want to rename or delete these logs as part of troubleshooting procedures, to reclaim disk space, or to minimize the size of an information bundle file, you must stop the application server representing Host Publisher. However, the WebSphere node can remain running.

Set Log Options

Using the Set Log Options window, you can control whether information and warning events are written to the log file, and define the file to which the log events are written. If the file already exists, new events are appended to it. Error events are always written to the log file.

The log file name template is messages.txt and is created separately for each application server. The application server name is added as a suffix to each file name, in addition to the suffixes used to identify the number of files. For example, if the application server name is *domain_node_server*, and the log file name template defined in server.properties is *f:\HP4\Log\messages.txt*, and the number of log files is set to 2, the log files are named as follows.

f:\HP4\Log\messages_domain_node_server_1.txt

f:\HP4\Log\messages_domain_node_server_2.txt

If the file already exists, new events are appended to it.

You can change the template name of messages.txt by editing the server.properties file.

View Trace

Using the View Trace window, you can view the last 200 lines of the Host Publisher trace file, download the entire file, or clear the file.

The trace file records details of the internal operation of the Host Publisher Server, and is not necessarily intended for you to read. Typically, the trace facility is used when requested by IBM service.

Turning on tracing adversely affects the performance of Host Publisher Server.

Set Trace Options

Using these windows, you can choose to trace one or more of the following:

- Integration Objects on the server
- Host connections, using the tracing facility in Host On-Demand
- Database activity, using the JDBC tracing facility in WebSphere

You can also select the name and location of the trace file.

To set up tracing, use one or more of the following windows:

Server Tracing

Trace events on the server. This window enables you to select one or more of the following trace sources:

Server Tracing
Enables tracing in Host Publisher Server for connections, administration, and so forth.

Remote Integration Object (RIO) Tracing
Enables tracing in Remote Integration Objects that you have built.

Integration Object Tracing
Enables tracing in specific Integration Objects that you have built. Use the trace specification field to specify which Integration Objects to trace.

Note: You **must** select at least one option to enable any tracing on the server.

Host Connection Tracing
Select either or both types of tracing options:

User tracing for Host On-Demand macros
Traces the playing of macros. Because it affects system performance, we recommend that you use this trace only for debugging macros.

Display Terminal
Enables you to view the host terminal screen (*green screen*) of any Host Access Integration Object as it runs in real time on the server. After you set the trace option, terminal settings are shown for only newly created connections. For more information about the Display Terminal, see "Using Display Terminal for testing and debugging" on page 66.

Host Publisher Server Administration also includes a **Display Service Tracing Options** button with which you can initiate traces (if instructed to do so by IBM service). *We recommend that you not use these traces unless you are asked to do so by IBM service.*

Database (JDBC) Tracing
Turn on database tracing. This option enables tracing of all Java Database Connectivity (JDBC) activity in WebSphere Application Server and directs the output to the Host Publisher Server trace file.

Note: All JDBC activity, not just Host Publisher's, is traced when this option is enabled. It is possible for another application to also enable JDBC tracing and redirect the output to another location. This would include Host Publisher trace data.

Trace File Name
The trace file name template is trace.txt and is created separately for each application server. The application server name is added as a suffix to each file name, in addition to the suffixes used to identify the number of files; for example, if the application server name is *domain_node_server*, the trace file name template defined in server.properties is *f:\HP4\Log\trace.txt*, and the number of trace files is set to 2, the trace files are named as follows.

f:\HP4\Log\trace_domain_node_server_1.txt

f:\HP4\Log\trace_domain_node_server_2.txt

If the file already exists, new events are appended to it.

You can change the template name of trace.txt by editing the server.properties file.

Multiple log and trace files

By default, the size of the log and trace files is 512 KB, and two files are saved. If you prefer to work with a different number of log or trace files, or if you want to change the size of the files, you can edit a ras_*xxx*.properties file, located in the Server subdirectory where Host Publisher is installed. The keywords and their default values are:

```
maxLogFile=2
maxLogFileSize=512k
maxTraceFile=2
maxTraceFileSize=512k
```

For detailed information about editing ras_*xxx*.properties files, see "The ras_xxx.properties file" on page 125.

Administering XML Gateway sessions

Use Host Publisher Server Administration to define, delete, and configure XML Gateway sessions. When you have defined an XML Gateway session, you can access a host using the XML Gateway servlet and xmlAppData bean. For details, see "Chapter 6. Using the XML Gateway to enable simplified access to host applications" on page 87.

Administering applications

Application administration, formerly a function of Host Publisher Server Administration, is now done using WebSphere Application Server. For details, refer to "Administering Applications" in the IBM WebSphere InfoCenter.

Administering Host Publisher from a remote machine

Distributed administration enables you to perform administration tasks on instances of Host Publisher Server running in application server clones defined using WebSphere. (For more information about cloning, see "Application server cloning and load balancing in WebSphere" on page 79.)

The Host Publisher Server Administration function consists of two components:

- A Java Remote Method Invocation (RMI)-based Host Publisher *administration server* that is created in every Host Publisher Server instance that is initialized in a WebSphere application server.
- An *administration client*, HPAdminServlet, that can run in any WebSphere application server and can manage a Host Publisher Server instance in another WebSphere application server by making RMI calls to its administration server. HPAdminServlet is configured within the Host Publisher application server (HostPubServer), which is defined, using WebSphere configuration, when you install Host Publisher Server.

Having these two components enables you to administer any Host Publisher Server instance remotely. Examples of environments where remote administration is required include those where:

- Cloned application servers are defined within or across machines. In this instance, an attempt to access the URL for the Host Publisher Administration servlet cannot be directed to a specific application server, because every application server is identical.

- A Host Publisher user runs Integration Objects in an application server where HPAdminServlet is not deployed. In this case, the administrator must have access to at least one application server in which HPAdminServlet is running.

Before you can perform any administration tasks, you must select a Host Publisher Server instance to administer on a given server machine. The default is the server instance that is running on the same application server as HPAdminServlet. After you select the server instance, all administration functions are assumed to apply to that server instance until the selection is changed.

Advanced Server topics

This section covers advanced topics which might not apply to all users of Host Publisher Server.

Securing access to Host Publisher Server Administration using WebSphere Application Server

You can protect a WebSphere Application Server (WebSphere) administrative domain that consists of a cluster of servers using the same WebSphere administrative database. When you do this, Host Publisher operations are also protected. This means that your system administrators must have proper authentication (user ID and password) to perform Host Publisher administrative tasks.

The security functions used by Host Publisher in WebSphere are based on the J2EE form-based authentication process. When you enable global security in WebSphere, the administrator receives a logon prompt when he or she tries to access Host Publisher Server Administration. To run the servlet, the administrator must type the user ID and password that you defined.

When an administrator uses Host Publisher Server Administration to administer remotely a host or application server (running the Host Publisher server) on a protected WebSphere Application Server, he or she must provide a user ID and password. This user ID and password must match the user ID and password used for authentication on that WebSphere Application Server.

When WebSphere Application Server security is enabled, no Host Publisher Server Administration operations are available unless the user provides a valid WebSphere user ID and password.

In addition, when you use Lightweight Third Party Authentication (LTPA) as the WebSphere authentication mechanism, you must enable WebSphere single sign-on (SSO). WebSphere SSO requires you to use the fully-qualified host name in the URL for Host Publisher Server Administration. For example, when SSO is enabled, you would access Host Publisher Server Administration using http://hostname.mycompany.com/HPAdmin/main.jsp rather than http://hostname/HPAdmin/main.jsp.

Opening Host Publisher Server Administration in a new browser window

If you have a browser window open when you open Host Publisher Administration, by default the Host Publisher Administration utility opens in the

browser window. If you want to open Host Publisher Administration in a new browser window, you must change your browser settings as follows:

1. In Microsoft® Internet Explorer, click **Tools > Internet Options**.
2. Click the **Advanced** tab, then clear the **Reuse windows for launching shortcuts** check box.
3. Click **Apply**.
4. Close, then restart your browser.

Notes:

1. The **Reuse windows for launching shortcuts** check box is not available in Microsoft Internet Explorer versions previous to 5.5x.
2. Netscape does not provide options for launching shortcuts.

Using Display Terminal for testing and debugging

When debugging applications on a test system, you can control whether terminal screens are created on the Server display. Display screens are created only for connections established while this option is turned on. They are not created for existing connections or for connections that are idle. However, the Host Connections page in Host Publisher Server Administration includes a **Toggle Display** button that turns Display Terminal on and off for selected connections.

CAUTION:
Turning on the Display Terminal option can seriously affect performance or overload the server. Do not use this on servers with many connections. Display Terminal is intended for use in debugging during application development on a test system; it is not intended for use on a heavily loaded production server.

You can turn Display Terminal on for any new host connections by selecting the **Display Terminal** box on the Host Connection Tracing page; however, you can use **Toggle Display** to turn Display Terminal on and off at any time, regardless of the Host Connection Tracing setting.

Notes:

1. To use Display Terminal on OS/400, Remote Abstract Windowing Toolkit (RAWT) must be enabled.
2. If you are using WebSphere Application Server AE on Windows NT or Windows 2000, you must enable the WebSphere Administration Server service to interact with the desktop.

Configuring the Display Terminal function for iSeries

To display and use the **Display Terminal** check box, some additional configuration is required on the iSeries WebSphere Application Server and on your choice of a secondary network system that is capable of supporting Java Abstract Windowing Toolkit (AWT) JDK™ classes. This configuration is necessary because the iSeries server does not support graphical display devices as direct connect devices; it does support them as client and server roles. Java has a graphical solution for this called Remote AWT. In this case, another system, perhaps Windows NT/9x, acting as an AWT server, handles the graphical display of the terminal screen and interaction of Java code running in the primary environment; specifically, Host Publisher Server for iSeries. Remote AWT works like this:

1. Configure your Windows NT/9x system to run the Java JDK Remote AWT server/listener.

2. Configure your iSeries WebSphere Application Server, which also runs Host Publisher Server, to send all AWT requests and functions to the Windows server/listener.
3. When setup and configuration is complete, use OS/400 commands to stop and start WebSphere Application Server.

   Host Publisher Server Administration detects whether Remote AWT is enabled and, if it is, displays the **Display Terminal** check box located in Server Administration at **Problem Determination > Set Trace Options > Host Connection Tracing**.
4. Select this check box and submit changes.
5. Click **Server Status** in Host Publisher Server Administration, then restart Host Publisher Server to enable Display Terminal support. Whenever the **Display Terminal** check box is changed, you must restart Host Publisher Server to ensure that the new setting is read.

Note: Do not disable Remote AWT without first clearing the **Display Terminal** check box and submitting changes in Host Publisher Server Administration. If you do not follow this sequence, a Host Publisher Server 500 Internal Server Error will occur every time a Host Access Integration Object is run. The correct sequence is to clear the **Display Terminal** check box and submit changes, then disable Remote AWT support in WebSphere Application Server.

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 51 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 51.

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" on page 68.

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 70.
- "Updating server properties files" on page 70.

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 FixPack 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 54 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. Server properties files” on page 123. 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.

Chapter 4. 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 and 2, see "Chapter 2. Using Host Publisher Studio to develop J2EE applications" on page 9. For a detailed description of the rest of this process, refer to *IBM WebSphere 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.

Chapter 5. Advanced features

This chapter discusses how to use advanced features in Host Publisher: Integration Object chaining, advanced EJB topics, cloning and load balancing, express logon, and security.

Integration Object chaining

When you use the Host Access component of Host Publisher Studio, Integration Object chaining enables you to create multiple Integration Objects which can be grouped together into a single major task within your Host Publisher application. Each Integration Object performs one subtask, and the major task is performed by an Integration Object chain. When the application is executed, the Integration Objects execute in sequence, each using the same connection.

This section describes Integration Object chaining in detail and shows you how to use it in your Host Publisher applications.

Deciding when to use Integration Object chaining

To determine how many Integration Objects are needed in the chain, begin by listing all of the subtasks that need to be performed. For example, imagine a host application called *fileview* that, when invoked, displays a list of files. From the list of files, a user can select any file by typing 1 next to its name, pressing **Enter**, then viewing its contents. There are two tasks to perform here:

1. Obtain the list of files to present to the user.
2. Retrieve file details for a selected file.

Two macros are required because the user must make a decision before the second macro can execute. The second macro must wait for user input. Defining where user input is required is the first step in separating your tasks into individual Integration Objects.

If the user has multiple choices and each choice causes different host actions, then each choice must be a separate Integration Object. You can then piece together the Integration Objects dynamically, depending on the selections of the user. An example of this is a menu of five items. If the user is allowed to select any of the five choices, each selection must be created as an independent Integration Object.

After you understand the number of distinct tasks in your application, you can decide how chaining will affect your application. If you have more than two tasks, chaining might help improve your application's response time or decrease the amount of macro recording you must do, thus reducing the overall complexity of your Integration Objects.

Using Integration Object chaining

Integration Object chaining can be used only with Host Access Integration Objects, which have connections to a terminal-based application, such as a 3270 application. An Integration Object in a chain leaves the connection in a state (at a particular screen). After the Integration Object finishes running, another Integration Object that begins in that state (that is, at that screen) can run.

You can use chaining to break up a complex application into multiple tasks, each task represented by an Integration Object. The Application Integrator component of Host Publisher Studio ensures that the order in which Integration Objects are invoked is correct; however, if you edit the .jsp files without using the Studio, you must ensure that the order of the Integration Objects is correct.

For example, if you have three Integration Objects in a chain-A, B, and C-then you must use A first, then B, then C. If Integration Object C is invoked before Integration Object B, then when C requests its connection, the connection is not available in the correct state, and the Integration Object fails. Host Publisher Studio ensures the correct order for you.

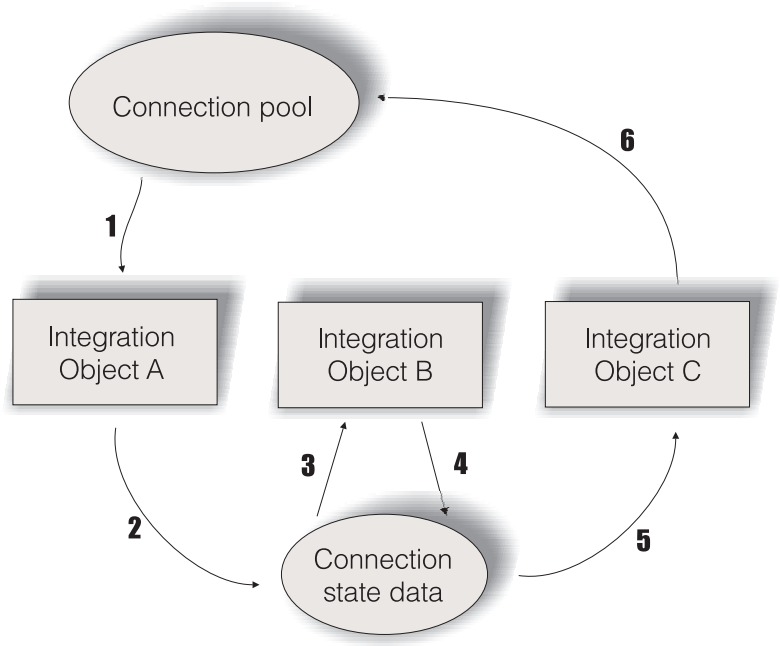


Figure 6. Connection lifetime with chaining

Figure 6 depicts the lifetime of a connection throughout the execution of three Integration Objects. Integration Object A is configured as first, Integration Object B as middle, and Integration Object C as last. Connection state data represents the connection and the state label of the last Integration Object executed.

1. When Integration Object A begins to execute, it retrieves a connection from the connection pool. If connection pooling is enabled and a connection is available, the connection is already logged on and ready. If connection pooling is enabled, but a connection is not available, the connection is created and the connect macro is executed. If connection pooling is disabled, a connection is created and the connect macro is executed.
2. Integration Object A runs the associated data macro, then saves the connection and its current state for the next invocation. (You would have defined this state as the stop state label during Integration Object chaining configuration in Host Access.)
3. When Integration Object B begins to execute, it retrieves a connection and its state from the connection state data because it is the middle Integration Object. You must have defined Integration Object B's start state label as Integration Object A's stop state label, which allows these Integration Objects to be chained.

4. When execution completes for Integration Object B, the connection and its state are saved in the connection state data. (You would have defined this state as the stop state label during Integration Object chaining configuration in Host Access.)
5. Integration Object C also retrieves the connection from the connection state data. When Integration Object C begins to execute, it retrieves a connection and its state from the connection state data because it is the last Integration Object. You must have defined Integration Object C's start state label as Integration Object B's stop state label, which allows these Integration Objects to be chained. Last-in-chain Integration Objects have no end labels because the connection is always returned to the connection pool.
6. When connection pooling is enabled, the connection returns to the pool; otherwise, the disconnect macro is executed and the connection ends.

To build an application using Integration Object chaining with Host Publisher Studio, you must first build the Integration Objects within the Host Access application, and then import these Integration Objects into Application Integrator and build Web pages around them.

To build the first Integration Object in the Integration Object chain:

1. Use the wizard in the Host Access application to define a connection as you normally would (see "Using the Host Access wizard" on page 13 for information about defining connections). All of the Integration Objects in the chain will use this same connection pool.
2. Record the connect macro. (See "Recording interactions with a host" on page 15 for additional information.)
3. Record your data macro.
4. You will probably not end your data macro at the same point where you began (so that another Integration Object can pick up where you've stopped). Navigate to the desired ending point for the data macro, and click **Stop Recording** on the toolbar.
5. Navigate back to the point where you can disconnect from the host; that is, where the data macro of the last bean in the chain ends.
6. Highlight the **Disconnect Macro** in the macro tree and click **Record**.
7. Record your disconnect macro.
8. Before you create the Integration Object, click **Options > Configure Integration Object chaining**. On the window that opens, specify a stop state label for this Integration Object and specify that this Integration Object should be **first**. Host Publisher uses start and stop state labels to identify the Integration Objects that can precede or follow this one. For example, if this Integration Object has a stop state label of **connected**, only Integration Objects that have **connected** as a start state label can follow this Integration Object in a chain.
9. Save the Integration Object.

Note: The connect and disconnect macros are part of the connection pool, not the Integration Object itself; however, the data macro is part of the Integration Object.

To build a middle Integration Object in the chain:

1. Use the wizard in the Host Access application to create another Integration Object. When you define the connection, click **Share an existing connection**

pool and select the configuration you used for the first Integration Object. All Integration Objects in the same Integration Object chain must use the same connection configuration.

2. Either play the connect macro or connect manually.

   Note: When you create a middle Integration Object, the connect and disconnect macros are displayed in the macro tree; however they do not necessarily run for that Integration Object when it is executed on the server. The connect macro runs before the first Integration Object and the disconnect macro runs after the last Integration Object.

3. You can play the data macros of preceding Integration Objects in the chain or use the terminal to navigate to the correct starting point for this Integration Object. To play the preceding macros, select **Play Another Macro** on the **Options** menu. Host Access plays the macros of preceding Integration Objects in the chain that you select. If the macro does not complete successfully, a window opens telling you why the macro did not play. You can play as many preceding macros as you want. Then record your new data macro by selecting **Data Macro** in the tree and clicking **Record**.
4. Click **Options > Configure Integration Object chaining**. On the window that opens, specify a start state label for this Integration Object that matches the stop state label of the previous Integration Object in the chain, specify a stop state label, and specify that this Integration Object should be **middle**.
5. Save the Integration Object.

To build the last Integration Object in the chain:

1. Use the wizard in the Host Access application to create the final Integration Object. When you define the connection, click **Share an existing connection pool** and select the configuration you used for the first Integration Object. All Integration Objects in the same Integration Object chain must use the same connection configuration.
2. Either play the connect macro or connect manually.

