IBM® SecureWay® Host Publisher User's Guide

Version 2

GC31-8728-00

IBM® SecureWay® Host Publisher User's Guide

Version 2

GC31-8728-00

Note

Before using this information and the product it supports, be sure to read the general information under "Appendix D. Notices" on page 123.

1st Edition (September 1999)

Contents

Chapter 1. About Host Publisher

About this information

This information is designed to help you, the Web Application Developer, plan for, install, and start using IBM SecureWay Host Publisher. This book includes information about how to record interactions with data sources within an Integration Object, how to include those Integration Objects in Web pages, and how to publish those pages on the Web.

This book helps you get started quickly. Additional information resources are available for learning to use Host Publisher features. These resources include the product README, online help, and product Web pages. (See “For more information” on page 7).

Note: Consult the product README and the Host Publisher Web site, http://www.ibm.com/software/network/hostpublisher, for corrections and additions to this information.

This book is available in hard copy (in the Host Publisher packaging), as an HTML file on the installation CD, as a PDF file on the CD, and as an HTML file on the product Web site. Visit the Web site for the most updated version of this document.

What is Host Publisher?

Web-to-host integration is an integral part of any e-business. 70% of business-critical data and applications reside on IBM host systems, such as S/390, AS/400, 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 that enable you to provide access to data on a legacy data source (a host machine with a 3270 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.

Host Publisher enables you to create *Integration Objects* that contain logic to perform host data access and retrieval tasks. You can use and reuse these

objects in other Java applications, or you can publish the objects on the Web. When you publish objects on the Web, you enable other people to interact with the data. For example, if you have an existing host application that enables you to look up telephone numbers for employees, you can create a Web page that lets a user with a browser enter the name of the person they want, and then displays the telephone number on another page.

Host Publisher enables you to create Integration Objects for common tasks, such as connecting to a host, and reuse them. You can chain Integration Objects together so that the user can log in, complete several tasks, and then log off. You can also create composite applications that integrate multiple sources of data into a single Web page. To the end user, this Web page appears to be a single new application.

Host Publisher uses IBM WebSphere Application Server to provide a consistent, reliable deployment environment for Java applications and HTML/JSP pages across platforms. Applications created in Host Publisher Studio can be accessed with all standard Web browsers, with or without Java.

Host Publisher supports terminal-oriented applications that use 3270, 5250, VT data streams, Java, and databases that provide a JDBC interface, such as IBM DB2 Universal Database®, Oracle®, and Sybase™.

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 includes the IBM SecureWay Network Dispatcher, which provides for large enterprises the same level of load balancing and hot standby used in the Nagano Olympics.

Host Publisher is divided into two major components: Host Publisher Studio and Host Publisher Server. The Studio provides the development environment for creating Web applications. The Server provides the runtime environment for executing Web applications created with the Studio. You create Web-to-host applications using the Studio, publish them to the Server, and provide access to the end user. The Web-to-host applications you build contain Integration Objects. When used with the Server, Integration Objects can:

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

You can optimize connection establishment within each Integration Object. You can use Integration Objects as part of fully-customizable HTML pages or reuse them with other Java application programs.

Host Publisher supports object chaining, which 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 step you down to the point where you can list everyone in a particular department. 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 Integration Objects into steps so that the end user does not have to navigate back down through the several menus to reach the department list again.

To enhance performance, Host Publisher provides connection pools, which are defined in the Studio. Connection pools are used during runtime to cache ready connections to improve response time to Web pages. A user-defined number of connections will remain active in the pool for subsequent requests from any user. Connection pools eliminate the overhead of establishing new connections to the host or database for every Web page request.

Comparing Host Publisher and WebSphere

Although Host Publisher and WebSphere complement each other very well, and Host Publisher integrates WebSphere into its runtime environment, there is a fundamental difference between the primary use of each product.

Host Publisher delivers a quick and easy way for companies to implement e-business applications by extending *existing* applications to the Internet. It focuses on applications with little or no new logic. In contrast, WebSphere provides a robust Java infrastructure for the development and execution of Web applications and Servlets. WebSphere focuses on adding new logic to existing applications to make them accessible on the Web or deploying totally new Web applications such as those used in business reengineering.

The two products are complementary. Host Publisher uses the WebSphere application server environment to support applications that include Integration Objects created by Host Publisher. You can reuse Integration Objects within new Java applications, or you can use WebSphere and your favorite Java interactive development environment (IDE), such as Visual Age for Java, to add new logic to Host Publisher Web applications. For information about using Integration Objects in Java servlets and applications, see the Host Publisher Web page:
http://www.ibm.com/software/network/hostpublisher/library.