   Note: When you create the last Integration Object, the full connect and disconnect macros are displayed in the macro tree; however they do not necessarily run for that Integration Object when it is executed on the server. The connect macro runs before the first Integration Object and disconnect runs after the last Integration Object.

3. You can play the data macros of preceding Integration Objects in the chain or use the terminal to navigate to the correct starting point for this Integration Object. To play the preceding macros, select **Play Another Macro** on the **Options** menu. Host Access plays the macros of preceding Integration Objects in the chain that you select. If the macro does not complete successfully, a window opens telling you why the macro did not play. You can play as many preceding macros as you want. Then record your new data macro by selecting **Data Macro** in the tree and clicking **Record**. Be sure to end at the same point where the first Integration Object in the chain started.
4. Click **Options > Configure Integration Object chaining**. On the window that opens, specify a start state label for this Integration Object that matches the stop state label of the previous Integration Object in the chain, and specify that this Integration Object should be **last**.

   Note: For the last in the chain, you supply only the start state label.

5. Save the Integration Object.

The last Integration Object in your Integration Object chain should return the host screen to the same state from which the first Integration Object started. To verify that this happens, play your disconnect macro.

After you have all of your Integration Objects defined, return to Host Publisher Studio, and import them into a new application. In Application Integrator, the Available Objects tree, when expanded for an Integration Object, specifies the logical next and previous Integration Objects for the selected Integration Object. This makes it easier for you to identify the order in which the Integration Objects can be used.

When you have finished building your application, transfer it to a server, and deploy it. (See "Transferring applications to a Host Publisher Server" on page 45 for details.)

Debugging applications that use Integration Object chaining

While Integration Object chaining is a powerful tool for modeling the most complex host applications, take care when assembling your applications. Application Integrator helps you build J2EE applications using chained Integration Objects by ensuring that Integration Objects are invoked in the proper order. This does not prevent you from trying to put the Integration Objects in a different order or from linking pages back to other pages that are earlier in the chain. If you do not use Application Integrator to create the application, there is a risk of creating a J2EE application with Integration Objects that are invoked out of order; therefore, you should be aware of the errors you get on the server when this happens.

Remember that the start and stop state labels for an Integration Object should model the screen that they represent. If an Integration Object starts on screen A and ends on screen B, the start and stop state labels should be different. If the Integration Object starts and ends on screen A, then the start and stop state labels should be the same; however, start and stop state labels are only labels. They have no direct correlation to the screens they represent. It is possible, for example, to give all your start and stop state labels the same label, even though the Integration Objects start and stop on different screens. Likewise, you can give two Integration Objects start and stop state labels that differ, even if one ends and the other starts on the same screen. We recommend that each label name uniquely identify a screen.

If you chain the Integration Objects incorrectly, you will experience problems when you run your Integration Object on Host Publisher Server. Here are the error messages you might see on the server and how to debug them.

HPS5075 Received STATE_PLAY_ERROR while playing macro in file zzz.macro
: This error occurs when the macro fails to play because it cannot match the current screen. This might happen if an Integration Object in a chain is invoked in the correct order (a connection in a specific chain was found for it to use), but the macro itself failed to play. It can be a problem if your start and stop state labels are all the same, or at least are the same for two Integration Objects, and you invoke the Integration Objects out of logical order.

HPS5035 There is no data source object {0} in HttpSession. A possible cause is the use of multiple browsers from a single machine to a chained application. See documentation for more information.
: This error occurs when an Integration Object with a start state label of last

Integration Object state fails to retrieve the connection and its state because its start state label does not match the stop state label of the preceding Integration Object.

When this happens, determine why this Integration Object is being invoked out of order.

- Are you sure that another Integration Object (either a first or middle Integration Object) has already run and has placed its connection in the state labeled last Integration Object state (its stop state label)?
- Are you sure that the JSP page running this Integration Object was linked in the correct order?
- Is there another Integration Object on the JSP that should have put the connection into the state labeled last Integration Object state and that failed? This does not always prevent other Integration Objects on the page from being invoked. You might be looking at a chain of errors.

In any case, check the message log for the Host Publisher Server and look for reasons for the problem. Refer to message HPS5035 in *IBM WebSphere Host Publisher Messages Reference* for details.

Performing advanced tasks with Enterprise JavaBeans (EJB)

Enterprise JavaBeans (EJB) is a server-side component architecture that enables rapid development of versatile, reusable, portable applications. This section describes the EJB support available in Host Publisher and describes the configuration tasks you perform for executing Integration Objects in EJB containers. For a description of the more basic EJB support in Host Publisher, see "Creating applications that use Enterprise JavaBeans (EJB) technology" on page 42.

A Host Publisher application can take part in WebSphere's workload management (WLM) function and is resource-managed by the EJB container, including passivation and activation of EJB instances as needed.

Note: EJB support can be generated for all Integration Objects except for those configured to use express logon.

For information about programming with EJB Access Beans, refer to the *IBM WebSphere Host Publisher Programmer's Guide and Reference.*

Modifying Host Publisher EJB-based applications

When you import an EJB Access Bean into your application, the Host Publisher EJB is included in the .ear file for the application. You can import the .ear file into the WebSphere application assembly tool to modify the default values set by Host Publisher Studio.

The following are Host Publisher attributes you might want to modify when customizing a Host Publisher EJB-based application.

- **JNDI name**

  The **JNDI name** attribute specifies the network name with which the Host Publisher EJB is registered in the *Java Naming and Directory Interface (JNDI)* directory and is used by the client (the Access Bean) to locate the EJB. In most cases the default name is appropriate; however, if you want to deploy the Host Publisher EJB-based application in more than one application server in the same WebSphere domain, the EJB within each application must have a different JNDI name.

- **Timeout value**

  The **Session Timeout** attribute specifies the number of seconds of inactivity of a bean instance before it times out. In most cases, the default value is appropriate; however, you can specify a different value that is specific to your applications and work environment. The session timeout value for the installed Host Publisher EJB is 600.

Changing the default values when running in a non-J2EE environment

For every Integration Object for which EJB support is generated, an EJB Access Bean properties file (for example, *IOName*Access1.properties) is included in the EJB Access Bean .jar file. This .properties file specifies parameter values used by the EJB Access Bean when it runs in a non-J2EE environment—for example, when used by a Java application thin client. (When running in a J2EE environment, the deployment descriptor for the EJB .jar file contains corresponding values.)

For a non-J2EE environment, the .properties file specifies the host name and port number of the JNDI server. The .properties file also specifies the JNDI name of the Host Publisher EJB, but this value is set by the Host Publisher Application Integrator when it builds an EJB-based application. The default values assume a local JNDI server.

To use values other than the defaults for the JNDI server, you can do one of the following:

- Edit the *Hostpub_install_dir*\Studio\IntegrationObjects\EJB\AccessEJB.properties file prior to generating the Integration Object. (This file is used as input to create the .properties file in the EJB Access Bean .jar file.)
- Replace the .properties file in the EJB Access Bean .jar file with a modified copy after the .jar file is generated.

Application server cloning and load balancing in WebSphere

WebSphere Application Server provides support for running multiple application servers on a single machine. Each application server consists of two distinct execution environments: a *Web container* for executing applications such as servlets and JSPs, and an *EJB container* for executing EJB objects. These application servers can be *clones* of each other; that is, they consist of the same EJB objects and applications. For applications, all clones can process the same URL request. Alternatively, the application servers can contain unrelated sets of applications and, for applications, they can process different sets of URLs.

Application servers that are clones of each other can be limited to one machine (vertical cloning), or be spread across multiple machines (horizontal cloning). Both types of cloning provide two benefits:

- Improved throughput, because requests can be distributed across multiple application servers
- Better fault tolerance, because when one application server fails a user still can request that an application be processed by any of the remaining clones that can handle the request.

Vertical cloning is also useful when configured on a machine that is powerful enough that a single application server cannot effectively use all of its processing power (such as on a multiprocessor machine).

WebSphere provides load balancing mechanisms for forwarding a client's requests across multiple cloned application servers to distribute the load on individual application servers. For more information, refer to the WebSphere InfoCenter documentation.

Load balancing options for the Host Publisher application server

Because Host Publisher Integration Objects can execute in any WebSphere application server on which Host Publisher Server is running, they exploit all the benefits of WebSphere load balancing and Workload Management (WLM). For WebSphere Advanced Edition (AE), the Host Publisher installation process configures a separate WebSphere Application Server (named HostPubServer) in the machine where Host Publisher is installed.

Because WebSphere enables you to run multiple application servers on a machine, you can create several variations of this basic configuration. You can:

- Vertically clone the basic HostPubServer application server within a machine.
- In addition to vertically cloning the HostPubServer application server, add horizontal cloning across machines for scalability and fault tolerance.
- As an alternative to cloning, partition the application servers in a noncloned setup so that they process different URLs representing different Host Publisher applications. This prevents applications deployed on one application server from being affected by application-related problems on another application server.

  For example, if the two application servers are called Server1 and Server2, the Host Publisher application in Server1 can be configured to respond to the URL *.../application1*, and the hostpublisher application in Server2 can be configured to respond to the URL *.../application2*. This provides a static form of load balancing based on the request pattern. It also protects each application server from application-related problems in the other.

Special considerations for running chained Integration Objects in a cloned configuration are covered in the next section.

Running chained Integration Objects in cloned application servers

Host Publisher applications consisting of chained Integration Objects that are being driven by servlets and JSP pages depend on a WebSphere feature called HTTP session affinity.

When a URL for a servlet or JSP is received by the application server and forwarded to the WebSphere *plug-in*, the plug-in normally forwards the request to any of the cloned application servers that have been configured to process that URL. However, when WebSphere's HTTP session affinity feature is enabled (it is enabled by default), and the browser has included a *session ID* in the HTTP request, the normal load-balancing behavior of the plug-in is altered. A session ID in an HTTP request is either present in a cookie or included in the URL in an encoded form if URL rewriting is used.

To enforce HTTP session affinity, the plug-in maps the session ID to one of the clones in such a way that a given session ID is always mapped to the same clone. Whenever a browser includes a given session ID on its HTTP request, the request is routed to the same clone. Thus, WebSphere's session affinity feature creates an

affinity between a browser with a given session ID and a given application server among the clones that can handle that request.

A set of chained Integration Objects share a connection to a host application. This connection is internally represented as a Java object that is valid only in the application server in which it is created. Because this object cannot be serialized, it cannot be written to a file or database by Host Publisher in one application server and then recreated and used in another. Therefore, execution of all the Integration Objects in a chained application must occur in the same application server. You can assure this by enabling WebSphere's HTTP session affinity feature.

Chained Integration Objects executing in a Web container use the *HTTP session object* (a standard, servlet API-defined object used to track a given browser across HTTP requests) to correlate a browser with the host connection being used by the chained Integration Objects. If an HTTP session object is not already present, the first Integration Object in the chain creates it. This ensures that WebSphere creates a session ID and returns it to the browser along with the HTTP response. The browser returns this session ID on subsequent requests to execute Integration Objects, and the session ID is used by WebSphere to direct those requests to the application server that contains the host connection object for that Integration Object chain.

When a set of chained Integration Objects are executed in an EJB container using an instance of the Host Publisher stateful session EJB, and the EJB Access Beans themselves are executing in a Web container on behalf of a browser, then HTTP session affinity does not have to be configured in that Web container even if the EJB Access Beans execute in cloned application servers. This is because whenever a request to execute an Integration Object is sent by an EJB Access Bean to the Host Publisher stateful EJB, the request includes the handle of that EJB instance. This unique handle guarantees that the EJB protocols direct the request to execute the EJB to the application server running that EJB instance. Therefore, all chained Integration Objects executed by the Host Publisher EJB instance on behalf of a browser execute in the same application server.

EJB Access Beans executing in Web containers use the HTTP session object associated with a browser to store and track the handle of the EJB instance being used to run Integration Objects for that browser. Therefore, in this configuration, *HTTP session persistence* must be configured if HTTP session affinity is not enabled. HTTP session persistence is a WebSphere feature that enables HTTP session objects to be serialized to a database by one application server and recreated from that database in another. This feature enables all EJB Access Beans running in cloned application servers to access the EJB handle associated with a browser running chained Integration Objects.

To enforce HTTP session affinity in Host Publisher Version 3.5 and earlier releases, it was necessary to configure the HTTP session persistence feature in WebSphere. This is no longer required in Host Publisher Version 4.0.

Working with connection pools for applications in a cloned environment

When you use WebSphere's cloning capabilities to create multiple instances of Host Publisher Server that share the same application files, bear in mind that each application server running Host Publisher Server enforces connection pooling restrictions only within the scope of that application server. For example, if an

Integration Object uses a host connection pool with a maximum of 100 connections, but there are three clones running the same application, then there could be a maximum of 300 connections.

Cloning and user lists

When you create multiple instances of WebSphere application servers running Host Publisher Server and sharing the same application files, you can use user lists on multiple-logon hosts without any special considerations.

However, there are special considerations for the use of user lists with single-logon hosts:

- In *vertical cloning*, where application-related files are not physically copied to each application server but are merely represented in memory, you cannot use user lists. This restriction exists because every application server running Host Publisher Server would need a user list for its exclusive use; but in fact there is only one user list.
- In *horizontal cloning*, where multiple copies of an application are running in separate application servers, you must modify the user list in each copy of the application so that no two copies have a user ID in common.

For more information about user lists, see "Defining user lists" on page 26.

Express logon

Express logon enables a user with a Web browser certificate to log on to a host system through a Web browser without having to enter the user ID and password. This function is designed to reduce the time spent by an administrator maintaining host user IDs and passwords. It also is designed to reduce the number of user IDs and passwords that users have to remember.

In Host Publisher, express logon allows a connect macro to log on to a host application without the browser user having to enter the user ID and password. All interaction with the host application is performed by macros executing on the server. Host Publisher uses express logon to log on to host applications using client certificates obtained from a browser.

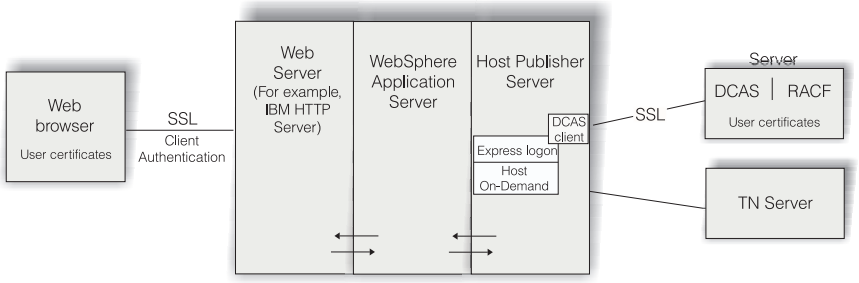


Figure 7. Express logon

The express logon architecture describes the following key components:

- A Digital Certificate Authentication Server (DCAS) component, provided by z/OS, that communicates with an enhanced version of the host-based Resource Access Control Facility (RACF) for user ID and password administration. DCAS dynamically supplies a user ID and a one-use-only passticket based on an X.509 certificate and the ID of the host application to which you want to log on.

- A DCAS client that communicates with the DCAS server over a client-authenticated SSL connection, performing a proprietary protocol to supply a user ID and passticket.

Host Publisher provides a Java-based DCAS client. To use Host Publisher's express logon capability, Integration Objects must be executed by JSPs or servlets, and the Host Publisher Integration Objects must be enabled for express logon in the Host Access application. The following things happen when the application containing the express logon-enabled Integration Objects executes on Host Publisher Server.

- When a Host On-Demand macro is executed to a point where you specified to insert a user ID or password using Host Access, Host Publisher Server uses express logon to provide the macro with a user ID or password.
- Host Publisher Server accesses the Web browser's client certificate using a servlet API and retrieves the application ID from the macro you recorded in the Host Access application. You defined the application ID in Host Access when you chose to insert a user ID or password in your macro. Host Publisher's DCAS client passes the certificate and application ID to the DCAS server, and the DCAS server provides the user ID and password requested by the macro.

Note: The browser must use HTTPS, and you must configure the application server to require client authentication.

The connection between the DCAS client and the DCAS server must be a client-authenticated SSL connection. Therefore, to execute express logon-enabled applications in Host Publisher Server, you must use IBM Key Management to create a key ring database, a password–protected Java class file that stores the X.509 certificate used for the client authentication of this connection. (See "IBM Key Management" on page 84.) You must name the keyring database HostPubELF.class and save it in <*WebSphere_install_dir*>\lib\ext. On non-Windows server platforms, you must create HostPubELF.class in Host Publisher Studio and transfer it to the WebSphere \lib\ext directory.

The following host applications support express logon: TSO, CICS, IMS, and NetView. Any 3270 application using RACF for logon validation is a candidate for express logon.

Express logon considerations when using Host Publisher Studio

If you enable express logon in the Host Access application, when you choose to insert a user ID or password, you must define an application ID and a user ID or password. The application ID is used when you run your Integration Object on Host Publisher Server. It is not used when recording or playing your macro in Host Publisher Studio. The user ID and password are only for recording the macro in Host Access. When you play the macro in Host Access, you must supply the user ID and password. When you run the macro on Host Publisher Server, the user ID and password are retrieved from DCAS.

Configuring express logon in Host Publisher Server

The following sections describe considerations for configuring express logon in Host Publisher Server.

Web browsers and application servers

Your application server requires client authentication from your Web browser. Refer to your Web browser's documentation for information on how to set up

certificates for your Web browser. Refer to your application server's documentation to set up client authentication over the SSL session.

IBM Key Management

IBM Key Management is a tool you can use to manage your digital certificates. With IBM Key Management, you can create a new key ring database or a test digital certificate, add Certificate Authority (CA) roots to your database, copy certificates from one database to another, request and receive a digital certificate from a CA, set default keys, and change passwords.

To start the IBM Key Management utility on Windows, click **Start** and then:

- In Host Publisher Studio, click **Programs > Host Publisher Studio> Certificate Management**
- In Host Publisher Server, click **Programs > Host Publisher Server> Utilities > Certificate Management**

To start the utility on a Host Publisher Server running AIX, invoke from the command line:

```
./usr/lpp/HostPublisher/common/IKeyman/IKeyman.sh
```

To start the utility on a Host Publisher Server running Solaris, invoke from the command line:

```
./opt/HostPublisher/common/IKeyman/IKeyman.sh
```

For Host Publisher Server on AS/400 , use the Key Management Utility from Host Publisher Studio and transfer the Express Logon key database file (HostPubELF.class) to the server with binary FTP.

DCAS server

For more information on how to configure the DCAS server for express logon, refer to the *Setting Up and Using the IBM Express Logon Feature*h white paper at http://www-4.ibm.com/software/network/library/whitepapers/elf.html. Refer to Part 2: Configuring the Express Logon Feature (ELF).

RACF

You must register all Web browser client certificates with RACF. This associates the certificates, which are passed by Host Publisher Server to the DCAS server, with the IDs of users attempting to log on. For more information on RACF commands, refer to *OS/390 SecureWay Security Server RACF Security Administrator's Guide* and *OS/390 SecureWay Security Server RACF Command Language Reference*.

WebSphere

For information on how to configure WebSphere for HTTPS access, refer to the WebSphere documentation.

Configuring security

The following sections describe considerations for configuring security in Host Publisher.

Configuring and using Secure Sockets Layer (SSL) support for host application access

The Host Access component of Host Publisher Studio uses Host On-Demand to provide connection support to 3270, 5250, and VT applications using Telnet protocols. Host Publisher uses the SSL support provided by Host On-Demand for

securing these connections. Using a secure connection over SSL encrypts data flowing over the connection and thus protects it against observation by a third party.

For a connection to be secured, both Host Publisher and the Telnet server it is connected to must support SSL. To secure the connection, the Telnet server must provide a certificate, which is used to encrypt the data. This certificate uniquely identifies a machine on one end of the connection. No other server in the network should have the same certificate. When the server provides this certificate, it is called *server authentication*.

If, while defining your Host Access Integration Object's connection to the host, you configure SSL support as well as server authentication, you are indicating that you require the Telnet server to provide the certificate with which to encrypt the data. If you select the server authentication option when configuring SSL, in addition to verifying the certificate itself, Host Publisher performs a check to ensure the server's TCP/IP address matches the one specified in the certificate. The server's address must be a part of the certificate for this option to work.

Host Publisher verifies that the certificate is signed by a well-known certificate authority. Host Publisher's well-known certificate authorities are Thawte, Verisign, and RSA.

If the certificate is not signed by a well-known trusted certificate authority, Host Publisher is required to have its own version of the server's certificate for verification purposes.

To enable server authentication in Host Access, you must build a .class file called CustomizedCAs.class. You must have a copy of this file on the Studio and on the server. On Host Publisher Studio, copy this file into the Studio subdirectory (c:\hostpub\Studio, for example). For WebSphere, copy the file to the <*WebSphere_install_dir*>\lib\ext directory.

The gencert.bat tool, part of Host Publisher Studio, is used to generate a certificate file for SSL enablement. The gencert.bat batch file, when used to generate a new certificate file, requests that you enter a password. To generate a proper certificate file, leave the password blank.

To use this utility, type the following command:

```
c:\install_dir\Studio\gencert.bat certificate_file
```

where *c:\install_dir* is the directory where you installed Host Publisher and *certificate_file* is the name of the self-signed certificate file from the server.

If the Telnet server has a *self-signed certificate*, the administrator of the server generated the certificate based on his or her own information without using a certificate authority. In that case, Host Publisher must have a copy of the self-signed certificate to secure the connection; however, if the server has a certificate signed by a well-known certificate authority, the certificate is guaranteed to be unique and secure, and Host Publisher is not required to have a copy of the certificate.

Using Host Publisher with forms-based security and SSL

There are two different security options for Host Publisher: forms-based security and Secure Sockets Layer (SSL) file transfer. You can select one, both, or none.

Forms-based security is activated when you select **Enable Security** from the WebSphere Administrative Console's Security Center. Forms-based security prevents unauthorized access to Host Publisher Server Administration.

SSL is activated when you enable SSL support within your Web server. Using SSL alone does not safeguard Host Publisher Server Administration from unauthorized access. However, SSL does prevent the monitoring of data passed between a client (running a Web browser) and a server.

The highest level of security is obtained by enabling both forms-based security and SSL. If both types of security are enabled and used, users trying to access Host Publisher Server Administration are prompted for an ID and password. The password, along with all data sent between the client and the server, is sent in encrypted form.

Activating forms-based security

To activate forms-based security, select **Enable Security** from the WebSphere Administrative Console's Security Center.

To grant or deny user access to Host Publisher Server Administration, use the **Role Mapping** tab in the Security Center. By default, all authenticated users are authorized to use Host Publisher Server Administration.

Activating SSL

To activate SSL, refer to the documentation for your Web server. After activation, it is a good idea to verify that an additional Host Alias entry for the SSL port (usually *:443) has been added.

Chapter 6. Using the XML Gateway to enable simplified access to host applications

The XML Gateway provides an HTML emulator for end-user access to 3270 and 5250 applications, and enables you to write a Java server program to access 3270 and 5250 application data in an XML format. (Refer to the *IBM WebSphere Host Publisher Programmer's Guide and Reference* for information on how to program Java beans and servlets related to the XML Gateway.) The XML Gateway supplies a portal to the emulator function, which relies on the XML Gateway servlet for interaction with the host applications.

The XML Gateway portal consists of the following:

- **hPubPortal servlet** — the portal servlet through which you can access the XML Gateway, which consists of host links created through Host Publisher Server Administration
- **XML Gateway servlet** (xmlGateway)—a Web-based terminal emulator that enables interaction with host terminal applications using XML data formats

Accessing the Host Publisher XML Gateway portal page

As an end user, you can use the hPubPortal servlet to access the Host Publisher portal page at this URL: http://*servername*/HPXGW/hPubPortal

where *servername* is the name of the Web server where Host Publisher is installed.

The portal page consists of host links stored in the hPubPortalData.xml file. These host links use defined parameters to access the XML Gateway servlet, and enable you to use the XML Gateway servlet to access a desired host session.

Interacting with the host application

Using the XML Gateway servlet to interact with the host application is similar to using a standard terminal emulator, except data input is performed using a Web browser as follows:

- The Tab key moves among entry fields
- Normal keyboard input is accepted in the entry fields
- Buttons that perform terminal function key commands (PF1, PF2, Clear, Enter, and so forth) are present at the bottom of the page.

The XML Gateway servlet provides two additional buttons: **Refresh** and **Disconnect**.

Refresh

Updates the browser page to reflect the latest state of the host application. This action is necessary because, unlike a traditional emulator, the XML Gateway servlet does not continuously update the host screen. The servlet updates the screen only when input is sent to the host application. Typically, this occurs when you click a function key button, such as Enter. Refresh enables you to receive an update of the host screen without first having to issue a command or type data.

Disconnect
Signals to the XML Gateway servlet that you are finished with the host application and the host connection. Resources used by this instance of the servlet are then freed, enabling more efficient access to the resources by other users. If you do not click Disconnect when you are finished, the host session is not freed until WebSphere Application Server determines that the Web session is no longer accessing the servlet. This timeout value is a configurable WebSphere Application Server parameter; it defaults to 30 minutes.

Note: We recommend that you run only one XML Gateway session from each browser window, and that each browser window be dedicated to the XML Gateway session.

Configuring time delays for XML Gateway

The XML Gateway uses specified time delays when interacting with a host application. These time delays control when the screen is read to display the latest host screen to the user. To modify or view the values for these delays, you must use XML Gateway Administration. You cannot modify or view them using Host Publisher Server Administration.

There are two delays:

Start Delay
Controls when to read the initial screen while accessing the host. The Start delay default is 2 seconds.

Interval Delay
Recognizes when a screen is no longer changing and is therefore ready to display to the user. The Interval delay default is 2 seconds.

To override the defaults for these delays, edit the hPubPortalData.xml configuration file created by Host Publisher Server Administration. This file is located in the Host Publisher Server installation directory under Server\hPubPortalData.xml.

Edit this file with a text editor that is compliant with UTF-8. Each record in the file describes one session as defined with the XML Gateway Administrative servlet. Specify values for DELAY_START and DELAY_INTERVAL to override the defaults. Specify the values in milliseconds; for example, a value of 1000 is 1 second.

Enhancing the XML Gateway sample servlet

In addition to being a Web-based terminal emulator, the XML Gateway sample servlet also enables Host Publisher to interact with host applications using XML data formats. The servlet can:

- Process host screens as XML data
- Combine the host screen XML data with XML data from other applications
- Present data to an end user in a traditional Web browser format
- Receive user input through the browser
- Use XML techniques to process the browser data
- Use an XML interface to send data back to the host application.

Refer to the *IBM WebSphere Host Publisher Programmer's Guide and Reference* for more information on how to use the Host Publisher XML Gateway interface to write applications and servlets.

Tracing the XML Gateway servlet

To see the trace file, in Host Publisher Server Administration, select **Problem Determination > View Trace**. Host Publisher Server Administration regards the XML Gateway servlet as an Integration Object. The procedures for tracing and error handling for the servlet are as follows:

1. Open Host Publisher Server Administration.
2. Open Server Tracing.
3. Check the **Integration Object tracing** box.
4. Click **Save**.
5. Enter `xmlL*` to trace the XML Gateway Integration Objects.
6. Click **Save**.

For more information about tracing and error handling in Host Publisher Server Administration, refer to "Administering problem determination components" on page 60.

Chapter 7. Using accessibility functions in Host Publisher Studio

Host Publisher Studio provides certain accessibility functions. This chapter describes how you can use the keyboard instead of the mouse to perform tasks in Host Publisher Studio.

Performing basic keyboard tasks

This section describes how to use the keyboard to navigate through the Host Publisher graphical user interface.

Using the menu bar

To navigate the menu bar and select items on it:

1. Use a mnemonic (such as Alt+F for **File**), or press F10 to activate the menu bar and then use the left and right arrow keys to move along the menu bar.
2. When you reach the desired menu, use the up and down arrow keys to select items in the menu.
3. Enter a mnemonic (such as Ctrl+N for **New** in the **File** menu), or press Spacebar to select the highlighted item in the menu.
4. To leave the menu bar, press Esc.

Using a drop-down list box

To make a selection in a drop-down list box:

1. Move the focus to the list box, using the Tab key.
2. Press F4 to open the list box.
3. Use the up and down arrow keys to select an item in the list.
4. Press F4 or Esc to close the list box.

Using a tabbed pane

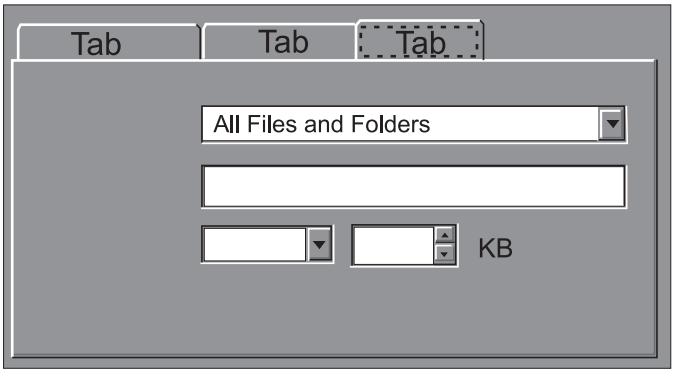


Figure 8. Using a tabbed pane

To make selections and enter data in a pane with tabbed sections:

1. Use the left and right arrow keys to move focus to the desired tab.
2. Use the Tab key to move focus among the controls for the selected tab.

a. When focus is on a text field, type text into the field.
b. When focus is on a button, press Enter to select the button.

3. Use Shift+Tab to return focus to the current tab, then use the left and right arrow keys to move focus to a different tab.

Using the terminal pane in Host Access

The following list explains the meanings of the different colored borders of the terminal pane.

yellow
The terminal pane has focus and any keystrokes are sent to the host.

gray The terminal pane has focus, but keystrokes are not sent to the host. The terminal allows special key combinations, but if you enter any other keystrokes the border flashes red and you hear a beep.

red You have entered a keystroke that will not be sent to the host.

blue Focus is not on the terminal pane.

Use the following keyboard combinations when the terminal pane has focus:

Ctrl+Tab
Move focus away from the terminal pane to the next control in sequence.

Ctrl+Shift+Tab
Move focus away from the terminal pane to the previous control in sequence.

Shift+Arrow keys
Draw a box at the cursor position. Use Shift+Arrow keys to extend the box up, down, to the right, and to the left.

Ctrl+Arrow keys
Move a box.

Using the keypad in Host Access

When the keypad is displayed, it immediately follows the terminal pane in the focus sequence. It is visually displayed to the right of the terminal pane. When focus is on the terminal pane, press Ctrl+Tab to move focus to the first button of the keypad.

To move focus on the keypad, press Tab to move forward and Shift+Tab to move backward. To move from one column to the next, you must tab through all the buttons in the column.

To send a key to the host, move the focus to that key and press the Spacebar. Focus moves either to the wizard pane (if the terminal pane was updated as a result of the key action) or to the terminal pane (if no update occurred).

Note: When the keypad has focus, you cannot activate the menu bar using mnemonics or using F10. You must first move focus off the keypad by tabbing (either backward or forward) until focus is no longer on the keypad.

Using keyboard remap in Host Access

The dialog for Load Keyboard Settings is a file chooser with a drop-down list box for the directory selection and a scroll pane for file selection. You can move from button to button, and into the scroll pane where the files are listed, with the Tab key. When the selected button has the focus, execute the action for that button by pressing either Enter or the Spacebar. When the focus is in the scroll window, you can move the cursor to select the file you want to load.

To edit keyboard settings, do the following:

1. Press Alt+T to display the choices in the **Terminal** menu.
2. Press the down arrow key so that **Keyboard** is highlighted.
3. Press the right arrow key to expand the list of **Keyboard** selections.
4. Press the Spacebar to select **Edit Keyboard Settings**.

You can move from button to button and field to field with the Tab key. When a button has focus, execute the action for that button by pressing the Spacebar. The dialog for **Edit Keyboard Settings** requires pressing the Spacebar to execute the action for the button that has focus.

Using help

In Host Publisher Studio, you can use the keyboard to navigate through the help window. Use the Ctrl+Tab keys to move from left to right along the toolbar and through the two panes. (Use Ctrl+Shift+Tab to move in the opposite direction.)

The left pane of help is a tabbed pane. It can be navigated as described in “Using a tabbed pane” on page 91.

Use Alt+F4 to close the help window.

Chapter 8. Performance and tuning

This chapter summarizes system requirements and recommendations for improving the performance of Host Publisher.

Host Publisher Studio requirements

Host Publisher Studio requires:

- Intel platforms that support Microsoft Windows 98, Windows ME, Windows NT, Windows 2000, and Windows XP
- Pentium™ processor with a speed of 366 MHz or greater
- Minimum of 256 MB memory
- Minimum of 120 MB free disk space

Host Publisher Studio is a Java application and therefore makes heavy demands on the processor. If you are developing multiple complex applications, consider a more powerful processor.

Host Publisher Server requirements

Host Publisher Server is supported on Intel, pSeries, Sun SPARC, and iSeries hardware platforms, and performs best on machines that are designed to operate as servers. These machines are typically built for improved scalability and performance. Server machines usually have the capacity for large amounts of memory and offer large Level 1 (L1) and Level 2 (L2) caches.

The four major hardware components that most affect the capacity and performance of Host Publisher Server are:

- Central processing unit
- Memory
- Network interface card
- Hard drive

Central processing unit (CPU)

Host Publisher runs as a plug-in to WebSphere Application Server and uses Host On-Demand to manage connections; Host Publisher, WebSphere Application Server, and Host On-Demand are all Java products. Typically, Java products perform best on fast or multiple processors. We recommend that Host Publisher run on processors that are designed to operate as servers. These processors operate at Java-compatible speed and usually have built-in technologies, such as a processor cache, that support greater capacity and performance.

Most designed-for-server processors contain two levels of cache: L1 and L2. These caches act as temporary storage spaces for instructions and data obtained from slower memory.

For Intel-based servers, Pentium II or greater processors with speeds of 450 MHz or greater are recommended. Xeon processors provide enhanced performance over non-Xeon processors.

For iSeries servers, use a model that is recommended for Java applications. The *AS/400 V4R5 Performance Capabilities Reference* (found at http://ca-web.rchland.ibm.com/perform/perfguideup/V4R5perfguide/V4R5perfguide.pdf) lists models that are recommended for use with Java applications. For performance reasons, iSeries models with a processor commercial processing workload (CPW) rating of 460 or more are recommended. Models with lower processor CPW ratings might function, but you might not be satisfied with your server performance. Refer to the most recent *AS/400 Performance Capabilities Reference* manual for the latest Java-recommended servers and their processor CPW ratings.

The iSeries models in the Table 2 are recommended; however, all recently-announced servers might not be included here.

Table 2. iSeries servers

iSeries Model	iSeries Feature	Processor CPW Rating
820	2398	3200
	2397	2000
	2396	950
740	2070	4550
	2069	3660
730	2068	2890
	2067	2000
	2066	1050
	2065	560
720	2064	1600
	2063	810
270	2252	950
	2253	2000
170	2388	1090
	2386	460
	2385	460
650	2243	2340
	2240	1794
640	2239	998.6
	2238	583.3
530	2162	509.9
	2153	459.3
S40	2261	2340
	2256	1794
S30	2260	1794
	2259	998.6
	2258	583.3
S20	2166	759
53S	2157	509.9
	2156	459.3

Memory

Another essential hardware resource is an adequate amount of physical memory; you must have enough to avoid memory depletion and excessive disk input/output.

By design, Java applications do not return their unused storage for reuse. The application server's garbage collection facility runs only occasionally to claim unused storage. To avoid memory depletion between garbage collections, large amounts of memory are therefore required. In addition, you might want to tune your server for performance, which almost always affects memory allocation. For these reasons, you cannot make changes to tuning parameters unless sufficient physical memory exists on the system.

We recommend that you allocate 512 MB of memory for Host Publisher running on uniprocessor machines, and 1 GB of memory for multiprocessor machines. These recommendations are in addition to what is required to run your other applications.

Network interface card

The network interface card (NIC) can be an important factor in the capacity and performance of your server. The NIC moves data between the system data bus and the network media. Because the system data bus can operate at much higher speeds than the network media, the NIC can become a performance bottleneck. If the NIC is too slow, your server throughput and capacity will be based mainly on how fast the NIC can move data to the network media.

Choose a high-performance NIC that is designed for network-intensive applications. Such a NIC will implement one or more of these features:

- Bus mastering
- RAM buffering
- Direct memory access (DMA)
- Shared memory
- Onboard microprocessor

In addition, you might consider multiple NICs if you find that you saturate a single NIC and if your server has the capability to handle more workload.

Hard drive

Your server must contain enough hard drive space to accommodate an operating system, a Java runtime environment (JRE), an application server, the WebSphere Application Server product, the Host Publisher product, and any other products you plan to install. Your total disk space requirements should include extra space for configuration files, product upgrades, user application files, and server operational files, such as error logs.

Actual disk space requirements are platform dependent. Host Publisher Server requires 200 MB of hard drive space. We recommend that you allocate at least 400 MB of hard drive space for Host Publisher Server, its operational files, and its user applications.

Hardware recommendations

Table 3 depicts system hardware recommendations for each platform operating as a standalone Host Publisher server.

Machines with less power or fewer resources might work, but you might not reach your response time objectives and user capacity goals. Machines that are running other applications might require more resources, depending upon the resource consumption of those applications.

Table 3. Host Publisher Server hardware recommendations

Platform	Processor Type	Hard Disk Space	Memory
Intel	Pentium 350 MHz or greater	2 GB	2 GB
RS/6000	PowerPC 375 MHz or greater for uniprocessor machines, PowerPC 332 MHz or greater for multiprocessor machines	2 GB	2 GB
Sun Sparc	UltraSPARC 333 MHz or greater	2 GB	2 GB
iSeries	Java-compatible with CPU rating of 460 or greater	2 GB	2 GB

Server capacity

Many factors determine the number of users that your Host Publisher Server will support. These factors include, but are not limited to:

- The platform: computer hardware, operating system, and networking software
- The application server: IBM HTTP Server, Microsoft Internet Information Services, and so forth
- The type of information served: 3270, 5250, database, VT, and so forth
- The application design:
  - Connections with pooling enabled or disabled
  - Amount of interactions required between the server and the data source to obtain the requested information
  - Other similar factors
- Client usage patterns; for example, how often will users access the server?
- System performance objectives; for example, what are the response time requirements and what are the server availability requirements?
- Server workload from other applications

It is, therefore, not possible to provide capacities that are accurate in every system environment for all applications. For accurate capacities for your specific server, measure your specific applications in your operational environment.

When you estimate server capacity, remember to plan for growth and surges in user activity.

Chapter 9. Troubleshooting

This chapter helps you identify problems and determine solutions with Host Publisher. If the solution is not documented, you are directed to gather the necessary information for IBM service.

Host Publisher problem determination procedure

Follow this procedure to resolve a problem found with Host Publisher:

1. Note whether the error occurred in Host Publisher Studio or Host Publisher Server, the task being performed when the error occurred, and the symptoms of the error.
2. If the error was caused by a memory shortage for the Java virtual machine, close some applications to free memory and attempt to perform the task again.
3. See "Common problems and limitations" on page 101. If the symptoms of your problem are listed, follow the recommended actions to resolve the problem.
4. Refer to the Host Publisher support page at http://www.ibm.com/software/webservers/hostpublisher/support.html. This page provides links to technical information such as Host Publisher News, Service Updates, Hints and Tips, and the Library.