Host Publisher provides WebSphere Standard Edition. However, if you need or already use the advanced features of WebSphere Advanced Edition or

WebSphere Enterprise Edition, you can substitute those products to support the Host Publisher runtime environment.

Comparing Host Publisher and IBM SecureWay Host On-Demand

Host Publisher and Host On-Demand are designed for different end users. Host On-Demand is intended for users who are already familiar with host applications. Host Publisher is intended for users who are not.

Host Publisher is designed primarily for Internet 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. These users might not have Java-enabled browsers and therefore require HTML. As with Web self-service applications, these users typically connect infrequently and for short periods of time. They 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 Publisher can also be appropriate for extranet users and, to a lesser extent, intranet users where their usage and requirements are similar to the Internet user.

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, and fast response times are important to maximize productivity.

Host Publisher's advantages

Host Publisher is built on open-industry standards, such as Java and HTML. Integration Objects are reusable components that can be used in Java applications created outside Host Publisher. Likewise, interactive development environment (IDE) tools can be used 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 to meet your design guidelines and personal preferences.

Host Publisher Server provides enterprise-class performance, scalability and availability through several key features, such as chaining, connection pooling, load balancing, hot standby, and cross-platform portability. Since the Host Publisher Server runs on OS/390™, AIX™, Windows NT, and Solaris, applications created with the common Host Publisher Studio will run unchanged in all environments.

Host Publisher's load balancing capabilities enable you to balance the load of Integration Object requests over 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 allow you to move your workload to a higher capacity platform as demands increase.

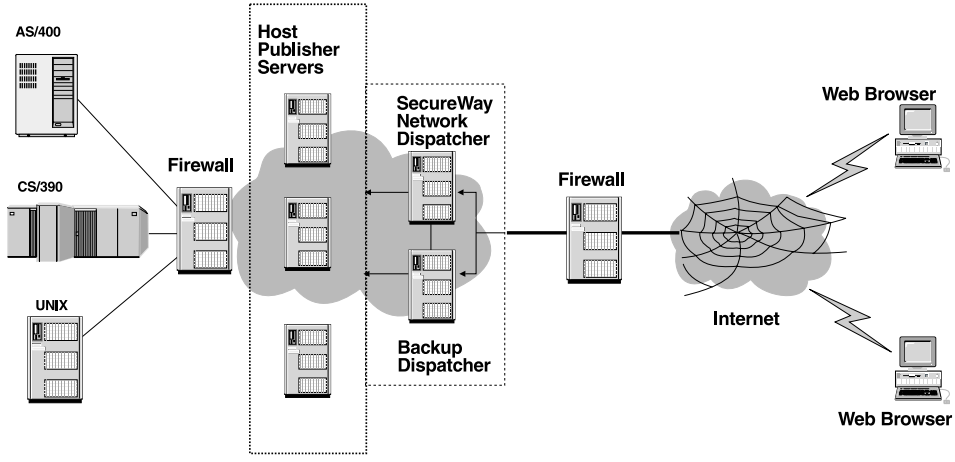

Figure 1. Host Publisher working in a network

About Host Publisher Studio

Host Publisher Studio is a collection of task-oriented, easy-to-use graphical user interfaces that assist the Web application builder in managing and creating Web-to-host publishing projects. 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 JavaBeans called Integration Objects, which encapsulate the interactions and data retrieval logic. You can use Host Publisher Studio to generate a fully customizable HTML Web page for modeling interaction with the Integration Objects and rendering the resulting data. You can enhance the generated HTML with your favorite Web authoring tool, such as NetObjects Fusion or Microsoft® Front Page™, to meet corporate guidelines on style and image. Once the Web page is completed, you publish it to a Host Publisher Server for production access by end users.

Host Publisher Studio runs on the Windows 95®, Windows 98, and Windows NT operating systems.

About Host Publisher Server

Host Publisher Server provides the run-time environment for supporting Web applications created with Host Publisher Studio. It consists of the IBM WebSphere Standard Edition Application Server and other run-time components such as connection management, license monitoring, run-time administration, and log and trace management.

Host Publisher Server is supported on OS/390, AIX, Windows NT, and Sun Solaris operating environments. It supports Web servers (through WebSphere) including Microsoft Internet Information Server, Lotus® Domino™ Go Webserver, Netscape®, Apache Power Web Server, and IBM HTTP Server.

What's new with Host Publisher Version 2?