   You can go directly to Hints and Tips. Hints and Tips are brief instructions explaining technical issues that were discovered too late to be included in the documentation. Hints and Tips cover installation, configuration, troubleshooting, and usage issues. Hints and Tips are dynamic; they are updated every few weeks, while the Readme is updated only when a CSD is made available.
5. If you are running Host Publisher Studio:
   - Error output for Host Access and Application Integrator is automatically routed to the error logs, HostAccess.con and Studio.con, respectively, which are located in the *install_dir*\Studio directory. Check these files for error information.
   - To route error output for Database Access to a console window:
     a. Copy the file dbaccess.bat, which is located in the *install_dir*\Studio directory, to a new .bat file.
     b. Edit the new .bat file and make the following changes:
        – Remove `start /B` from the beginning of the file.
        – In the invocation section of the file, change `javaw` to `java`.
     c. Execute the new .bat file from the command line.
   - Tracing often provides helpful information for diagnosing problems. See "Enabling tracing" on page 49 for information about enabling tracing in the Host Publisher Studio.
   - Additional error output might be located in the messages_HostPubStudio_0822_x.txt file located in the *install_dir*\Studio directory.
6. If you are executing your Host Publisher application on the server, check the Host Publisher logs for error messages that indicate the cause of the problem.

   If the error log indicates an internal error and that service should be contacted, first try to recreate the problem. Turn on all Host Publisher Server trace

options, as well as tracing for the Integration Object being run. Clear the current Integration Object trace files, then recreate the problem.

Note: Do not turn on Host Connection tracing unless IBM service tells you to do so.

If the problem appears when starting WebSphere Application Server or Host Publisher Server, or when accessing the first page of your Host Publisher application, use the problem determination procedures for WebSphere Application Server. For information on how to obtain problem determination data from WebSphere Application Server, access the WebSphere Application Server Web page at http://www.ibm.com/software/webservers.

7. Contact IBM service to resolve the problem. When you report a problem, be ready to provide information such as product version, current *Authorized Program Analysis Report (APAR)* levels, and so forth. You can obtain this information by issuing the **report** command (or, in Host Publisher Server Administration, by opening the Version Information window). The **report** command is located in the *install_dir*\Common\service directory; see "Updating Host Publisher using the Software Maintenance Utility" on page 115 for more information.

Host Publisher Server Administration Troubleshooting

The following sections contain suggestions for actions you can take if you have difficulty using Host Publisher Server Administration.

Server prerequisites and general information

To begin troubleshooting, ensure that the following Host Publisher Server Administration requirements are met. For more information, refer to the *IBM WebSphere Host Publisher Planning and Installation Guide* for your platform.

- Is a supported application server installed?
- Is the correct version of WebSphere Application Server installed?
- Is the locale set correctly on the server on which Host Publisher Server has been installed?

In addition:

- Examine the application server, WebSphere Application Server, and Host Publisher log files for information relating to your problem.
- Flush the cache in your Web browser, close all instances of the browser, and restart.

WebSphere Application Server

To troubleshoot on the application server:

- Make sure that WebSphere is started.
- Make sure that Host Publisher Server is started
  1. Start the WebSphere Administrative Console.
  2. Click **WebSphere Administrative Domain**.
  3. Click your host name.
  4. Make sure that Host Publisher Server (shown as *HostPubServer*) is listed as Running.

Secure Sockets Layer (HTTP server)

Make sure SSL is configured correctly on the server.

- Are certificates provided?
- Is the SSL port enabled in the application server configuration file? Refer to your application server documentation for information on enabling the SSL port.
- Is the SSL port configured in the alias list in WebSphere Application Server?

WebSphere pagecompile

If you receive a message saying that main.jsp is not found, it is possible that Host Publisher Server has been uninstalled and reinstalled on different drives.

WebSphere Application Server compiles JavaServer Pages (JSPs) into Java servlets, then invokes those servlets to render the actual page to a browser. This Java code remembers the exact location of the original page (for example, d:\WebSphere\AppServer\installedApps\HPAdmin.ear\HPAdmin.war\main.jsp) so that it can reproduce its HTML content. The servlet is rebuilt from the original JSP only if the page is changed (date stamp is updated). If the location of the JSP changes, but its date stamp does not, you receive an internal error. WebSphere reports the error after trying to process the JSP because it can no longer find the original file. This can happen if you reinstall the same version of Host Publisher Server in a different location.

To correct this problem, remove WebSphere's record of the JSP. To do this, remove the corresponding Java and class files from the WebSphere installation directory; for example:

```
d:\WebSphere\AppServer\temp\host_name\hostpubserver\
\HPAdmin.ear\HPAdmin.war\main_jsp*.*
```

Common problems and limitations

The following sections document common problems and limitations you may encounter while using Host Publisher. For each symptom, the probable cause and suggested action is provided.

Problems with Host Access and execution of Host Access Integration Objects

You might be able to avoid some common pitfalls by having both of the **Display Warning Messages** options enabled while you are working in Host Access. This ensures that you see confirmation messages and validation messages as you develop you applications. For more information about this option and how to set it, see "Display Warning messages" on page 20.

Failures creating Integration Objects

When creating an Integration Object, Host Publisher builds Java source code and assembles it into an executable file. During this process, you might receive a message stating that the Integration Object could not be created. This message indicates that the compilation of the generated Java source contained errors.

To solve this problem:

- Refer to the iofailed.txt file located in the *install_dir*\Studio directory.
- If you have manually modified the host macro files, use the messages in iofailed.txt to determine whether your changes caused the error. If you have not customized any of the Host Publisher files, contact IBM and report the problem.

Macros fail in Host Access Integration Objects

Macros can fail for many reasons. Successfully running a macro depends on the application behaving as you expect. The screen flow logic of an application must not change unpredictably over time. Macros must be recorded such that they can process the content of the screens they encounter and are be precise in identifying the screens processed.

See "Defining a screen" on page 16 for some tips for defining a screen in a macro.

Macro Play error in Host Access Integration Objects

If you receive a Macro Play error, your screen definition might be the cause.

If a screen definition uses the cursor position as its recognition criterion and the screen has more than one action, you might not be able to step through the actions of this screen or start the play on an action after the first action.

In these cases, Host Access tries to start playing the macro on the given screen with the given action; however, if the first action has caused the cursor position to change, the screen no longer meets its recognition criteria.

This problem will not occur if you select **Play** instead of **Step**, and start the play on the first action of the screen or at some position in the macro tree above the first action.

New tags are deleted from a macro

You can edit macros with the macro editor in Host Access (see "Editing a macro" on page 19) or another editor that supports UTF-8 encoding. Macro syntax is defined in the *IBM WebSphere Host Publisher Programmer's Guide and Reference*. Not all tags are displayed in the Host Access macro tree, but all tags are saved in the macro and used when the macro is played.

The macro file is an XML file. XML standards apply to any data that is typed into the file. If you add data to the macro that is not recognized as XML data or that contains a tag that is not recognized as macro syntax, the data is removed from the macro with no warning when you save the macro in Host Access.

You are notified if there are other syntax errors. If you are using the macro editor, you are shown a list of the errors and you must correct them before you can save the macro. Macros containing syntax errors fail to load when Host Access is started or when an Integration Object that includes the macro is opened. An error message shows which macro failed, and details are available in the HostAccess.con file.

Private characters display incorrectly on client workstations

The private character editor in Windows 2000 and Windows XP enables an application developer to extend the font library by defining additional characters. Host Publisher supports these private characters on Windows 2000 and Windows XP if they are developed using the Unicode character set (the default) and are linked to all system fonts.

To ensure that private characters are displayed correctly for users of your Host Publisher application, you must distribute an up-to-date version of the private character set to all client workstations where the application will be used. The private character set is defined in *Windows_dir*\fonts\eudc.tte, and the copy on each workstation must reside in the Host Publisher installation directory as \Studio\jdk\jre\lib\fonts\eudc.ttf.

Screen paint problems

To reduce screen painting problems on the Windows 2000 or Windows NT desktop when you are using the frame minimize, restore, and maximize functions in Host Publisher Studio applications, make the following changes:

In Windows 2000:

1. Open the Control Panel and click **Display**.
2. Click the **Effects** tab.
3. Clear **Use transition effect for menus and tooltips**.

In Windows 2000 and Windows NT:

Do not use 256 color mode. Use a high color (16 bit) or true color (24 or 32 bit) mode. To select a color:

1. Right-click on the Windows 2000 or Windows NT desktop.
2. Click **Properties**.
3. Click the **Settings** tab.
4. Select a color from the Color Palette.

Host Access through firewall configured in nontransparent mode

When using a firewall configured in nontransparent mode, you might encounter these problems:

- You cannot draw a box around the `Wingate>` prompt, which is the only text on the screen. To identify the screen, you must refer to the cursor position.
- On the next screen, if you try to type the destination IP address (for example, 141.211.228.169), as soon as you type the 1 , Host Publisher changes to a new screen.

To avoid these problems, configure a port on the proxy server that maps to the destination address; for example, use the firewall in a transparent mode. This is similar to the way in which you would configure Host On-Demand's Redirector.

Problems with Database Access and execution of database Integration Objects

This section lists common problems associated with the Database Access application.

Database Access does not start

If Database Access does not start, there might be a conflict between your system CLASSPATH and the classes upon which Database Access depends. To fix this problem:

1. Make a note of the JDBC drivers that appear in your CLASSPATH.
2. Edit dbaccess.bat, located in *install_dir*\Studio.
3. Remove %CLASSPATH% from the -classpath statement and replace it with the JDBC drivers you use to connect to your database.

Failures creating Integration Objects

When creating an Integration Object, Host Publisher builds Java source code and assembles it into an executable file. During this process, you might receive a message stating that the Integration Object could not be created. This message indicates that the compilation of the generated Java source contained errors.

You might find useful information about the cause of the failure in the iofailed.txt file located in the *install_dir*\Studio directory.

Microsoft Access date fields do not appear correctly

When you use the JDBC/ODBC bridge to connect to a Microsoft Access database, Date columns do not appear on the Condition panel of the Database Access application. This is because the JDBC/ODBC bridge does not correctly report the column type to the Database Access application.

This is a known JDBC/ODBC bridge driver problem that Host Publisher cannot correct.

Database interface does not work with Lotus® Domino™ JDBC driver

The Lotus Domino driver for JDBC is not supported by Host Publisher. This driver is not JDBC-compliant and is missing some classes required by Host Publisher.

Java errors with Oracle database driver

When you connect to an Oracle database using the Oracle8i 8.1.6 JDBC Thin Driver, and you use a variable name for a non-character data type and then click **Run SQL**, you receive a java.lang.ClassCastException: java.lang.String message. This problem does not occur when the Integration Object is deployed to a server and an application invokes the Integration Object.

Run SQL works correctly for a variable that represents a character data type.

Connecting to an iSeries database using Windows 98

If you are running Host Publisher Database Access on Windows 98 and are unable to connect to an iSeries database, complete the following steps to change your DBAccess.bat file.

1. Edit the DBAccess.bat file in the HostPub/Studio directory.
2. Remove the double quotes at the beginning and end of the classpath value for the SET CLASSPATH statement.
3. Save the file.

If you are using Host Publisher Studio to create a database Integration Object by launching Host Publisher Database Access from within the Application Integrator, and you are unable to connect to an iSeries database, make the following changes to your webbridge.bat file.

1. Edit the webbridge.bat file in the HostPub/Studio directory.
2. Remove the double quotes at the beginning and end of the classpath value for the SET CLASSPATH statement.
3. Save the file.

DBCS User Defined Character (UDC) input in form data

If a DBCS User Defined Character (UDC) is input as text in form data, both Netscape V6 and Microsoft Internet Explorer encode the UDC incorrectly.

According to standard encoding rules (application/x-www-form-urlencoded), the browser should convert any non-alphanumeric characters into a percent (%) sign followed by the hexadecimal code of the character. Netscape V4.7 or newer correctly encodes each byte of the double-byte UDC.

For example, if you build a Host Publisher Database Access application that used an HTML form to update database information, a DBCS UDC input using

Netscape V6 or Microsoft Internet Explorer appears in your database as ?, whereas a DBCS UDC input using Netscape V4.7 or newer would appear correctly.

Note: User Defined Characters can be displayed correctly as output by both Netscape and Microsoft Internet Explorer.

Receive error: No suitable driver found

If you receive a `No suitable driver found` error, make sure you have added the JDBC drivers for your database using the WebSphere Application Server console.

Unsupported JDBC Server configuration error (CLI0621E)

You might receive this error message when you run an application containing Database Access Information Objects on the server, and the application attempts to connect to a remote DB2 database.

This problem occurs because the JDBC driver on the server is not at the same FixPack level as the JDBC driver on the Studio machine where the Integration Objects were generated. If the FixPack levels do not match, you must upgrade the JDBC driver on the Studio machine, regenerate the Integration Objects in Database Access, reassemble the application in Application Integrator, and transfer the new application to the server.

JDBC error: db2jdbc not found error appears in jvm_stderr.txt

If you receive a `db2jdbc not found` error, make sure you have added the JDBC drivers for your database using the WebSphere Application Server console.

Connect timeout in a database Integration Object has no effect

The connect timeout value is not implemented by all JDBC drivers.

The JDBC driver will time out while trying to connect to a database, based on the value for the driver. JDBC driver vendors are expected to implement the connect timeout value.

Requested data not returned to end user of a Database Access Integration Object

When you create a Database Access Integration Object, you can specify several types of conditions on the **Condition** tab. A runtime requirement exists if you create a variable for a column with a data type of character (CHAR), and you select one of the following operators:

- contain the character(s)
- start with the character(s)
- end with the character(s)

When such an Integration Object is run on Host Publisher Server, the end user **must** enter the wildcard character (%) to indicate which operator is used. After you create the Integration Object, the operators can be used interchangeably by entering the wildcard character in varying positions of the value, as shown in Table 4.

Table 4. Wildcard values used for operators at runtime

Operator	Value
contain the character(s)	%value%
start with the character(s)	value%
end with the character(s)	%value

For example, if the Integration Object was created specifying ″contain the character(s)″ as the operator, the user can change the query by specifying ″%XXX″ as the search criterion; the output shows all values that end with the characters XXX.

DB2 V6.1 fixpack 5 does not support variables

If you use DB2 Version 6.1 fixpack 5 with Database Access, you might have difficulty creating variables for SQL statements. After you click **Run SQL** in the SQL window, DB2 returns this exception:

```
COM.ibm.db2.jdbc.DB2Exception: [IBM][CLI Driver] CLI0122E Program
type out of range.  SQLSTATE=HY003
```

To avoid this problem, install fixpack 6 or newer.

Displaying the Specify Variable Value columns on double-byte systems

When you click **Run SQL** for an SQL statement containing a variable, the Specify Variable Value(s) window opens, where you can enter a value.

On a double-byte system, the Name and Type column values might not display.

To correct the problem, try one of the following:

- Click the Name or Type column value that is missing.
- Resize the Specify Variable Value(s) window.
- Cancel the Specify Variable Value(s) window, then click **Run SQL** again.

Problems with Application Integrator and transferring applications

This section lists common problems associated with the Application Integrator component of Host Publisher Studio.

Ownership of installation files on UNIX® operating systems

WebSphere Application Server on Solaris or AIX runs with the user ID and group of *nobody*, and Host Publisher runs with the same permissions. Host Publisher Server application directories must also be owned by the *nobody* user ID and group. The user ID specified when transferring applications to a server must be the *nobody* user ID.

To change the ownership of all the files shipped with Host Publisher from *nobody* to your chosen ID, type the following at a command line after installation:

1. `cd /usr`
2. `find HostPublisher -exec chown` *new_user_ID* `{} ";"`
3. `find HostPublisher -exec chgrp`*new_user_group* `{} ";"`

When you specify a Sun Solaris or AIX FTP server (**Preferences > Options**) in Host Publisher Studio, or when you configure a server in the Host Publisher Studio wizard, specify the nobody user ID to use when transferring application files to that server using FTP.

You must use this new user ID when transferring applications to this Host Publisher Server.

Application Integrator preview function does not display preview Web page

If Application Integrator does not display your preview Web page (using Netscape on a Windows system), you must apply a service pack.

- For Windows NT, apply SP5 or higher.
- For Windows 2000, apply SP1 or higher.

Shortcut errors occur when previewing a page with Netscape

Under the following circumstances, shortcut errors might occur when you use Netscape to preview a page in Application Integrator.

- **If you do not have Netscape open** and you try to preview a page, you receive an error message stating that the shortcut was not found; however, the page loads.
- **If you already have Netscape open** and you try to preview a page, nothing displays.

To avoid this problem:

1. Click **My Computer** on your desktop.
2. From the menu bar, click **View > Options**. (In Windows 2000, click **Tools > Folder Options**.)
3. Click the **File Types** tab.
4. In the **Registered file types** box, select Netscape Hypertext Document.
5. Click **Edit**. (In Windows 2000, click **Advanced**.)
6. In the **Actions** list box, select Open.
7. Click **Edit**.
8. In the **Use DDE** box, make sure the Application is NSShell (not Netscape).
9. Click **OK**, then click **Close** on the next two windows.

This procedure might not prevent the error message if you are running Netscape Version 6. You can prevent the error message by clearing the Use DDE check box.

Host Publisher experiences a Java page violation on startup

If you are running a keyboard translation application called IME, it is possible that a Java page violation will occur when you start Host Publisher.

To avoid this problem:

1. From the **Start** menu, click **Windows Update** to open your browser to the Microsoft Web site that provides updates to the Windows platform and IME updates.
2. Download the updates to your machine.

Unwanted characters display in a VT terminal window

If you are running Host Publisher Studio on Windows, and you click the mouse inside the text display of a VT session, you might see unwanted garbage characters displayed on the screen. This usually occurs when the host is in a command-line mode. The characters usually consist of open brackets, carets, and the letters D and A.

These characters appear because Host On-Demand gives you the ability to move the screen cursor using the mouse. Unlike in 3270 or 5250 sessions, however, the connection to the VT host is asynchronous. As a result, cursor movement, rather than being a local event on the terminal, is governed by the host, not the terminal.

When you click the mouse, the Host On-Demand framework attempts to move the mouse from its previous location on the screen to the location where the click occurred. For this to happen in a VT environment, the host has to be notified through a series of simulated cursor movement keystrokes (arrow keys). If the VT host is currently running a full-screen application such as an editor, this action is seamless to the user and the cursor is simply moved from the old location to a new location. But, if the host is at the command-line prompt where there is no concept of a full screen, the host rejects the cursor movement keys and returns them to the terminal. These are the characters you see.

If this problem occurs, simply delete the unwanted characters and proceed.

Problems with the Server and execution of Integration Objects

This section lists common problems associated with the runtime environment in Host Publisher Server.

Characters in file are not read correctly

When Host Publisher Server reads a file, it assumes that the file was written with UTF-8 encoding. As a result, files written with native (non-UTF-8) encoding might be read incorrectly. Moreover, it is possible—particularly in the case of files that contain double-byte (DBCS) characters—that Host Publisher Server will not recognize the incorrect characters and will not return an error message.

The best way to avoid this problem is to ensure that all files read by Host Publisher Server are written with UTF-8 encoding. Your application files are written with UTF-8 encoding if:

- You use Host Publisher Studio to develop new applications.
- You use the Host Publisher migration tool, StudioAppMigrator, to migrate applications that were developed prior to Host Publisher Version 4.0.
- You use the AppMigrator tool to migrate applications on the server.
- You use an editor compliant with UTF-8 whenever you edit application files on the server.

Connection for chained Integration Objects in a cloned application server

When there are cloned application servers running applications containing chained Integration Objects, the HttpSession invalidation timer thread runs in the application server where the HttpSession object was created. A chained application might not be running in that application server; therefore, on HttpSession timeout, the connection associated with a chained application might not get cleaned up if the HttpSession invalidation occurs on a different application server.

The easiest solution is to modify the maxBusyTime in the .poolspec file for the connection, or use Host Access to set Maximum busy time before disconnection on the Connection Pool Configuration page.