Host Publisher Version 2 runs on more platforms and provides many new features that make it easier to use and broaden its function. New features include:

- Support for OS/390, AIX, and Solaris operating systems, in addition to Windows NT
- A separately-installable Host Publisher Studio, available on Windows 95, Windows 98, and Windows NT, with a friendlier, easier-to-use interface for building Integration Objects and Web pages
- Host Publisher-generated Integration Objects that you can publish to the Web or reuse within other Java application frameworks
- Use of IBM SecureWay WebSphere in Host Publisher Server, which provides synergy with an industry-leading Web application server
- Secure Sockets Layer (SSL) support for secure TN3270E and TN5250 communications with hosts and AS/400s
- Use of open standard formats: HTML, JavaBeans™, JavaScript™, JavaServer Pages™ (JSPs), and XML

Migrating from Host Publisher 1.0

Host Publisher Version 2 is very different from Host Publisher Version 1.0. Host Publisher 2 uses Host On-Demand for all screen recognition and keyboard macro logic. Because there are many changes and improvements in the way Host On-Demand functions in this area over Host Publisher 1.0, there is no automated way to migrate Host Publisher 1.0 Integration Objects to Host Publisher 2.

The simplest and most straightforward way to move from Host Publisher 1.0 to Host Publisher 2 is to rerecord your Integration Objects using the new Host

Publisher Studio. Contact your IBM representative to find out about services IBM provides to help customers migrate their applications from Host Publisher 1.0 to Version 2.

For more information

To access online documentation installed with Host Publisher, use a Web browser to open the following HTML file on your local system:

AIX	/usr/lpp/HostPublisher/common/doc/*lang*/guide.htm
Solaris	/usr/lpp/HostPublisher/common/doc/*lang*/guide.htm
S/390	/usr/lpp/HostPublisher/common/doc/*lang*/guide.htm
Windows NT	*install_dir*\HostPublisher\common\doc*lang*\guide.htm where *install_dir* is the directory in which Host Publisher is installed.

lang is the language-specific subdirectory for your language, and is one of the following:

de_DE	German
en_US	English
es_ES	Spanish
fr_FR	French
it_IT	Italian
ja_JP	Japanese
ko_KO	Korean
pt_BR	Brazilian Portuguese
zh_CN	Simplified Chinese
zh_TW	Traditional Chinese

Online help, including the HTML version of this book, is available from the product's user interface.

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/network/hostpublisher

Chapter 2. Using Host Publisher

You want to extend applications to the Web. Where do you start? This section helps you understand how to get started and how to accomplish some of the most common tasks you will complete.

Getting your information onto the Web using Host Publisher has four main steps. You need to:

1. Create Integration Objects that define the logic for accessing the information you want to publish. See "Creating an Integration Object for host access" on page 11 and "Creating an Integration Object for database access" on page 20.
2. Create Web pages that use the Integration Objects to generate Web content. See "Creating Web pages for Integration Objects" on page 21.
3. Put the Web application onto a Host Publisher Server. See "Transferring applications to a Host Publisher Server" on page 27.
4. Make the pages available over the Web. See "Using the Server" on page 28.

Overview

Host Publisher Studio enables you to create JavaBeans called *Integration Objects*. Integration Objects 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 or other Java applications. Integration Objects connect to the data source, navigate to the desired data, extract the data, and store them in Java properties, which can then be accessed.

Host Publisher also helps you generate special Web pages called JavaServer Pages (JSPs), that allow you to include Integration Objects right on the page, invoke them, and return their output for display on the page.

Integration Objects and the JSP pages that reference Integration Objects together form a Web application. Host Publisher Studio publishes the application to Host Publisher Server. WebSphere then parses the JSP pages and converts the special JSP tags on the pages into Java servlets. These servlets are Java programs that run under WebSphere. They are used to invoke the Integration Objects and display their output to an HTML page (derived from the original JSP). The Web browser displays this HTML page.

When WebSphere detects that an original JSP has been updated, the Java servlet is recreated; otherwise, JSPs are converted into servlets only once.

To use Host Publisher Studio and Host Publisher Server to build and publish a Web application:

1. Build an Integration Object in Host Publisher Studio to access a data source and return desired data.
2. Build Web pages around one or more Integration Objects to form a Web application.
3. Transfer the Web application to one or more Host Publisher Servers.
4. Using Host Publisher Server's administration facility, start the server and deploy the new application.
5. Use a standard Web browser to access the first page of the application.

About macros

Host Publisher records macros that contain information about the way you connect to the host and how you navigate to the information you want to publish. These macros become part of the Integration Object you create.

In Host Access, the Integration Object includes several types of macro:

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. Refer to "Using conditionals while recording" on page 15 for more information.

Data
: Includes the information about how to navigate to and extract 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.

Macros work by following a sequence of host screens that you define. When Host Publisher encounters a host screen, it performs the actions you have associated with that screen. You define the screens, the actions to take, and the next screen that should appear after the action has completed. It also allows you to define loops within the macro and the conditional paths 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.

Host Publisher uses IBM SecureWay Host On-Demand to provide an interface where you can interact with a host and record the actions you take. See “Appendix B. Macro script syntax” on page 79 for more information.

Using Host Publisher Studio

Host Publisher Studio has three parts: the Host Publisher Studio, Host Access, and Database Access. You can launch Host Access and Database Access from the Host Publisher Studio application.

Host Access and Database Access provide wizards and other tools that help you build Integration Objects. Once you create Integration Objects, you can use the Host Publisher Studio to create Java Server Pages (JSPs) that use them to generate dynamic Web content. You can also use the Integration Objects in other Java applications, or you can use Host Publisher Studio to publish other Java objects and beans.

The following sections provide instructions and information about how to complete common Host Publisher Studio tasks.

Creating an Integration Object for host access

To create Integration Objects that collect data from applications on the host, use the Host Access application in Host Publisher Studio. 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 information to help you with the following tasks:

- Using the wizard (“Using the wizard” on page 12)
- Defining host connections (“Defining host connections” on page 12)
- Creating connection pools for host access (“Creating connection pools for host access” on page 12)
- Specifying a host and a user ID (“Defining host connections” on page 12)
- Recording interactions with a host (“Recording interactions with a host” on page 13)
- Using conditionals while recording (“Using conditionals while recording” on page 15)
- Using looping while recording (“Using looping while recording” on page 16)
- Identifying screen data areas and labelling data (“Identifying information to use” on page 18)

- Replaying the interaction and inspecting the data (“Verifying a macro” on page 18)
- Generating an Integration Object (“Creating an Integration Object” on page 19)

Using the wizard

When you use Host Access to create an Integration Object, it launches a wizard that can guide you. To start Host Access, open the Host Publisher Studio application, then click **Create > Host Access Integration Object**. The wizard starts automatically. To use the wizard, provide the information it requests and click **Next** to continue until the wizard tells you the object has been created.

The wizard will ask 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 connect 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.

Defining host connections

Before you can record the steps Host Publisher will take to reach the data you want to publish, you need to define how to get to the application itself. 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 pool name, if any, to use.

Creating connection pools for host access

By default, Host Publisher uses *nonpooled* connections. When a user requests information from a Host Access Integration Object on a Web page, Host Publisher connects to the host, logs on, extracts the information, and logs off. To reduce the time it takes to return information, you can specify that Host Publisher should use *pooled* connections. The second time a user requests information, Host Publisher can use an established host connection to extract the data. You can use the Connection Pools tab in Host Access to define pools of connections. You can also use this tab to modify attributes of these 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.

Modify attributes of the connection pool to specify how Host Publisher should handle multiple connection requests.

To create a pool of connections, click the Connection Pools tab and check **Enable connection pooling** on the Connection Pool panel. The number of connections in the pool are defined in the Connection Limits section of this panel. To add more user IDs and passwords to your connection pool, follow the instructions for adding additional user IDs and passwords.

When creating an Integration Object for host access, you can also define host connection pools. To define a new connection pool:

1. Select **Create a new pool** from the Host Configuration panel.
2. Provide a new pool name.
3. Define the connection configuration to use for the pool.

Connection configurations define a particular host and its connection parameters.

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. If you want to use a previously-defined connection configuration:

1. Select **Share an existing connection configuration**.
2. Select a name from the list.
3. If you are creating a new connection configuration, provide the information to establish a connection to the specified host.
4. Click Next to create the connection pool with a set of default values. To view or change these values, click the Connection Pool tab in Host Access.

From the Connection Pool tab you can access all connection pools created in Host Publisher Studio. 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
- Change connection configuration information
- Add users

When you finish customizing your connection pool, click the Macros tab to continue recording your Integration Object.

Recording interactions with a host

When you use the wizard, it guides you as you interact with the host to navigate to the screen that contains the data you want to publish, specify the specific data, and define the way it should be presented. You use a terminal

connection just as you normally would, and Host Publisher records the interactions as a macro. The macro is displayed in the left pane in the window in a tree view.

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

1. Record interactions without using the wizard
2. Define an entry field
3. Add conditional paths or looping to the macros you create

You can use the shortcut menu to modify a specific step in the macro by selecting a step in the tree and right-clicking on it.

After you import a completed Integration Object into an application and publish it, Host Publisher will follow the steps you followed to access the data that will be returned to the Web browser.