Error page does not display for an error in the second Integration Object on a page

If a JSP page contains multiple Integration Objects and an error occurs after output is written to the HTTP output stream, the error page might display incorrectly. If enough output is written to the HTTP output stream so that the JSP output buffer fills before the error occurs, then forwarding of the browser to the error page might not work. Instead, you might see an error message like `Error 500: ERROR: Cannot forward. Writer or Stream already obtained`.

To avoid this problem in your JSP page files, ensure that an error is detected before any substantial output to the page can take place. In the following example, make sure all Integration Objects have been created and executed before any HTML code is written to the output stream. Also, explicitly check for errors and stop processing if any doHPTransaction statements fail. Otherwise, HPS5035 errors can occur when subsequent Integration Objects try to use the back-end connection.

```
<%@ page contentType="text/html;charset=UTF-8" errorPage="DefaultErrorPage.jsp">

<jsp:useBean id="Oi7" type="IntegrationObject.Oi7"
      class="IntegrationObject.Oi7" scope="request">
<jsp:setProperty name="Oi7 property="*" />
</jsp:useBean>

<jsp:useBean id="Oi8" type="IntegrationObject.Oi8"
      class="IntegrationObject.Oi8" scope="request">
<jsp:setProperty name="Oi8 property="*" />
</jsp:useBean>

<HTML>
<BODY>
<%  // Only minimal output has been written to the output buffer to this point.
  // Now perform the Integration Objects.
  Oi7.setHPubStartPoolName("Oi7");
  Oi7.doHPTransaction(request,response);

  // You can insert code here for retrieving data from the
  // previous Integration Object, but don't write any output
  // until all of the Integration Objects have executed,
  // in case a later one should fail with an error.

  // An error in the previous Integration Object will cause a
  // BeanException, and the following Integration Object
  // will not be called.

  Oi8.setHPubStartPoolName("Oi8");
  Oi8.doHPTransaction(request,response);

  // Now that the Integration Objects have completed without an
  //  exception occurring, we can begin to write output to the page.
%>
  // Insert Java code and HTML here for writing detailed output to the page.

</BODY>
</HTML>
```

You can also enclose the Integration Object calling and output page writing code in a try/catch block and handle your own errors separately from the error page:

```
<% try {
  // perform Integration Objects and write output to page (as above)
 }
%>
<% // Error handling section -- The Integration Objects will generate BeanException
  //   if they fail, or we may have another error resulting in an Exception.
  //   Insert your error handling code in the appropriate place below.  Refer
  //   to DefaultErrorPage.jsp for how the default error page handles exceptions.

 } catch (BeanException e) {
  // One of the Integration Objects failed.  Insert your
  // Integration Object error handling code and HTML here.
 } catch (Exception e) {
  // A failure in the Java code of this JSP file.  Insert your
  // error handling code and HTML here.
 }
%>
```

Multiple accesses to chained Integration Objects from the same machine (HPS5035) not performing

Netscape browsers use a single HttpSession for Web access, even when accessing from different browser windows. The multiple windows try to use the same data source connection resource, resulting in confusion for the chained Integration Objects. You might see HPS5035 errors as a result. (Refer to *IBM WebSphere Host Publisher Messages Reference* for details about message HPS5035.)

To avoid this problem, do not attempt to run multiple chained Integration Object applications concurrently from the same Netscape client machine.

When you double-click the Internet Explorer icon to open the Internet Explorer browser, this problem does not occur; however, if you use any method other than double-clicking the icon to open the browser, the problem occurs.

HPS5035 errors occur when a chained Integration Object attempts to retrieve a data source resource from the HttpSession, but the resource is not there. This might occur under the following conditions.

- A single JSP page or servlet runs multiple Integration Objects created with an earlier release of Host Publisher (Version 3.5 or earlier), without checking to see whether an error occurred in a previous Integration Object. The previous Integration Object will not have placed the back-end data resource back in the HttpSession if it encountered an error. JSP pages or servlets should not attempt to continue running subsequent chained Integration Objects in this case, but should instead allow the Integration Object in error to redirect the client to an error page.

  You can protect against this situation by using the following servlet or JSP code:

  ```
  // Run the Integration Object transaction. Does a sendRedirect() on error.
  myBean.doHPTransaction(request, response);
  if (myBean.getHPubErrorOccurred() != 0)
  {  // Do not call another Integration Object's doHPTransaction() or produce output.
     return;
     // Instead, we allow the sendRedirect() to answer the client.
  }
  ```

- A user clicks **Submit** twice or clicks a link twice before the first click can be completely processed at the server. An impatient user might click on a link to the next page of a chained Integration Object application twice rather than waiting for the next page to process its Integration Object and send the output page back to the client. The second click causes the next page to be processed again while the Integration Object still has ownership of the data source resource. During this processing, a second instance of the Integration Object cannot find the resource in the HttpSession. This causes an HPS5035 error, while the output of the first processing is lost.

  You can protect against this using the following JavaScript:

  ```
  <SCRIPT Language="Javascript">
  /*********************************************************************
  *  Prevent multiple Submit or href click invocations from this page.  *
  *  The 'submitFlag' variable is the switch.  When the function is     *
  *  called and the flag is zero, 'true' is returned to the onSubmit    *
  *  parameter of the form. Otherwise a 'false' is returned, preventing *
  *  additional submit buttons/links from being reselected.             *
  *********************************************************************/
  var submitFlag = 0;
  function chkSubForm()
  {
    if (submitFlag == 0)
  ```

```
  { submitFlag = 1;   // first time a button has been clicked
    return true;
  }
  else
  { return false;    // submit has already been clicked
  }
}
</SCRIPT>
<a    href="<%= response.encodeUrl("TheNextPage.jsp") %>"
   onClick="return chkSubForm()">Next</a>
```

- A long-running Integration Object (usually more than a minute) causes the application server to time out while Host Publisher and the Integration Object is still processing. If such a timeout occurs, the application server might destroy the HTTP connection to the browser client. Internet Explorer sometimes reacts by immediately requesting the same page again without user intervention. This second request would be just like clicking **Submit** or a link twice, causing a second instance of the Integration Object to run. This second instance fails to find the data resource in the HttpSession, which causes an HPS5035 error. In this case, too, the output of the first processing is lost.

  The simplest way to prevent this problem is to have no Integration Object run for a very long time. Alternatively, you might be able to adjust the application server timeout value.
- The maxBusyTime on a poolspec expires because a chained Integration Object application is abandoned between Integration Objects. If this occurs, the next Integration Object in the chain logs an HPS5035 error if a user continues with the application after the expiration. If you do not want the user to continue, you should either lengthen or disable the maxBusyTime timer.
- The HttpSession Activity Timeout (set in WebSphere) occurs because a chained Integration Object application is abandoned mid-application. If this occurs, the next Integration Object in the chain logs an HPS5035 error if a user continues with the application after the expiration. If you do not want the user to continue, you should either lengthen or disable the HttpSession Activity Timeout.

Server macro play error — unexpected delay or possible hang

If you experience a delay or a session hang while playing a macro in Host Publisher Server, it might be caused by the insertion of a custom action into the macro when it was recorded in Host Publisher Studio.

When you are recording a macro in the Host Access component of Host Publisher Studio, and you perform an action that causes the connection to drop (such as logoff in a 3270 session), a custom action is inserted into the macro after this action:

```
<actions>
     <input value="logoff[enter]" row="0" col="0" movecursor="true"
                  xlatehostkeys="true" encrypted="false" />
        <custom id="waitForDisconnect" args="Please DO NOT remove this tag!"  />
```

The purpose of the custom action is to wait for the event associated with the disconnect to ensure that the connection is dropped and left in the expected state when the macro is played on the server. The custom action is not displayed in the macro tree, and when the macro is played in Host Access, the custom action is ignored. You might not know the action exists unless you edit the macro.

If you manually edit a macro that contains a custom action with id=″waitForDisconnect″ and remove or modify the previous action so that it no

longer causes a disconnect, you must also remove this custom action to prevent an error when the macro plays on the server.

Macro times out

When a macro times out, or does not receive a response from the host within a predetermined period of time, it is usually because the macro receives unexpected screens that it cannot handle. For example, the host might send screens that are detected by the macro even though they are displayed so quickly they cannot be detected by the human eye. This might not occur during routine testing but only during stress conditions.

Sometimes, it is the lack of specific screen identification that contributes to macro time outs. To enhance your macro's ability to handle all the screens it receives from the host, you should use very specific screen identification. At the Host Publisher support Web site (http://www.ibm.com/software/webservers/hostpublisher/support.html), you can view three examples of this problem with detailed information on fixing it. Go to the Hints and Tips database and search on ″Macro timed out.″

Servlet generated by page compilation reports exception: Wrong name

This error occurs when aliases in httpd.conf are created using the same characters but different case; for example, HostPublisher and hostpublisher. Most application servers do not consider the URL to be case sensitive, so it is possible to specify /HostPublisher/ or /hostpublisher/ in a URL and have it go to the same resource, even if the application server is configured only with the alias /HostPublisher/. The problem is that the PageCompile depends on a case-sensitive path to the JavaServer Page (JSP). After a URL is used to access a JSP, you must use the same capitalization within the URL to access that page in the future.

The same problem occurs if you use a different case the second time you reference a file, for example Tax_Init_Page.jsp and then Tax_init_page.jsp. You must request the page by the *exact* name that was first used, and you must delete the information in the PageCompile.

WebSphere Application Server compiles JSPs into Java servlets, then invokes those servlets to render the actual page to a browser. This Java code remembers the exact location of the original page (for example, d:\WebSphere\AppServer\installedApps\HPAdmin\HPAdmin.war\HPAdmin\main.jsp) so that it can reproduce its HTML content. The servlet is rebuilt from the original JSP only if the page is changed (date stamp is updated). If the location of the JSP changes, but its data stamp does not, you receive an internal error. WebSphere Application Server reports the error after trying to process the JSP because it can no longer find the original file. This can happen if you reinstall the same version of Host Publisher Server in a different location.

To correct this problem, remove WebSphere Application Server's record of the JSP. To do this, remove the corresponding Java and class files from the WebSphere Application Server installation directory; for example:

```
d:\WebSphere\AppServer\temp\host_name\hostpubserver\
\HPAdmin\HPAdmin.war\HPAdmin\main_xjsp.*
```

Unable to access Host Publisher directories on iSeries

If you are attempting to access Host Publisher directories from a browser and you receive a message saying that you are not authorized, add the **DirAccess On** statement to the http configuration.

Generic Web browser timeout message is received instead of an expected Host Publisher error message

To avoid this problem, configure the application server and Web browser timeouts (typically 2 minutes) to be greater than the Host On-Demand Macro Timeout (default is 60 seconds).

To configure the Host On-Demand Macro Timeout, edit the macro. In the first line of the HASCRIPT tag, set the timeout field to the desired value (1000 equals 1 second).

When you specify connection pooling values in Host Access and Database Access, if you set the Maximum Busy Time Before Disconnection to something other than –1 (never), the sum of this timeout and the Host On-Demand Macro timeout should be less than the application server and Web browser timeouts.

Note: In some cases, the Maximum Busy Time Before Disconnection expires and causes the macro to stop; however, a macro timeout error message is generated instead of the busy timeout error message.

Making WebSphere Application Server handle 100 or more requests

Host Publisher is configured to handle a maximum of 50 concurrent connections. If you want more than 100 concurrent connections, you must modify several Host Publisher Server parameters.

1. Increase Max Heap Size for Host Publisher Server.
2. Increase Max Connections for Host Publisher Servlet Engine.
3. Update the application server configuration.

Most application servers have a configurable limit for the number of concurrent clients they will support. You might have to modify the application server configuration to make this limit greater than the Max Connections parameter for the servlet engine.

After making changes to WebSphere and the application server, stop and restart both servers to pick up the new changes.

PluginTester servlet for debugging WebSphere Application Server problems

Host Publisher Version 2.2 and above includes the servlet showCfg, which is the equivalent of the PluginTester. You can access this servlet at the following URL: http://*server*/HPAdmin/showCfg, where *server* is your local server.

Pages are not returned

The error log indicates that Host Publisher ran out of time while trying to set up a connection.

To solve this problem, increase the connecttimeout parameter in the application's .connspec file, and the timeout attribute of the <HAScript> tag in all the macros.

Shutting down Host Publisher Server

If you shut down Host Publisher Server by shutting down the application server or the WebSphere Application Server product, host connections might not be cleaned up properly.

To avoid this problem, stop Host Publisher Server before you stop WebSphere Application Server.

Out-of-memory error starting 20 sessions

Host Publisher creates one thread per pool, and up to 50 threads during session recovery and shutdown. Host Publisher uses Host On-Demand, which creates a maximum of 100 threads regardless of how many sessions are created.

When you configure per process thread limits for a platform, remember that the process is typically used to handle threads of the application server, as well as threads of the WebSphere Application Server product.

CONNECTION_READY errors in the Host Publisher Server logs

If you see multiple error messages in the Host Publisher Server log file that say CONNECTION_ACTIVE but not CONNECTION_READY, the following might have happened:

- You connected to the wrong TN server, for example, a vanilla Telnet server, instead of a TN3270 server.
- You connected to the correct TN server, but did not give the connection setup process enough time to complete.

The Host On-Demand connection goes to the CONNECTION_ACTIVE state when the basic TCP/IP connection between Host On-Demand and the TN server has been set up successfully. The Host On-Demand connection goes to CONNECTION_READY state when SSL, basic Telnet negotiation, and 3270 negotiation are completed. This process requires time, due to 3270 negotiation that involves an SNA session set up between the TN3270 server and the SNA front-end processor.

You should increase the connecttimeout value in your .connspec file. For example, if your connecttimeout value is currently 120, try changing it to 1200. Then stop and restart WebSphere. Changing this value can eliminate most, if not all, occurrences of this error in your log file.

Problems with performance in TN3270E sessions

If you experience poor performance with Host Publisher applications running in a TN3270E environment, you might be able to alleviate the problem by setting the TCPNoDelay property to **true** in the Host On-Demand session properties for your application. The Host On-Demand session properties are defined in the .connspec file for your application. Add the line `TCPNoDelay=true` to your session properties in the .connspec file, as illustrated in the following example:

```
......
<sessionprops>
   TCPNoDelay=true
   host=ralvm17
   autoReconnect=false
   SSL=false
   TNEnhanced=false
   port=23
   ......
</sessionprops>
......
```

When executing macros on TN3270E connections, Host Publisher can send multiple small requests to the TN server as part of the TN3270E protocol, only the last of which results in a response (screen update) from the host application. Normally, the Java networking code waits for a small interval between the sending of subsequent requests to conditionally receive a response from the connection

partner, and thus avoids sending many small packets in the network. For host applications, such responses never arrive, so the wait causes an unnecessary delay that adversely affects performance.

Setting the TCPNoDelay property to **true** should improve performance because the Java networking code does not introduce this unnecessary delay.

Updating Host Publisher using the Software Maintenance Utility

To apply service to Host Publisher, you must use the Software Maintenance Utility. The Software Maintenance Utility automates the process of installation and, where necessary, removal of development-supplied fixes to Host Publisher. The Software Maintenance Utility also stores information about your installation to aid the IBM Software Support Center in determining the cause of problems you might be experiencing. You can find out more information about available fixes to Host Publisher from the Host Publisher Support Web site, at http://www.ibm.com/software/hostpublisher/support.html.

Visit this Web site first upon experiencing problems with Host Publisher. You should download and apply any fix that is called a cumulative APAR. Any other fix is simply an APAR; you can install APARs if it appears that the problems addressed by an APAR will remedy the problems you are experiencing.

Command syntax

The Software Maintenance Utility consists of five programs with a command-line interface. From the Host Publisher installation directory, these programs are in the \Common\service directory. Invoke each command by typing the command in the service directory, followed by case-sensitive command parameters. There are two possible parameters that can follow the command. The first parameter is the fix number, and the second parameter is the component the fix will be applied to, either Studio or Server. In the event that you only have Studio or Server installed, you do not need to specify the second parameter.

For Windows 98, Windows NT, and Windows 2000, type the command. For example, to apply JR12345 to Host Publisher Studio on Windows 2000, type:

```
apply JR12345 studio
```

For AIX, Solaris, and OS/400, prefix the name with sh and add the .sh file extension. For example, to apply JR12345 to Host Publisher Studio on AIX, type:

```
sh apply.sh JR12345 server
```

The following commands are available from within the service directory:

- apply
- restore
- commit
- report

apply

Applies a fix to Host Publisher. Once a fix is applied, you should verify that the fix corrected the problem by attempting to recreate the problem. After you verify that the fix is working as expected, run the **commit** command to make the change permanent. There is no time limit between applying fixes and committing fixes, but you cannot apply a second fix until the first fix is committed.

Examples

Windows 98, Windows NT, Windows 2000:

```
apply JR12345 studio
apply JR12345 server
```

AIX, Solaris, OS/400:

```
sh apply.sh JR12345
sh apply.sh JR12345 server
```

restore

Removes an applied fix from Host Publisher. If a fix does not correct a problem or you experience unexpected results, you can reject the fix and restore Host Publisher to the state prior to applying the fix. After a fix has been removed, you can reapply the fix at a later time. After a fix has been committed, the fix cannot be removed.

Examples

Windows 98, Windows NT, Windows 2000:

```
restore JR12345 studio
restore JR12345 server
```

AIX, Solaris, OS/400:

```
sh restore.sh JR12345
sh restore.sh JR12345 server
```

commit

Makes an applied fix permanent. After you apply a fix and verify that it is working as expected, you should commit the fix. After a fix has been committed, you can no longer use the **restore** command to remove the fix. There is no time limit between applying fixes and committing fixes, but you cannot apply a second fix until the first fix is committed.

Examples

Windows 98, Windows NT, Windows 2000:

```
commit JR12345 studio
commit JR12345 server
```

AIX, Solaris, OS/400:

```
sh commit.sh JR12345
sh commit.sh JR12345 server
```

report

Displays the status of all fixes or a single fix. Issue **report** to display all fixes; issue **report fix_number** to display a single fix.

Examples

Windows 98, Windows NT, Windows 2000:

```
report
report JR12345
```

AIX, Solaris, OS/400:

```
sh report.sh
sh report.sh JR12345
```

Contacting IBM for service

This section lists a number of ways you can reach IBM. Depending on the nature of your problem or concern, the service representative expects you to provide us with information so that we can serve you better.

If you have a technical problem, take the time to review and carry out the actions suggested in this chapter. Use your local support personnel before contacting IBM. Only persons with in-depth knowledge of the problem should contact IBM; therefore, support personnel should act as the interface with IBM.

Before you contact IBM, gather the following information:

1. A complete and accurate description of the problem, including whether the problem occurred in Host Publisher Studio or Host Publisher Server, the task being performed when the error occurred, and the symptoms of the error.
2. If you are running Host Publisher Studio, use the methods described in 5 on page 99 to get information about the problem and its cause.
3. If the error occurred in Host Publisher Server while attempting to access an application, the files for the application might be needed. Refer to the application's .ear file on the server.
4. When you report a problem, be ready to provide information such as product version, current APAR levels, and so forth. You can obtain this information by issuing the **report** command. The **report** command is located in the *install_dir*\Common\service directory; refer to "Updating Host Publisher using the Software Maintenance Utility" on page 115 for more information.

If you determine that you need to contact IBM, you can do any of the following:

- Consult the *Customer Service and Support Guide*, which is a card contained in the product package.
- Access the Host Publisher Web page at http://www.ibm.com/software/webservers/hostpublisher. You can go directly to the Hints and Tips.
- Access the IBM Personal Software Services Web page, which links to the IBM Software Support Handbook, at http://ps.software.ibm.com/.

If you are so directed by IBM service, use the **hppdtool** command to gather problem determination information from Host Publisher. Following are syntax examples for **hppdtool**.