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”, “Using conditionals while recording” on page 15, and “Using looping while recording” on page 16, 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 **Enter input variable** button on the toolbar.
3. On the window that appears, type a name for the variable and the default 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 `Lewis, Sera` for the value of the input variable to be used during macro recording. The value you provide is used only in the current recording session and does not become part of the macro.

Using conditionals while recording: Conditionals enable you to handle the situation where a host application could respond to a command (or keystrokes) with more than one screen.

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

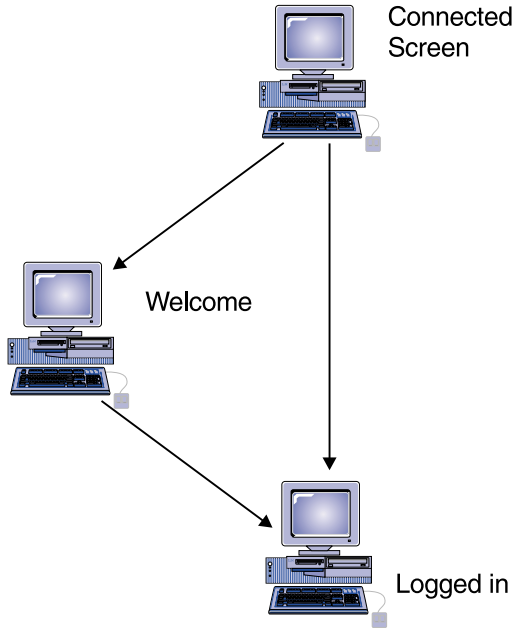

Figure 2. Host Publisher conditionals

To insert a conditional in a macro:

1. Record one version of the macro. For example, record the macro assuming that the logged in screen appears.
2. Replay the macro and cause the different condition to occur. This may involve specifying different values for input prompts, or connect to the session in a different state.
3. Host Access will report that a different-than-expected screen was encountered. 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. You might have several screens to define before you return to the macro, or you might not return to the main path at all.

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.

Note that you must have already defined the screens before you can specify them and that you can have several conditions for one step of a macro. Each screen that you define has actions associated with it. When Host Access encounters one of the possible screens you specify, it performs the actions associated with that screen.

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

Using looping while recording: Looping enables you to define an action that should be repeated until a condition is met.

For example, Figure 3 on page 17 shows an example of a possible loop in a data macro. When the user requests data, the connection might return one or several screens of information. You want to display all the information, so you need to define a way for the macro to display all the screens and recognize the last screen of data. You can use **Start loop** to define how Host Publisher handles either condition.

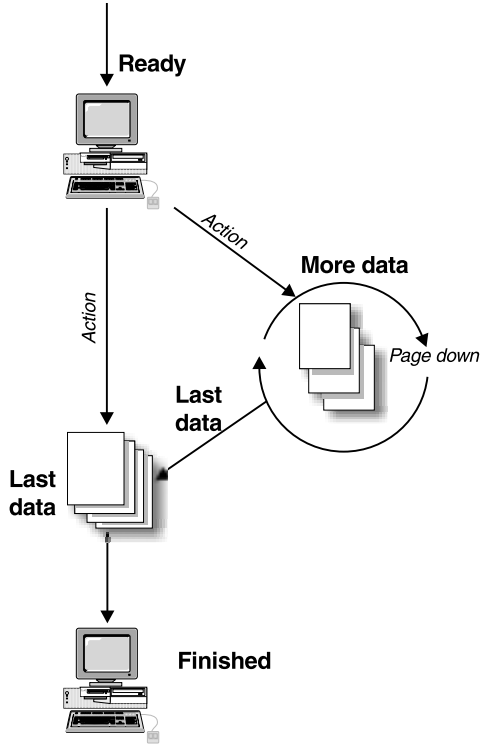

Figure 3. Host Publisher looping

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 will perform to record the loop. Click **Next** to begin.
4. Follow the wizard instructions. Navigate the terminal to the screen that contains the data to extract.
5. Follow the instructions in the Recording Loop wizard to identify the data to extract and its format. Click **Next** to continue.
6. Type the keys that will cause the terminal to advance to the second screen in the loop. The keys you type will be repeated continuously until the ending loop condition is met. If you performed a data extraction, the data will be accumulated as the loop repeats. Click **Next**.
7. Host Access stops recording your actions. Use the host application to navigate to the last screen in the loop. When the screen appears that indicates that the loop has finished, click **Next**.

8. Host Access starts recording your actions again. Define the current screen displayed in the terminal. In your definition, identify something on the screen that 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.
9. Press **Next**. The Recording Loop wizard ends and the loop is complete.

There are several ways to start recording loops. You could record a loop until there is an error and then record the way to recover from the error. You can cause new screens to appear when you are working with input variables by playing the macro and entering different strings.

Identifying information to use

When you use the Host Access wizard to create a macro, it helps you navigate to the data you want, identify the data, and structure the data for publication.

The wizard first guides you through defining a host connection, then starts a terminal connection with the host you defined. The wizard will ask you to use the terminal to navigate to the desired data, and will guide you through selecting the data for extraction. You can:

- Select a table of data
- Select paragraphs of data
- Select multiple regions on a screen by choosing **Capture More Data**

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

After Host Access guides you through creating an Integration Object, you can use Host Publisher Studio to create a Web page where the data will appear.

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. You can select all the information you want, arrange it, and display it to the user as though it were one piece of data.

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 macro displayed in the tree under the Macros tab.
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, inspect the captured data, and 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).

About global screens

Global screens handle application screens that might appear at any time during the execution your macro. When the same action is required each time such a screen is encountered, you can use global screens. During the use of a macro, each *next* screen is checked sequentially in the order it is listed. (Each step in a macro has at least one screen defined as the possible next screen; Host Publisher waits for the expected screen to appear before playing the next step of the macro.) 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. The next screens are examined again in the order they are listed after the global screen actions have executed.

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

Creating an Integration Object

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

You are prompted to name your Integration Object. Use a unique name for each object to avoid overwriting existing objects on the Server. If you publish an application to the Server that contains an object with the same name as an

existing object, the existing object will be replaced. The application that uses the existing object will then use the object you published in place of the original object.

Creating an Integration Object for database access

You use the Database Access application to create Integration Objects that encapsulate a database transaction. To create the Integration Objects, you query a relational database and retrieve data from the tables in the database using structured query language (SQL) statements.

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

- Using the wizard ("Using the wizard")
- Recording transactions with a database ("Recording transactions with a database")
- Retrieving information from a database ("Retrieving information from a database")
- Generating an Integration Object ("Generating an Integration Object" on page 21)
- Creating connection pools for Database Access ("Creating connection pools for Database Access" on page 21)

Using the wizard

When you use Database Access to create an Integration Object, it launches a wizard to guide you. Tabs guide you through the process of building and executing a valid SQL statement. You can navigate by clicking **Back** or **Next** at the bottom of the panel. Provide the information it requests for each tab. When you complete the wizard, the object is created when you save the file.

Recording transactions with a database

When you use the wizard, it guides you to specify the JDBC URL for the database you want to access. Using the tabs, you create an SQL statement to retrieve data from the tables in the database.

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.

The wizard first guides you through defining a database connection. Then, using the tabs, 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.

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

Generating an Integration Object

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

After you import a completed Integration Object into an application and publish it, Host Publisher will follow the steps you followed to access the data that will be returned to the Web browser.

Creating connection pools for Database Access

When creating an Integration Object for database access, you can also define database connection pools. You can create a new connection pool or select a previously-created pool using the Connection Pools tab. To create a new connection pool:

1. Select **Create new connection pool**.
2. Provide a new pool name.

By default, the data you provided by connecting to the current database is copied into the Connection Configuration section.

Select a previously-defined configuration to share Connection Configurations between connection pools. The list of pool users can also be shared between pools, or you can create a new list of users.

Creating Web pages for Integration Objects

You use the Host Publisher Studio to create Web pages that access the Integration Objects you created using Host Access or Database Access. Host Publisher helps you design a page and integrates the objects for you. You can also use Host Publisher to import other Java beans or objects.

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

- Using wizards (“Using wizards” on page 22)
- Specifying Integration Objects to publish on the Web (“Specifying Integration Objects to publish on the Web” on page 22)
- Choosing properties to render and control appearance (“Choosing properties to control appearance” on page 22)
- Previewing a page (“Previewing a page” on page 23)

Using wizards

When you use Host Publisher Studio, you can build your application using wizards to guide you. The wizards guide you through naming the new application, importing Integration Objects into the application, and creating pages for publishing Integration Object data. You can create an entire working project using the new application wizard. To build your application using the wizards, select the new application item on the **File** menu on the main panel.

If you prefer to create your application manually, you can bypass the wizard and use the menus. Some of the items on the menus also launch wizards to help you accomplish some of the tasks.

Specifying Integration Objects to publish on the Web

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

The Host Publisher Studio enables you to build a series of Web pages using standard HTML tags in conjunction with special JavaServer Pages (JSP) tags. 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 the Host Publisher Studio to import other Java classes or beans, in addition to Host Publisher Integration Objects, into your application project and publish them into Web pages.