Windows 98, Windows NT, Windows 2000:

```
hppdtool
```

AIX, Solaris:

```
sh hppdtool.sh
```

OS/400 (**hppd400** is the OS/400 version of the **hppdtool** command):

```
sh hppd400.sh
```

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any

obligation to you.

Appendix A. Technical overview

This section provides a brief description of how a Host Publisher application is built and how it executes on the server. It lists two different ways, or models, for executing Integration Objects, which are the building blocks for Host Publisher applications. Then it gives a step-by-step description of how an application is developed and executed under one of the models, that of an application running in a Web container.

Integration Objects are Java beans that encapsulate interactions with data sources like 3270 and 5250 applications and JDBC databases, and return desired data as Java bean properties. Integration Objects can be embedded within JavaServer Pages (JSPs), which enable users of a Web browser to interact with the data. You use the Host Access and Database Access components in Host Publisher Studio to create Integration Objects, and you use the Application Integrator component in Host Publisher Studio to create applications that use the Integration Objects.

On the server, Host Publisher applications execute in the WebSphere Application Server V4.0 environment. They follow the J2EE architecture: each one resides on the server as an .ear (Enterprise Archive) file and consists of one or more J2EE modules, an application deployment descriptor, and any other files referenced by the J2EE modules.

Host Publisher applications can be deployed as Web Services in the WebSphere environment. For more information about developing such applications, see "Chapter 4. Using Web Services" on page 71.

Execution models for Integration Objects

There are three primary models for executing Integration Objects that have been created by Host Publisher Studio. They are:

- An Integration Object executed in a Web context (for example, a Web container) can be used to build dynamic HTML (or other markup language) content using JavaServer Pages (JSP) technology. Alternatively, a servlet that drives one or more Integration Objects can also be developed for dynamic HTML generation.
- An Integration Object can be developed to exploit the benefits of EJB technology such as resource location, replication and load balancing, and to facilitate its use in composite EJB applications. Host Publisher Studio provides special support for building EJB 1.1 support for Integration Object execution, enabling you to develop applications in which Integration Objects are packaged with EJB Access Beans and the Host Publisher EJB. See "Creating applications that use Enterprise JavaBeans (EJB) technology" on page 42 for more information.
- An Integration Object or EJB Access Bean can be developed to become a Web Service. 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. See "Chapter 4. Using Web Services" on page 71 for more information.

Note: It is also possible to develop Integration Objects and EJB Access Beans that execute from a non-WebSphere application server. These are called Remote Integration Objects (RIOs). Rather than developing RIOs, we recommend

developing Web Services applications, which can perform the same functions while running in the WebSphere container.

An Integration Object running in a WebSphere Container

To illustrate the basic functions of Host Publisher, the following section elaborates on the first alternative—an Integration Object using JavaServer Pages. The following figure illustrates the Host Publisher development and execution environments for Web-based Host Publisher applications.

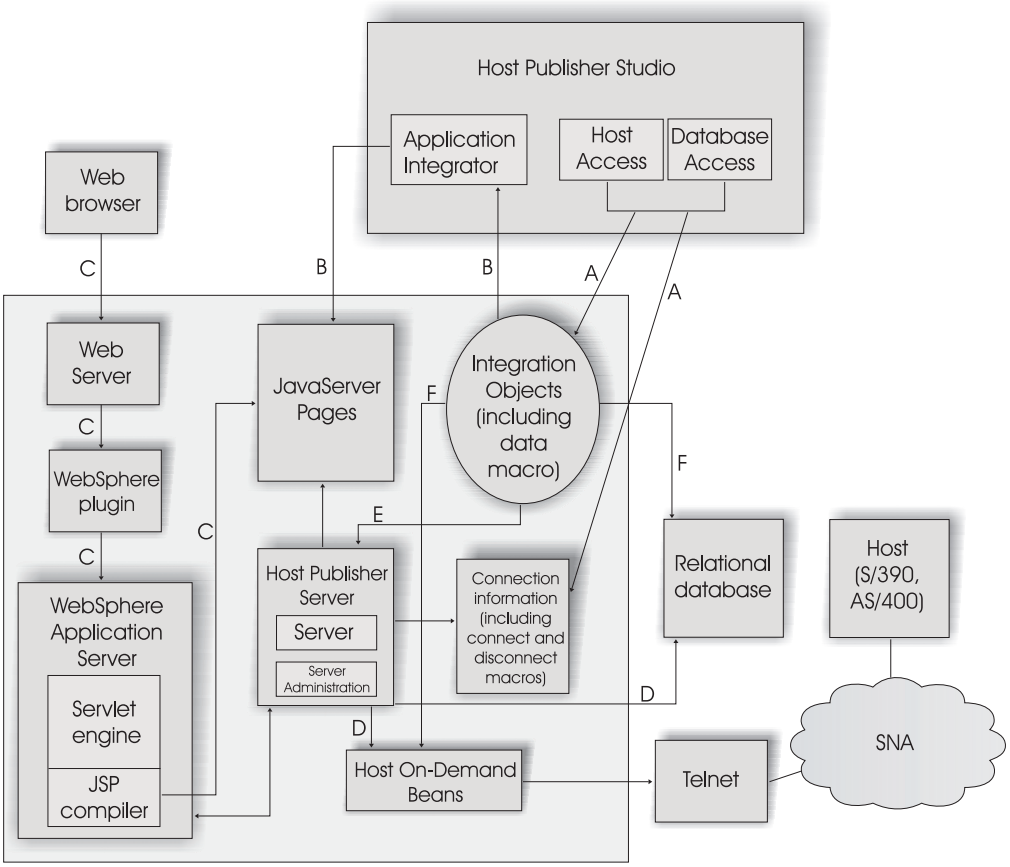


Figure 9. Host Publisher development and execution environments

The Host Publisher Studio applications, Host Access and Database Access, create Integration Objects and connection information representing access to either the host terminal-oriented application or to the relational database (see **A** in Figure 9.) In addition, Host Access creates Host On-Demand macros for executing the connect, data, and disconnect macros for a terminal-oriented application. The Application Integrator component of Host Publisher Studio uses the Integration Object to build JSPs that create the Integration Object, set its parameters, execute it, and retrieve its output **(B)**. These objects and files produced by the Studio are packaged into a J2EE-compliant .ear file which is transferred to the server, where it can be executed as a WebSphere application.

When application .ear files are published and deployed to the server, they are ready to be executed. The execution environment is provided by:

- The Java support provided by an application server created by the WebSphere Application Server.

- Host Publisher Server, which provides administration support for items such as connection pooling, connection setup and priming, user ID management, administration, tracing, and logging.
- Host On-Demand or JDBC code for access to various data sources.

  Note: Host Publisher uses the components of the Host On-Demand product, a Java applet-based terminal emulator, to communicate with terminal-oriented applications. Host On-Demand consists of a set of Java-based Telnet (TN) clients that can communicate with:
  - TN3270(E) servers for the 3270 data stream to communicate with z/OS-based applications
  - TN5250(E) servers for the 5250 data stream to communicate with iSeries-based applications
  - Telnet servers for the VT datastream to communicate with applications that run on various UNIX, OS/2®, and other platforms.

  Host On-Demand also provides programmatic access to these TN client functions to create connections to terminal-oriented applications and navigate through them. Host Publisher uses the programmatic access to Host On-Demand functions.

Execution of a Studio-generated application starts with a browser accessing an application .ear file. WebSphere Application Server contains support for JSP compilation and a Web container that supports the servlet API. The JSP is compiled into a servlet, and the servlet executed **(C)**.

The resulting servlet first creates an Integration Object, then sets its parameters, then invokes it. Invocation of an Integration Object begins with it asking the server for a host connection **(E)**. The server creates the connection either to a TN server for a terminal-oriented application or to a relational database **(D)**. For terminal-oriented applications, the server either establishes a new connection and runs the connect macro using the Host On-Demand API, or it returns an existing connection from a pool. For database applications, the server either establishes a connection with the database or it returns an existing connection from a pool.

After the Integration Object acquires the connection, it either runs a data macro (in the case of a terminal-oriented application) or executes a database statement (in the case of a database application), and retrieves the data from the data source **(F)**. The Integration Object's output properties are now set and ready for retrieval. The rest of the JSP execution uses those output properties for dynamic HTML generation. Integration Object execution ends with it returning the connection to the server, where the server decides whether the connection is returned to a pool or disconnected, and its output properties set.

Appendix B. Server properties files

Server properties files are Java properties files that contain Host Publisher Server settings. There are two types of server properties files: one that contains global settings for a node, and one that contains settings for a particular application server. The server properties files are written to the `Server` directory of the Host Publisher installation directory on the server. When you use Host Publisher Server Administration to make changes, the values you specify are written to the server properties files.

The server properties file for the global settings is server.properties. The file is created when Host Publisher Server is installed. The server.properties file is primed with default values when you start the Host Publisher Server for the first time.

The server properties file for a particular application server is ras_*xxx*.properties, where *xxx* represents the application server name (for example, *domain_node_server*). There is one server properties file for each Host Publisher runtime instance running in an application server. The ras_*xxx*.properties file contains the trace and log settings for that particular application server. If an application server starts the Host Publisher runtime instance within the application server, and the same name is generated for that application server, the trace and log settings are restored to the values that existed prior to a shutdown.

If no value is specified for any of the properties in the server properties files, Host Publisher uses the default value defined for that property.

In addition to using Host Publisher Server Administration to update this file, you can edit the server properties files manually to make changes to the server settings. If you manually edit the files, you should restart the server to ensure that the changes you make take effect.

The server.properties file

The server.properties file contains the following properties:

num_licenses
Specifies the number of licenses you purchased.

The value is an integer that is set during installation. There is no default.

Specify num_licenses = –1 if you purchased an unlimited license.

licenseTracking
Specifies whether Host Publisher tracks license usage or not.

Note: You can only modify this property by editing the server.properties file.

0 Host Publisher does not track license usage.

1 Host Publisher tracks license usage for all application servers in a node. Host Publisher Server tracks the number of Host Publisher connections to host or database resources and logs a message when the value exceeds the number of licenses purchased. The license

usage information is written to a file named license*x*.txt in the `log` directory of the Host Publisher installation directory on the server, where the *x* is either 1 or 2.

The maximum size of the license usage files is 512 KB. When the file size of the license1.txt file reaches 512 KB or if the Host Publisher Server is restarted, the file is renamed to license2.txt, and a new license1.txt file is created. The new license1.txt file contains the most recent license usage information. When the new license1.txt reaches 512 KB and is renamed, the old license2.txt is deleted.

The license usage files contain the following information, arranged in rows, with each row representing one hour of operation. The values are separated by a space ( ).

1. Date
2. Time
3. The highest license count since the server was started
4. The highest license count in the last hour (the maximum of the last 60 entries)
5. The license count for each minute (1–60)

The value is binary. The default is 0.

%logFile
The name used as a template to generate names for each application server file to which log messages are written. Refer to the maxLogFiles property comment in any ras*.properties file for a description of the algorithm for generating each log file name. A ras*.properties file is generated for each WebSphere application server configured to run Host Publisher. When editing the file, specify any backslash in the path with a double slash (\\).

%traceFile
The name used as a template to generate file names for each application server file to which Host Publisher trace messages are written. Refer to the maxTraceFiles property comment in any ras*.properties file for a description of the algorithm for generating each trace file name. A ras*.properties file is generated for each WebSphere application server configured to run Host Publisher. When editing the file, specify any backslash in the path with a double slash (\\).

%HPAdminPortNumber
The port number used by Host Publisher Server on a machine to enable each installation running in an application server on that machine to be administered remotely. Remote administration is currently based on remote method invocation (RMI).

You can specify any valid TCP port number. The default is 30099.

%stashKeyringPW
Specifies whether the password for the express logon keyring database is saved in this file. (See %keyringPW below.)

0 The password is not saved in this file.

1 The password is saved in this file.

The value is binary. The default is 0.

%stashUserListPW
Specifies whether the password for accessing encrypted user list data is saved in this file. (See %userListPW below.)

0 The password is not saved in this file.

1 The password is saved in this file.

The value is binary. The default is 0.

%keyringPW
The password used to access the express logon keyring database. This value is valid if %stashKeyringPW is 1.

%userListPW
The password used to access encrypted user list data. This value is valid if %stashUserListPW is 1.

The ras_xxx.properties file

The ras_*xxx*.properties file contains the following properties:

maxLogFiles
The maximum number of log information files.

The base log file name in server.properties is used as a template to generate unique sets of log files for each application server. The default base name for a log file is *messages.txt*, which can be changed in server.properties. The application server running Host Publisher is the concatenation of: the underscore (_) character, followed by the name of the Host Publisher Server instance, followed by another underscore (_) character.

The Host Publisher Server instance ID (for example *_SSS_*) is then appended to the base file name to generate the template for the log files for an application server. For the log file, this becomes *messages _SSS_.txt*. Finally, an index (1, 2, 3, and so forth) is added to this name to distinguish multiple files. So, for example, if the Host Publisher Server instance ID is *domain_node_server*, the log file for the application server is named *messages_domain_node_server_.txt*. With multiple log files configured, the log file names for this application server are *messages_domain_node_server_1.txt, messages_domain_node_server_2.txt*, and so forth.

When *messages_domain_node_server_1.txt* reaches maxLogFileSize, it is closed and renamed to *messages_domain_node_server_2.txt*. A new *messages_domain_node_server_1.txt* is opened.

When *messages_domain_node_server_1.txt* reaches maxLogFileSize again, previous log files are renamed—for example, *messages_domain_node_server_2.txt* is renamed to *messages_domain_node_server_3.txt*. Then *messages_domain_node_server_1.txt* is renamed to *messages_domain_node_server_2.txt*, and a new *messages_domain_node_server_1.txt* file is opened.

When the maxLogFiles number is exceeded, the oldest file is deleted.

maxLogFileSize
Specifies the maximum size, in kilobytes, that a message log file reaches before an additional log file is opened.

Note: You can only modify this property by editing the ras_*xxx*.properties file.

The value is an integer. The default is 512 KB.

maxTraceFiles
The maximum number of trace information files.

The base trace file name in server.properties is used as a template to generate unique sets of trace files for each application server. The default base name for a trace file is *trace.txt*, which can be changed in server.properties. The application server running Host Publisher is the concatenation of: the underscore (_) character, followed by the name of the Host Publisher Server instance, followed by another underscore (_) character.

The Host Publisher Server instance ID (for example *_SSS_*) is then appended to the base file name to generate the template for the trace files for an application server, which becomes *trace _SSS_.txt*. Finally, an index (1, 2, 3, and so forth) is added to this name to distinguish multiple trace files. So, for example, if the Host Publisher Server instance ID is *domain_node_server*, the trace file for the application server is named *trace_domain_node_server_.txt*. With multiple trace files configured, the trace file names for this application server are *trace_domain_node_server_1.txt*, *trace_domain_node_server_2.txt*, and so forth.

When *trace_domain_node_server_1.txt* reaches maxTraceFileSize, it is closed and renamed to *trace_domain_node_server_2.txt*. A new *trace_domain_node_server_1.txt* is opened.

When *trace_domain_node_server_1.txt* reaches maxTraceFileSize again, previous trace files are renamed—for example, *trace_domain_node_server_2.txt* is renamed to *trace_domain_node_server_3.txt*. Then *trace_domain_node_server_1.txt* is renamed to *trace_domain_node_server_2.txt*, and a new *trace_domain_node_server_1.txt* file is opened.

When the maxTraceFiles number is exceeded, the oldest file is deleted.

maxTraceFileSize
Specifies the maximum size, in kilobytes, that a trace file reaches before an additional trace file is opened.

Note: You can only modify this property by editing the ras_*xxx*.properties file.

The value is an integer. The default is 512 KB.

%logMask
Specifies the types of messages that are logged. Add the values for each of the following message types to determine the value for this property:

1 Informational messages

2 Warning messages

4 Error messages

The value is a decimal integer. The default is 4.

Note: Error messages are always logged.

%traceMask

Specifies the types of traces for the Server and the Integration Objects. This property does not affect JDBC or Host On-Demand tracing. Add the values for each of the following traces to determine the value for this property:

16 API traces

384 Entry and exit traces

1024 Miscellaneous traces

The value is a decimal integer. The default is 1424, meaning that all trace types are enabled.

%HODDisplayTerminal

Specifies whether Host On-Demand displays a terminal window for each connection or not.

0 Host On-Demand does not display a terminal window for each connection.

1 Host On-Demand displays a terminal window for each connection.

The value is binary. The default is 0.

If %HODDisplayTerminal is 1, when Host Publisher creates a host connection (for example, in response to a request from an Integration Object), it automatically creates a host terminal display. If this property is set to 0, this does not occur; however, whether or not the host terminal display is created automatically during host connection creation, a Host Publisher administrator can use Host Publisher Server Administration to turn host terminal displays on or off for individual host connections.

%HODMacroTracingLevel

Specifies the level of tracing for the Host On-Demand macros.

The value is an integer in the range 0 to 3, where 3 is the maximum level of tracing. The default is 0, which means there is no tracing.

%HODPSTracingLevel

Specifies the level of tracing for the Host On-Demand presentation space.

The value is an integer in the range 0 to 3, where 3 is the maximum level of tracing. The default is 0, which means there is no tracing.

%HODSessionTracingLevel

Specifies the level of tracing for Host On-Demand sessions.

The value is an integer in the range 0 to 3, where 3 is the maximum level of tracing. The default is 0, which means there is no tracing.

%HODSupportTracing

Specifies whether Host On-Demand Support Tracing is enabled.

0 Host On-Demand Support Tracing is not enabled.

1 Host On-Demand Support Tracing is enabled.

The value is binary. The default is 0.

%HODTransportTracingLevel

Specifies the level of tracing for the Host On-Demand transport.

The value is an integer in the range 0 to 3, where 3 is the maximum level of tracing. The default is 0, which means there is no tracing.

%HODUserMacroTracing
Specifies whether Host On-Demand User Macro Tracing is enabled.

0 Host On-Demand User Macro Tracing is not enabled.

1 Host On-Demand User Macro Tracing is enabled.

The value is binary. The default is 0.

%ioPatternKey
One or more patterns specifying the Integration Objects to be traced. (See %ioTrace below.)

Each pattern can contain one or more wildcard (*) character. For example, *IntegrationObject.Callup** specifies that tracing is enabled for all Integration Objects that start with the letters `Callup`. To trace all Integration Objects, specify *IntegrationObject.**

If multiple patterns are specified, they should be delimited with commas.

%ioTracing
Specifies whether Host Publisher traces Integration Objects specified by %ioPatternKey..

0 Integration Objects are not traced.

1 Integration Objects are traced.

The value is binary. The default is 0.

%JDBCTracing
Specifies how much JDBC activity Host Publisher traces.

0 Host Publisher does not trace JDBC activity.

1 Host Publisher traces all JDBC activity in the application server.

The value is binary. The default is 0.

%rioTracing
Specifies whether Host Publisher traces Remote Integration Object (RIO) servlet processing.

0 RIO servlet processing is not traced.

1 RIO servlet processing is traced.

The value is binary. The default is 0.

%runtimeTracing
Specifies whether Host Publisher traces runtime activity or not.

0 Host Publisher does not trace runtime activity.

1 Host Publisher does trace the runtime activity.

The value is binary. The default is 0.

Appendix C. Example of developing an application in Host Publisher Studio

In this example, you will develop a simple Host Publisher application that extracts and displays data from a host application. For example, the host application might be a company telephone directory that displays employees' names and phone numbers. The example guides you through the three major steps of developing an application:

- Creating an Integration Object
- Building the application
- Transferring the application to a server

During the example you'll log onto a host session, run an application on the host, extract data from the application, and log off from the application. Then, using the Integration Object you built during the first part of the example, you'll use Application Integrator to build a Web application. The application will define how the extracted data is displayed for an end user and how the end user can manipulate the data.