The Host Publisher Studio 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 (HTML and JSP), which have been generated or modified to interact with Integration Objects.

Choosing properties to control appearance

Integration Objects have properties that return data to the user. The output data can be embedded within graphic controls. Host Publisher will format table, list box, text entry, and text area controls for you.

Satisfying Integration Object input

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

Previewing a page

When you are creating an HTML page in the Host Publisher Studio, you can preview the page with a Web browser to verify the layout. The layout of the page is displayed, but the JSP tags and associated Java objects are not shown.

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 described in the following sections.

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 using the Host Publisher Studio wizards:

1. Create a new application.
2. Import the Integration Objects into the project.
3. When defining how the Integration Objects are used, specify that they are to be added to the same execution, or 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 select Insert > Output Control > Table from the menu bar. Another wizard guides you through the process of creating a table, including selecting the Integration Object outputs to use to fill the table with data.

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 page. Be sure that the Integration Objects appear on the page in the correct order.

To create this type of composite application using the Host Publisher Studio wizards:

1. Create a new application.
2. Import the Integration Objects into the project.
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 only accept single-valued inputs, so you can only use 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, allowing 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, allowing 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 using the Host Publisher Studio wizards:

1. Create a new application.
2. Import the Integration Objects into the project.
3. For the first Integration Object, specify that it should be placed on execution 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 (since you can select an item out of the list), a form is defined for it on output1. Do no specify a page for this form's destination.
5. For the second Integration Object, specify that it should be placed on execution 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 execution 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 24, but the Integration Objects reside on different pages. 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 using the Host Publisher Studio wizards:

1. Create a new application.
2. Import the Integration Objects into the project.
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. Do not render any of this Integration Object’s output.
4. For the second Integration Object, specify that it should be executed on page execute2. Satisfy its inputs from the outputs of the first Integration Object. You can render the output of the Integration Object on this same page (execute2).

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 must ensure that execute2 occurs after execute1 in your page hierarchy.

Creating user lists

User lists contain user IDs, passwords, and other information that are associated with host accounts.

Imagine that you are creating an application that connects to a host and has user accounts it can access to connect. The accounts have user IDs and passwords associated with them.

Host Publisher runs the macro you've recorded and tries to connect to the host. If you specified that you want the macro to use connection pooling, Host Publisher uses the list you defined to supply the values that it passes on to the host. If the first ID in the list is in use, Host Publisher uses the next 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**
sera	serapw
sarkar	sarkarpw
villari	villaripw

If **sera** is in use, Host Publisher uses **sarkar**. If **sera** then becomes available, Host Publisher uses **sera** for the next connection. If **sera** and **sarkar** are both in use, Host Publisher uses **villari**.

Importing Java objects

You can import Java objects into a Host Publisher Studio application. The Java objects can be Java classes, beans, or Host Publisher Integration Objects. You can import Java Class files, ZIP files, or JAR files. ZIP or JAR 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 access applications, such as the Database Access application or Host Access application, the Host Publisher Studio knows the inputs, outputs, and execution methods.

If you import a Java class or bean into the application project that is not a Host Publisher Integration Object, you verify what the inputs, outputs, and execution methods should be.

Transferring applications to a Host Publisher Server

The Host Publisher Studio gives you the ability to transfer your Web application to Host Publisher servers, making the application ready for deployment.

When you transfer your application, all application parts are collected and organized in a directory structure similar to the structure on a Host Publisher server. The directory structure contains the following subdirectory types:

Shared
Contains the parts of the application, such as Integration Objects and connection configuration files required by the Integration Objects, that can be shared with other applications.

Application-specific
Contains the application pages. The name of this subdirectory is the same as the application name.

The wizard that guides you through the transfer process asks you how to secure potentially sensitive user data, such as passwords. You can choose to:

- Not secure the data, leaving the values readable from the configuration files
- Scramble the data
- Encrypt the data

Encryption requires that you specify an encryption key.

Once the application is organized into a staging directory on the Host Publisher Studio machine, the file transfer protocol (FTP) is used to transfer the contents of the application to the specified Host Publisher Servers.

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:

Host Publisher Studio	webbridge.trc
Host Access	hostaccess.trc
Database Access	dbaccess.trc

The trace files will be located in the Studio directory.

Using the Server

Once you have published pages to the Host Publisher Server, the server administrator can use the administration interface to manage the way connections and applications are handled.