This example assumes that the following menu items in the Host Access Options menu are checked (these are the default values):

- Automatically Generate Integration Objects on Save
- Prompt on Unrecognized Screens

In the following steps, the underlined text corresponds to the title that shows in the wizard pane as you perform each step.

For details about the procedures described in this example, see "Chapter 2. Using Host Publisher Studio to develop J2EE applications" on page 9.

Creating an Integration Object

1. Open the Application Integrator component of Host Publisher Studio. The Welcome to Host Publisher! window displays.
2. Click **Create Integration Objects**.
3. Select **Host Access Integration Object** to specify the type of Integration Object you will create.
4. Click **OK**.

After a few seconds, the Host Publisher Host Access main window displays. Most Host Access windows are divided into three sections, or panes:

- The Object Configurations pane on the left
- The wizard pane in the upper right
- The terminal pane in the lower right

Note: The terminal pane does not appear until you have connected to the host.

Welcome

Click **Next**.

Integration Object

The **Create a new Integration Object** radio button is selected by default, so click **Next**.

Host Configuration

Supply information with which the Host Publisher application connects to the host. Click **Help** for more information about the settings.

1. In the Connection Pooling Information box, the **Configure a default connection pool** radio button is the default selection.
2. In the Connection Information box, the **Create a new connection configuration** radio button is the default selection.
3. In the Host Server Information box, fill in the following fields with your installation-specific information or accept the defaults.
   - *Server name* – this is always required.
   - *Terminal type* – TN3270 is the default. This is typically used for connecting to a zSeries (S/390®) host.
   - *Port number* – 23 is the default.
   - *Code page* – defaults to the language setting for your workstation.
   - *Screen size* – 24x80 is the default.
4. Leave the **Express Logon enabled** box unchecked.
5. Click **Next**.

The terminal pane, containing a host session, should open. If it does not open, confirm that you have connectivity to the host and are using the correct host configuration.

Begin Recording

Click **Next** to assign a name to the first screen in the host session (the screen displayed in the terminal pane). This action begins a series of screen-definition windows.

New Screen Definition

This is the first of four panes that prompt you for information to define a screen. The panes are presented whenever you define a screen.

Accept the default name *Screen1* and click Next.

Add Recognition Criteria to Screen Definition – Screen1

Accept the default, **Text Area**, to indicate that the screen is recognized by text on the screen. Click **Next**.

Text Area Recognition

1. Use the mouse to draw a box around some text on the terminal. Select text that is unique to this screen and does not appear on other screens.

   The box can include more than one line of text. For example, if the words User ID and Password are on the screen, you can draw a box around them.
2. Click **Next**.

Screen Definition

This is the last screen-definition window. It enables you to add additional screen recognition criteria.

Click **Next**.

Connect Your Session

In the terminal pane, connect your session by logging on to the host as you would from a terminal emulator. The keystrokes you enter are recorded, and you can see them added to the macro tree in the left pane. This is a tree representation of the connect macro. (In Host Publisher, a macro is an XML script that defines a set of screens, actions, and next screens.)

When the terminal screen changes as a result of your input, the Unrecognized Screen window displays.

Unrecognized Screen

Because the Prompt on Unrecognized Screen option is selected, this window displays whenever the terminal screen changes to a screen you have not already defined. (You can check the check box to turn this option off.)

You might have to navigate through more than one screen to reach the data you want to extract for this example. Throughout this example, whenever you see the Unrecognized Screen window, perform the following steps:

1. Click **Yes**.
2. The New Screen Definition window displays. Respond to this window, and to the screen-definition windows that follow it, as you did for *Screen1*, accepting the default screen names.
3. When screen definition is complete, you are returned to one of the following windows:
   - Connect Your Session if you were recording the connect macro.
   - Gather Data if you were recording the data macro after selecting **Gather Data**.
   - Disconnect Your Session if you were recording the disconnect macro.

   (The Gather Data and Disconnect Your Session windows are described in the sections below.)
4. Click **Next**.

You have finished recording the connect macro, and you are ready to begin recording the data macro.

Begin Data Macro

1. In the wizard pane, click **Gather Data**.
2. Click **Next**.

Gather Data

1. In the terminal pane, navigate to the data you would like to gather. As you enter keystrokes and define screens, Host Access updates the tree representation of the data macro in the Object Configurations pane.

2. When you reach the screen from which data is to be gathered, define the screen.
3. Click **Next**.

Select Data

1. Using the mouse, draw a box in the terminal pane around the data you want to gather.
2. Click **Next**.

Data Format

1. Select **Extract data as plain text**. (Selecting **Extract data as a table** takes you through a different set of windows. However, both paths end at the Finish Data Extraction window.)
2. Click **Next**.

Text Extraction

1. The terminal pane is replaced by the extracted text. In the wizard pane, enter a name for the extracted text. You will use this name again later, as you build your application, but the end user of the application will not see the name.
2. Click **Next**.

Finish Data Extraction

1. When the data extraction is complete, click **Extract more data** if you want to gather more data from the same screen or from another screen in the application.
2. When you are finished gathering data, select **Finished extracting data**.
3. Click **Next**.

Finish Data Macro

1. In the terminal pane, navigate to the screen from which you will log off from the host. Typically, this is the same screen as the first screen in the data macro.
2. In the wizard pane, click **Next**.

   Note: If the current screen does not match the first screen in the data macro, a warning message asks whether you would like to proceed. (A similar warning will also display when you save the Integration Object.) For this example, which does not involve connection pooling, you can click **Yes** in response to the warning.

You have finished recording the data macro, and you are ready to begin recording the disconnect macro.

Disconnect Your Session

1. In the terminal pane, log off from the host.
2. When the host connection has been terminated, click **Next** in the wizard pane.

Save Integration Object

1. In the wizard pane, click **Next** to save your Integration Object. The connection pool and macros will also be saved.
2. Type a name for the Integration Object and click **Save**.
3. Wait a few seconds while the Creating Integration Object window displays the status. Then click **OK** in the Creating Integration Object window.

At this point you can verify that your macros play correctly.

1. Select Connect Macro in the Object Configurations pane.
2. Play the connect macro by clicking the Play Macro icon in the menu bar or by clicking **Macro > Play**. Host Access identifies host screens with the criteria you have defined and sends the keystrokes you recorded for the identified screens to the host.
3. Select Data Macro in the Object Configurations pane.
4. Play the data macro. A popup window displays the data that is extracted. This data will be available to send to end users.
5. Select Disconnect Macro in the Object Configurations pane.
6. Play the disconnect macro.

Building the application

To begin building an application which will contain the Integration Object you just defined:

1. Select **File > Exit** from the menu bar to exit the Host Access component. You are returned to Application Integrator.
2. Click **Create Application** on the welcome window, or select **File > New Application** from the menu bar.

New Application

1. Enter the name of the application.
2. Click **Next**.
3. Click **I prefer to start with data**.
4. Click **Next**.
5. Click **Import** to bring the Integration Object you defined earlier into the application.
6. Select the Integration Object from the list and click **Open**.
7. The name of the Integration Object appears in the table with No in the **Defined** column. Click **Define** to create Web pages (HTML and JSP pages) containing the Integration Object. The Web pages will contain the code required to execute the methods of the Integration Object and to render (display) the resulting data.

New Integration Object

You'll now use wizards to define how the data you extracted from the host application will appear on your Web page. This process is known as rendering.

1. In the first window, **Create a new page** is the default selection. Type the name of the page which will display the data output from your Integration Object, then click **Next**.

   A table displays, containing a list of the output variables you defined in the Integration Object. These variables represent the data you extracted from the host application.
2. Select an output for which Single appears in the **Values** column (meaning that the data was extracted as a single plain text string), then click **Render**.

Insert Output Control

This wizard enables you to specify how the selected output data will be displayed on the Web page. For a single data string, your choices are an edit box or a

password box (where an end user can modify the data), or a normal text string (where the data is displayed and cannot be modified).

1. Select **Normal text**.
2. Click **Next**.
3. Type a label to describe to end users what this data represents.
4. Click **Finish**.

New Integration Object

The output data is now shown in the table with the type of control you selected (normal text) and the first few words of your label.

1. If you extracted more than one data element from the host application, you can repeat the previous steps with other output variables.
2. Click **Next** when you are finished defining your output. You have defined your output Web page.
3. Click **Preview** to see the Web page in a Web browser. All you will see now, however, is the label you defined. The Integration Object gathers data from the host only after you have imported it into an application and transferred the application to a Host Publisher Server.
4. Click **Finish** to finish defining the Integration Object.

New Application

You are returned to the New Application wizard, where you started to define your Integration Object. At this point you could click **Create** to create another Integration Object or **Import** to add a previously created Integration Object and repeat the definition process.

To continue with this example, follow these steps:

1. Click **Next**.
2. Click **Create** to create an error page—the page that is invoked when an application error is encountered.

New Error Page

To create a custom error page that will handle execution errors found in your application, follow these steps.

1. Type the name of the error page.
2. Click **Next**.
3. Customize the error message that will appear when an error is found. For example, your error message can tell end users how to reach the support staff and what information to give them.
4. Click **Finish** to finish creating the error page.

New Application

Click **Finish**.

You have finished creating the new application. In the Application Pages pane, on the right side of the screen, you can see a list of the pages you created. The error page is displayed at the top of the list because it is the most recent page you created.

You can update your application pages using the menus on the menu bar. Many of the menu items take you to the same wizards you used to create your application. You can use them to:

- Add more new pages to the application
- Add additional variables or controls to the pages

When you have finished, click **File > Save Application** to save the application, or continue with this example by transferring the application to a Host Publisher Server.

Transferring the application to a server

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

Transfer to Server

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**.

Host Publisher Server Definition

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. 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.
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.

Appendix D. Examples for designing custom Web pages

Use the following examples to design custom Web pages. To cut and paste an example, use the online version of this book, located at Start > Programs > Host Publisher [Server or Studio] > Library > Administrator's and User's Guide.

Note: You can find more code examples on the Host Publisher Web page (http://www.ibm.com/software/webservers/hostpublisher/library.html) under Product documentation.

Java access to page parameters

How can I get access to parameters passed to my Web page?

Write a Java scriptlet to call the request.getParameter method and specify the name of the input parameter.

This example gets the values of input parameters tu_in and dupno_in and places them in String variables. They are then displayed as part of HTML h2 headers.

```
<%
   String tu_num = request.getParameter("tu_in");
   String dup_num = request.getParameter("dup_in");
%>

<h2>Val1 is <%=tu_num %></h2>
<h2>Val2 is <%=dup_num %></h2>
```

Redirecting based on Integration Object results

How can I test the output of an Integration Object and call subsequent JavaServer Pages (JSPs) based on the results of my test?

Write a Java scriptlet to get the desired property from the executed Integration Object. Test the property and redirect to the appropriate JSP based on the test.

This example calls getTaxmenuid and tests the value to determine the page to redirect to.

```
<%
   String status=Tax_Menu_Init.getTaxmenuid();
   if (status.indexOf("Sign") != -1){
      response.setStatus(response.SC_MOVED_TEMPORARILY);
      response.setHeader("Location","TaxSignon_Error.jsp");
   out.close();
 return;
}
%>
```

Invoking Integration Objects based on previous Integration Object results

How can I test the output of an Integration Object and not invoke a subsequent Integration Object if the first Integration Object failed?

Write a Java scriptlet to test a property from the first executed Integration Object. Use an IF statement for the test and wrap the execution of the subsequent bean within the IF.

In this example, the Tax_Details_F12 Integration Object will execute ***only*** if the word DUPLICATE is ***not*** found in variable status.

```
<%
   String status = Tax_Addr.getTaxstatus();
   if (status.indexOf("DUPLICATE") == -1) {
   %>

      <jsp:useBean id=Tax_Details_F12" type="IntegrationObject.Tax_Details_F12"
       class="IntegrationObject.Tax_Details_F12" scope="request">
      </jsp:useBean>
      <jsp:setProperty name="TaxDetails_F12" property="*" />
      <% Tax_Details_F12.setHPubStartPoolName("Tax_Details_F12"); %>
      <% Tax_Details_F12.doHPTransaction(request, response); %>

<%}%>
```

Building dynamic HTML based on Integration Object properties

How can I build dynamic output based on the properties of an Integration Object?

Write a Java scriptlet that will use out.println to write output into the HTML stream. Properties of the Integration Object can be embedded in the out.println statements.

```
<%
String empno;
try
{
empno = Dbfirst.getDANAPEMPLOYEEEMPNO_(0);
out.println("<P>Employee number: <b>" + empno + "</b><p>");
out.println(
   Dbfirst.getDANAPEMPLOYEELASTNAME_(0) + "," +
   Dbfirst.getDANAPEMPLOYEEFIRSTNME_(0) + "" +
   Dbfirst.getDANAPEMPLOYEEMIDINIT_(0) + "." +
);

out.println("<p>Show <A HREF=\"showfirst.jsp?empno=" + empno +
 "\">Next</A> Employee");
out.println("<p> <A HREF=\"Deleted.jsp?empno=" + empno +
 "\">Delete</A> Employee");

}
catch(Exception e)
{
}
%>
```

Validating user input

How can I validate user input from an HTML form?

Use JavaScript to validate user input on an HTML page.

In this example, the function is called by *onSubmit="return padzero(this);"* on the input HTML FORM statement. Function padzero ensures that the input value is numeric and is between 1 and 999999 inclusive.

```
<SCRIPT LANGUAGE="JavaScript1.1">
function padzero(f)
{
   var num = parseInt(f.elements[0].value,10);
   if (num < 1 || num > 999999 || isNaN(num) ||
     num != f.elements[0].value )
    {
     alert("Employee Number not valid ... value must be between" + " " 1 and 999999");
     return false;
    }
   var len = f.elements[0].value.length;
   var zeros = "";
   for (var i = len; i < 6; i++)
    zeros = zeros.concat("0");
   f.elements[0].value = zeros.concat(f.elements[0].value);
   return true;
}
</SCRIPT>
```

Testing for successful database record deletion

How can I test to ensure a database access record delete is successful?

Use a Java Scriptlet to query the Integration Object.

This example calls getHPubNumberOfRowsChanged and tests to see if a row was changed.

```
<%
   DbDelete.setHPubStartPoolName("Db");
   DbDelete.doHPTransaction(request, response);
   int changed;
   changed = DbDelete.getHPubNumberOfRowsChanged();

   if (changed == 0)
{   out.println("Error deleting employee number");   }
   else
   if (changed == -1)
   {   out.println("No entry found for employee number ");   }
   else
   {   out.println("Successfully deleted employee number");   }
%>
```

Testing for successful database record addition

How can I test to ensure a database access record addition is successful?

Use a Java Scriptlet to query the Integration Object.

This example calls getHPubNumberOfRowsChanged and tests to see if a row was changed. Note that a write of the empno and name into the HTML stream is also in the scriptlet.

```
<%
   DbAddMin.setHPubStartPoolName("Db");
   DbAddMin.doHPTransaction(request, response);
   int changed;
   changed = DbAddMin.getHPubNumberOfRowsChanged();

   if (changed == 0)
   {   out.println("Error adding employee number");   }
   else
   {   out.println("Successfully added employee number");   }
```

```
   out.println("<b>" + DbAddMin.getEmpno());
out.println("</b> (" + DbAddMin.getLast() + "," + DbAddMin.getFirst()    + " " +
 DbAddMin.getMiddle() + ".).");

   if (changed == 0)
   {   out.println("<p>Employee number may already be assigned.");   }
%>
```

Passing Java variables to JavaScript function

How can I pass Java variables to JavaScript?

In this example, maxfields is passed to JavaScript function CheckData.

```
<% int maxfields = 99; %>

<SCRIPT Language="Javascript">
function CheckData(formitem, max)
{
   if (formitem.fgn_number.value > max)
      alert("Value is too large. Please re-enter")
}
</SCRIPT>


Sample HTML form to call function:
   <FORM Name="testform" Method="pose"
      onSubmit="return CheckData(this,<%= maxfields %>)" >
     <input type="text" name="fgn_number" size="12" maxlength="12"></p>
   </FORM>
```

Using Java to display variables passed into a page

How do I display the variables that are passed into a page?

Use a Java scriptlet and call request.getParameter(″<name>″), then use out.println. Parameters are sent to pages as Name/Value pairs.

```
<%
   out.println("Test Bad");
   String payee_name = "";
   String fgn_input = request.getParameter("fgn_number");
   out.println(fgn_input + " ");
   out.println(fgn_input.length());
%>
```

Using Java to pad an input value and passing it to an Integration Object

How can I pad input data to the correct length without using JavaScript on the client side?

You can use Java to pad an input variable to the proper length for the host application.

In this example, the dcn_number is padded to be 9 characters.

```
<%
   String dcn_input = request.getParameter("dcn_number");
   String final_dcn_value = "";
   dcn_input = dcn_input.trim();
   for( int i = dcn_input.length(); i < 9; i++)
     final_dcn_value = final_dcn_value.concat("0");
```

```
   dcn_input = final_dcn_value + dcn_input;
%>


<jsp:useBean id="Okdhs_dbupdate" class="IntegrationObject.Okdhs_dbupdate
 SCOPE="request">
</jsp:useBean>
<jsp:setProperty name="Okdhs_dbupdate" property="*" />
<% Okdhs_dbupdate.setHPubStartPoolName("OKLADHSDB"); %>
<% Okdhs_dbupdate.setDcn_number(dcn_input); %>
<% Okdhs_dbupdate.doHPTransaction(request, response); %>
```

Using a function type passed from a hidden HTML form variable to determine page to execute

How can I use one input form to redirect to different host functions?

Use an onClick method to set a function type in a hidden HTML form variable. This variable can then be tested by Java to determine correct JSP to handle the function. A sample form follows the example.

```
<%
   String function_selected = request.getParameter("funtype");
  try {
     db_pin_value = Okdhs_dbselect.getDMCPEAKEOKLADHSDCN_(0);
     if (db_pin_value.indexOf(pin_number) != -1)
     {
      if(function_selected.indexOf("CHANGE") == -1)
        newurl = "cfrr_page.jsp?fgn_number=" + fgn_input;
      else
        newurl = "change_pin.jsp?fgn_number=" + fgn_input;
     } else {
      newurl = "cfrr_error.jsp?fgn_number="+
        request.getParameter("fgn_number");
      response.setStatus(response.SC_MOVED_TEMPORARILY);
      response.setHeader("Location",newurl);
     }
    }
  catch(exception e)
  {
   newurl = "cfrr_error.jsp?fgn_number=" +
      request.getParameter("fgn_number");
   response.setStatus(response.SC_MOVED_TEMPORARILY);
   response.setHeader("Location",newurl);
  }
%>
```

Sample HTML form:

```
<form name="subform" method="POST"
   ACTION="<%=response.encodeURL("csml_page.jsp") %>">
   <p align="center"><font face="Comic Sans MS">Please type your identification
   number</font>
   <input type="text" name="fgn_number" size="12" maxlength="12"></p>
   <p align="center"><font face="Comic Sans MS">Please enter your PIN</font><input
type="password" name="pin_number" size="9" maxlength="9"></p>

   <input type="hidden" name="funtype" size="6" maxlength="6">

   <p align="center"><input type="submit" value="Inquire" name="B1"
onClick="subform.funtype.value='INQUIRE'">  
   <input type="reset" value="Clear Form" name="B2">  
   <input type="submit" value="Change PIN" onClick="subform.funtype.value='CHANGE'">
   </p></p>
</form>
```

Using Java to prevent blank lines in an HTML table

If I extract more data than is available, how do I prevent blank lines from being added to the HTML table?

Use a Java scriptlet to test the property of the Java bean. If it is blank, break out of the repeat loop.

```
<HTML>
<BODY>
<jsp:useBean id="Ha_mcat1" class="IntegrationObject.Ha_mcat1" scope="request">
</jsp:useBean>
<jsp:setProperty name="Ha_mcat1" property="*" />
<% Ha_mcat1.setHPubStartPoolName("mcatpool"); %>
<% Ha_mcat1.doHPTransaction(request, response); %>
<P>Search Results:<TABLE BORDER>
<th>title</th>
<th>author</th>
<% for (int idx1 = 0; idx1 %= 2147483647; idx1++) { try { %><tr>
<%
if (Ha_mcat1.getTitletitle(idx1).indexOf(" ")!=-1) break;
%>
<td><%= Ha_mcat1.getTitletitle(idx1) %></td></tr>
<% }
catch (java.lang.ArrayIndexOutOfBoundsException ae) {
break;
}
catch (java.lang.NullPointerException ae) {
}
}
%>
</TABLE>
</Body>
<% out.close(); %>
</HTML>
```

Using Java to control display of an HTML table based on host results

If there is no data for the table extracted from the host, how can I prevent the display of an empty table?

One way is to also extract a status line from the terminal screen, assuming a status line exists. Then use an **if** statement to test the status line to determine if the table should be displayed.

Note: This pertains only to Integration Objects created by the Host Access application.

```
<% String status_value = Spurs_add1.getStatus_line();
   // If "Vendor not found" was in status line, do not display
   // the table and instead display a message.
   if (status_value.indexOf("VENDOR NOT FOUND") == -1 ) {
%>
<P>Vendor Results:<TABLE BORDER>
<th>Vendors</th>
<% for (int idx1 = 0; idx1 <= 2147483647; idx1++) {
       try {
%>       <tr>
         <td><%= Spurs_add1.getVendors_data(idx1_ %></td>
<%     } catch (java.lang.ArrayIndexOutOfBoundsException ae) {
         break;
       }  catch (java.lang.NullPointerException ae) {
         break;
       }
```

```
    }
%>
</TABLE>
<% } else { %>
     <h3 align="center">Vendor not found in vendor file.</h3>
<% } %>
```

Determining number of page downs and tabs for making a selection

How can I code the Host Publisher application such that it selects one item when the host application contains lists of items on multiple terminal screens?

There are two ways to accomplish this.

1. Use Integration Object chaining and return one screen of items at a time. When the user clicks an item in the list box, use the index of the selected item to select the entry on the screen.
2. Return all the items and display them in a list box. Position the host screen on the first item. Use the following example to calculate the number of page downs and tabs required to get to the proper entry. The value passed in from the previous page is the index of the item in the list box.

Note: This pertains only to Integration Objects created by the Host Access application.

This example assumes there are 9 items per host screen. If your application has a different number per screen, change this value.

```
<HTML>
<BODY>
<!-- Retrieve the parameter passed into this page -->
<!-- This parameter indicates which entry was selected from the inspection list -->
<% int selNum = new Integer(request.getParameter("sel")).intValue(); %>

<!-- Calculate the number of pages downs and tabs -->
<% int numPgDowns =0;
   numPgDowns = selNum / 9;
   int numTabs =0;
   numTabs = selNum%9;
   out.println("numTabs string is "+numTabs);
   out.println("selNum string is "+selNum);
%>

<!-- Build the strings to cause the macro to return to the proper screen -->
<%
   String pgDownString = new String();
   String tabsString = new String();
   //
   //out.println("adding "+numPgDowns+" page downs");
   //out.println("adding "+numTabs+" tabs");

   for(int i =0; i < numPgDowns; i++)
      pgDownString=pgDownString+"[pagedn]";

   //out.println("page down string is "+pgDownString);
   for(int j =0; j < numTabs; j++)
      tabsString += "[tab]";

%>

<jsp:useBean id="ha_lab4" class="IntegrationObject.Ha_lab4" scope="request">
</jsp:useBean>
<jsp:setProperty name="Ha_lab4" property="*" />
```

```
<% Ha_lab4.setHPubStartPoolName("Halab4"); %>
<% Ha_lab4.setOpt(pgDownString+tabsString); %>
<%Ha_lab4.doHPTransaction(request, response); %>
```

Changing the action value of a form based on the clicked button

How can I use one form to go to a different page based on the user's selection?

The action value for a form can be changed dynamically.

```
<%
   String invoice_url = response.encodeURL("WCB_clinic.jsp");

   String cheque_url = response.encodeURL("WCB_chk_logon.jsp");
%>
<FORM>
   <INPUT type="submit" value="Enter Invoice"
      onClick = "document.forms[0].action = '<%=invoice_url %>'">

   <INPUT type="submit" value="View Check Information"
      onClick = "document.forms[0].action = '<%=cheque_url %>'">
</FORM>
```

Using HTTP Session object to pass values

How can I pass values from one page to a subsequent page?

Store the values in an HTTP session object.

```
<!-- First page stores the values. -->
<%
HttpSession sess = request.getSession(true);
if (sess!=null) {
  sess.setAttribute("caregiver_type", WCB_caregiver.getCaregiver_type()); ;
  sess.setAttribute("caregiver_idn", WCB_caregiver.getCaregiver_idn());
}
%>

<!-- Subsequent page retrieves the values. -->
<%
HttpSession sess = request.getSession(false);
if (sess != null) {
  WCB_fee_code.setCaregiver_type((String)sess.getAttribute("caregiver_type"));
  WCB_fee_code.setCaregiver_idn((String)sess.getAttribute("caregiver_idn"));
}
%>
```

Disabling the browser back button

How can I disable the browser back button?

Place the following JavaScript on every page, including the logon page.

This example uses the browser's history object to tell the browser to go forward after the page is loaded.

```
<HTML>
<HEAD>
<SCRIPT Language="Javascript">
function goHist(a)
{
   history.go(a);   //Go forward one.
}
</SCRIPT>
</head>
```

```
<body onLoad="goHist(1);">
   some HTML ...
</body>
</HTML>
```

Using Java to control which HTML table to display based on host results

If there are multiple tables on the JSP that can display, how can I prevent the display of an empty table HTML on the JSP?

You can extract a property and use Java logic to test that property. Then use an IF statement to test the status line to determine if the table should display. Note the use of the ELSE statement.

```
<HTML>

<%@ page contentType="text/html;charset=UTF-8" errorPage="DefaultErrorPage.jsp" %>
<BODY>
<jsp:useBean id="Mcat_conditional" class="IntegrationObject.Mcat_conditional"
 scope="request>
</jsp:useBean>
<jsp:setProperty name="Mcat_conditional" property="*" />
<% Mcat_conditional.setHPubStartPoolName("Mcat_keyw"); %>
<% Mcat_conditional.doHPTransaction(request, response); %>
<% String s = Mcat_conditional.getJim();
   if (s.indexOf("NAUGLE J.C.") != -1) {
%>
<P>Author Index Results:<TABLE BORDER>
<th>indexall_data</th>
<% for (int idx1 = 0; idx1 <= 2147483647; idx1++) {
       try {
%>
<tr><td><%= Mcat_conditional.getAuthor_indexall_data (idx1) %></td></tr>
<%
       }
       catch (java.lang.ArrayIndexOutOfBoundsException e) {
          break;
       }
       catch (java.lang.NullPointerException e) {
          break;
       }
     }
%>
</TABLE>
<% } else { %>
<P>Author Guide Results:<TABLE BORDER>
<th>guidelast</th>
<th>guidefirst</th>
<% for (int idx2 = 0; idx2 <= 2147483647; idx2++) {
       try {
%>
<tr><td><%= Mcat_conditional.getAuth_guidelast(idx2) %></td>
<td><%= Mcat_conditional.getAuth_guidefirst(idx2) %></td></tr>
<%     }
       catch (java.lang.ArrayIndexOutOfBoundsException ae) {
          break;
       }
       catch (java.lang.NullPointerException ae) {
          break;
       }
     }
%>
```

```
</TABLE>
<% } %>
</BODY>
</HTML>
```

Appendix E. Glossary

This glossary lists some of the terms used in this book along with their meanings.

administrator
The person who uses Host Publisher Server Administration to configure and monitor Host Publisher Server. Also the person who typically deploys applications on the server. (Contrast with **user**.) (See also **developer**.)

APAR Authorized Program Analysis Report. A code fix generated by IBM service in response to a software problem.

application bundler
A function in Application Integrator that enables you to create a .zip file for copying application files to another Host Publisher Studio without assembling the application into an .ear file. This function is invoked using **Bundle Application Source** on the File menu.

Application Integrator
The component of Host Publisher Studio with which the developer creates applications and transfers them to the server. (See also **Database Access** and **Host Access**.)

application server
A Java virtual machine (JVM) started by WebSphere on a node, on which assembled applications run.

AppMigrator
A tool in Host Publisher Server that migrates applications to a Host Publisher 4.0 level. It migrates the applications to JSP 1.1 and EJB 1.1 levels, inserts UTF-8 encoding, updates error pages, and places application files to *install_dir*\Server\migration\migratedApps.

assemble
To create application .ear files in preparation for transferring the application to the server. (Synonymous with **package**.) (Contrast with **bundle**.)

bundle
To collect directory structures and application files in preparation for sharing the application with another user of Host Publisher Studio. (See also **application bundler**.) (Contrast with **assemble** and **package**.)

connection pool
A collection of communication links to back-end data sources, such as 3270 applications or databases. When an Integration Object is run on behalf of a client request, the Integration Object obtains an available connection from a pool, uses it for access to the data source, then returns the connection to the pool for reuse by another Integration Object. When connections are pooled, the overhead of establishing a connection is absorbed in its first use. Each Integration Object reusing this connection benefits from the prior establishment of the connection and can run faster. (See also **connection pooling**.)

connection pooling
A method of keeping a connection to a host open for many data transactions, instead of opening and closing a connection with each new request. Designed to reduce the response time between a request from a

client browser and the display of the requested information on a Web page, connection pooling refers to the sharing of connections—which are listed in a connection pool—by applications. Connection pooling is enabled using the Host Access component of Host Publisher Studio. (See also **connection pool**.)

container
In J2EE, an entity that provides life-cycle management, security, deployment, and run-time services to components.

Database Access
The component of Host Publisher Studio with which the developer creates Integration Objects that encapsulate a database statement. (See also **Application Integrator** and **Host Access**.)

deploy
To make a Host Publisher application ready to use on the server, using functions in WebSphere Application Server, after transfer has taken place. Note that WebSphere documentation often uses the term *install* as a synonym for this process. (See also **publish** and **transfer**.)

developer
The person who uses Host Publisher Studio to develop applications; also application developer or Web developer. (Contrast with **user**.) (See also **administrator**.)

.ear file
The J2EE-format file that contains the Integration Objects, other Java objects, and configuration information for an application. .ear stands for Enterprise Archive.

EJB See **Enterprise JavaBeans**.

EJB container
An application server container that implements the EJB component contract of the J2EE architecture. This contract specifies a runtime environment for enterprise beans that includes security, concurrency, life cycle management, transaction, deployment, and other services.

Enterprise JavaBeans (EJB)

1. A specification of Sun Microsystems, Inc., that is part of the WebSphere Application Server Advanced Edition. EJB support allows your application to include sophisticated business components that run on your server. These components may include business logic with automatic distributed transactions and persistence to a relational database.
2. A component architecture defined by Sun Microsystems for the development and deployment of object-oriented, distributed, enterprise-level applications. (Trademark of Sun Microsystems, Inc.)

Extensible Markup Language (XML)
A standard metalanguage for defining markup languages that was derived from and is a subset of SGML.

Host Access
The component of Host Publisher Studio with which the developer creates Integration Objects that collect data from applications on the terminal-oriented host. (See also **Application Integrator** and **Database Access**.)

Host Publisher Server
The collection of functions that run on the server—including Host Publisher Server Administration and functions associated with the execution of applications.

Host Publisher Server Administration
The collection of functions for configuring and monitoring Host Publisher Server.

Host Publisher Studio
The application-development environment of the Host Publisher product. It runs on a Windows workstation and consists of three components: **Host Access**, **Database Access**, and **Application Integrator**.

HTML
Hypertext Markup Language.

install To copy product code from its distribution medium (such as CD or Web download) and make it ready to use. Note that WebSphere documentation often uses the term *install* as a synonym for the **deploy** process.

Integration Object
Java beans created by Host Access or Database Access that encapsulate interactions with data sources like 3270 and 5250 applications and JDBC databases, and return desired data as Java bean properties. Integration Objects can be embedded within JavaServer Pages (JSPs), which allow users of a Web browser to interact with the data.

Interactive development environment (IDE)
A program that enables programmers to develop applications through a graphical user interface. WebSphere Studio tools, such as WebSphere Studio Application Developer, are examples of IDEs.

J2EE Java 2 Platform, Enterprise Edition. An environment for developing and deploying enterprise applications, defined by Sun Microsystems Inc. The J2EE platform consists of a set of services, application programming interfaces (APIs), and protocols that provide the functionality for developing multitiered, Web-based applications. (Trademark of Sun Microsystems, Inc.)

JavaBeans
In Java, a portable, platform-independent reusable component model. (Trademark of Sun Microsystems, Inc.)

Java Database Connectivity (JDBC)
An industry standard for database-independent connectivity between the Java platform and a wide range of databases. The JDBC interface provides a call-level API for SQL-based database access. (Trademark of Sun Microsystems, Inc.)

Java Naming and Directory Interface (JNDI)
In the WebSphere Studio products, an extension to the Java platform that provides Java applications with a standard interface to heterogeneous naming and directory services.

JavaServer Pages (JSP)
A server-side scripting technology that enables Java code to be dynamically embedded within Web pages (HTML files) and executed when the page is served, in order to return dynamic content to a client. (Trademark of Sun Microsystems, Inc.)

JDBC See **Java Database Connectivity**.

JNDI See **Java Naming and Directory Interface**.

JSP See **JavaServer Pages**.

JVM Java Virtual Machine. See **application server**.

macro An XML script that defines a set of screens. Each screen includes a description of the screen, the actions to perform for that screen, and the screen or screens that can be presented after the actions are performed. Host Access records macros within an Integration Object.

package
See **assemble**.

plug-in
A piece of code that interacts with an application server using a standard, server-specific programming interface.

publish
To transfer an application to the server and then deploy it. (See **transfer** and **deploy**.)

runtime
An instance of Host Publisher Server that provides middleware needed to execute J2EE applications developed with Host Publisher Studio. The runtime environment includes a Web-based administration function, Host Publisher Server Administration, for managing the middleware components.

Secure Sockets Layer (SSL)
A security protocol that provides communication privacy. SSL enables client/server applications to communicate in a way that is designed to prevent eavesdropping, tampering, and message forgery. SSL was developed by Netscape Communications Corp. and RSA Data Security, Inc.

servlet
An application program, written in the Java programming language, that is executed on an application server. A reference to a servlet appears in the markup for a Web page, in the same way that a reference to a graphics file appears. The application server executes the servlet and sends the results of the execution (if there are any) to the Web browser.

session
In network architecture, for the purpose of data communication between functional units, all the activities which take place during the establishment, maintenance, and release of the connection.

SGML
See **Standard Generalized Markup Language**.

Simple Object Access Protocol (SOAP)
A lightweight, XML-based protocol for exchanging information in a decentralized, distributed environment.

SOAP See **Simple Object Access Protocol**.

Standard Generalized Markup Language (SGML)
A syntax for markup languages that formalizes markup and frees it of system and processing dependencies.

Software Maintenance Utility
A tool in Host Publisher for applying and committing fixes (APARs).

SSL See **Secure Sockets Layer**.

StudioAppMigrator
A tool in Host Publisher Studio that migrates applications to a Host Publisher 4.0 level. It migrates the applications to JSP 1.1 and EJB 1.1 levels, inserts UTF-8 encoding, updates error pages, and moves application directories to *install_dir*\Studio\applications.

transfer
To copy an application .ear file to the server, usually by FTP. (See also **deploy** and **publish**.)

user The end user of an application that runs on Host Publisher Server. (Contrast with **administrator** and **developer**.)

UDDI See **Universal Description, Discovery, and Integration**.

Universal Description, Discovery, and Integration (UDDI)
A registry mechanism with which you can perform lookups for descriptions of Web Services.

user list
A list containing information about accounts (user IDs) that a Host Publisher application can use to access a host or database. User lists contain user IDs, passwords, and descriptions for the accounts.

UTF-8 Unicode Transformation Format, 8-bit encoding form, which is designed for ease of use with existing ASCII-based systems.

VT Virtual terminal.

Web container
A container that implements the Web component contract of the J2EE architecture.

Web Service
A self-contained, modular application 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.

WebSphere
A family of IBM software products that provide a development and deployment environment for basic Web publishing and for transaction-intensive, enterprise-scale e-business applications.

Note: In this book, **WebSphere** is often used as a shorthand way of referring to the **WebSphere Application Server** product.

WebSphere Application Server
An IBM software product that provides the core software needed to deploy, integrate and manage e-business applications. Host Publisher applications, when assembled and transferred to a server, run as WebSphere Application Server applications.

Note: In this book, **WebSphere** is often used as a shorthand way of referring to the **WebSphere Application Server** product.

XML See **Extensible Markup Language**.

XML Gateway
A Host Publisher function that provides an HTML emulator for end-user

access to 3270 and 5250 applications, and enables you to write a Java server program to access 3270 and 5250 application data in an XML format.

Appendix F. Notices

This information was developed for products and services offered in the U.S.A. IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

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
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:
INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "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.

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

Any references in this information 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 use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
TL3B/062
3039 Cornwallis Road
RTP, NC 27709-2195
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrates programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Programming interface information

This Administrator's and User's Guide contains information on intended programming interfaces that allow the customer to write programs to obtain the services of Host Publisher.

Trademarks

The following terms are trademarks of the IBM Corporation in the United States or other countries or both:

- AIX
- AS/400
- Client Access
- DB2 Universal Database
- IBM
- iSeries
- OS/2
- OS/390
- OS/400
- S/390
- RS/6000
- WebSphere

Other company, product, and service names may be trademarks or service marks of others.

Pentium is a trademark of Intel Corporation in the United States, other countries, or both. (For a complete list of Intel trademarks see http://www.intel.com/sites/corporate/tradmarx.htm.)

Lotus and Domino are trademarks or registered trademarks of Lotus Development Corporation.

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both. SPARC is a trademark of Sun Microsystems, Inc. in the United States, other countries, or both.

Microsoft, Windows, and Windows NT are trademarks or registered trademarks of Microsoft Corporation in the United States, other countries, or both.

Netscape is a registered trademark of Netscape Communications Corporation in the United States and other countries.

Oracle is a registered trademark of Oracle Corporation.

UNIX is a registered trademark in the United States and other countries licensed exclusively through The Open Group.

Index

Special Characters

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

R

S

T

U

V

W

X

Readers' Comments — We'd Like to Hear from You

IBM® WebSphere® Host Publisher
Administrator's and User's Guide
Version 4.0

Publication No. GC31-8728-03

Overall, how satisfied are you with the information in this book?

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Overall satisfaction	☐	☐	☐	☐	☐

How satisfied are you that the information in this book is:

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Accurate	☐	☐	☐	☐	☐
Complete	☐	☐	☐	☐	☐
Easy to find	☐	☐	☐	☐	☐
Easy to understand	☐	☐	☐	☐	☐
Well organized	☐	☐	☐	☐	☐
Applicable to your tasks	☐	☐	☐	☐	☐

Please tell us how we can improve this book:

Thank you for your responses. May we contact you? ☐ Yes ☐ No

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

Name _____ Address _____

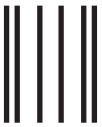
Company or Organization _____

Phone No. _____

Readers' Comments — We'd Like to Hear from You
GC31-8728-03

Cut or Fold
Along Line

Fold and Tape | **Please do not staple** | Fold and Tape



NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

IBM Corporation
Host Access Information Development
Department E40D/Building 502 Research
Triangle Park, NC 27709-9990

Fold and Tape | **Please do not staple** | Fold and Tape

Cut or Fold
Along Line

Part Number: CT0ZWNA

Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.

(1P) P/N: CT0ZWNA

GC31-8728-03