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

- Using Host Publisher Server Administration (“Using Host Publisher Server Administration” on page 29)

- Deploying applications (“Deploying applications”)
- Managing licenses (“Managing licenses” on page 30)
- Logging and tracing components (“Logging and tracing components” on page 30)
- Balancing the server’s load (“Balancing the server’s load” on page 33)
- Application administration (“Application administration” on page 35)

Using Host Publisher Server Administration

The Host Publisher Server provides the runtime environment for supporting Web applications created with the Host Publisher Studio. The runtime environment includes Web-based server administration for controlling the runtime and the applications it serves. To access Server Administration, load this URL in your browser: **http://***server*/**HostPublisher/HPAdmin/main.jsp** (where *server* is your server name).

Deploying applications

Host Publisher deploys applications that have been transferred to the server into production.

By default, applications must be deployed manually. If you want an application automatically deployed when the server starts, you must change the server.properties file. See “Editing the server.properties file” on page 73 for instructions on editing the properties file.

Some examples of problems are: errors in connection pool configuration files, missing files, conflicting information, or applications already deployed.

To deploy an application:

1. Start Host Publisher Server Administration.
2. In the navigation area on the left, click Administration > Application Administration > Deploy.
3. Check the boxes next to the applications you want to deploy.
4. Click Deploy.

Once you choose the applications you want to deploy, Host Publisher Server scans the staging area for the application files and, if no problems are found, moves the files over to the production area for immediate use. The staging area is then cleared.

If one or more of the applications selected for deployment contain strongly-encrypted user pools, you must enter the Host Publisher encryption key in the entry field provided. If you do not, only selected applications that do *not* contain strongly-encrypted user pools are deployed; however, you will

receive a message informing you that one or more applications were not deployed because they require the encryption key.

Notes:

1. If a previous version of an application, or another application with the same name, is currently deployed, deploying a new version may overwrite existing files. Next to each application there is a column that indicates whether an application of the same name is currently deployed.
2. If you deploy an application that has already been deployed, WebSphere will stop and restart all servlets. This might cause a disruption in service.

Managing licenses

The Host Publisher Server tracks the number of requests per minute and automatically logs a message when the value exceeds the number of licenses purchased. One license is equivalent to the right to execute one Integration Object per minute.

Host Publisher Server can optionally track license usage history over time. This history maintains the maximum number of licenses used (or Integration Objects invoked) during a one-hour period, logging this information to a file each hour. By default, this option is disabled.

To enable the license tracking option, set the licenseTracking property in the server.properties file from 0 to 1. This file is located in the Server subdirectory under your Host Publisher installation directory.

For detailed information about editing server.properties, see “Editing the server.properties file” on page 73.

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

1. Start Host Publisher Server Administration.
2. In the navigation area on the left, click Administration > License Management.
3. Type the new value, then click Set.

Logging and tracing components

The Host Publisher Server produces messages.txt and trace.txt files to help you in problem determination.

The log file (messages.txt)

The Host Publisher 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 may 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 this book and contain suggestions for actions you can take to correct problems, when necessary. (See"Appendix C. Error Messages and Recovery Actions" on page 95.)

The trace file (trace.txt)

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.

Controlling logging and tracing

The Problem Determination panels in Host Publisher Server Administration enable you to control tracing and logging. Using these panels, you can perform the following tasks:

View Log
View the last 200 lines of the log file, download the entire file, or clear the file.

Note: If you click Clear, all the current information is deleted from the trace file.

Set Log Options
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.

Note: Error events are always written to the log file.

View Trace
View the last 200 lines of the trace file, download the entire file, or clear the file.

Note: If you click Clear, all the current information is deleted from the log file.

Trace File Name

Define the file to which trace events are written. If the file already exists, new events are appended to it.

Host Connection Tracing

Control the tracing levels of None, Minimum, Normal, or Maximum, and select one of three types of trace for host connections:

Macro tracing

Traces the playing of macros

Presentation space tracing

Traces the internal display screen

Transport tracing

Traces the network connection

Control whether display terminal screens are created on the Server display (except OS/390)

Display screens are created only for connections established while this option is turned on. They are not created for existing connections, or for pool connections that are idle.

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.

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 while 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.

Server tracing

Trace events on the Server. This panel contains two groups of options:

Trace Types

API Tracing
Traces calls in and out of other components, such as Host On-Demand and JDBC.

Entry/exit Tracing
Traces internal calls within the server. This is where most tracing occurs.

Miscellaneous Tracing
Traces events not covered by the other two trace types.

Trace Sources

Server Tracing
Controls tracing of the core of the Server. This is the part the manages connections, administration, and so forth.

Integration Object Tracing
Enables tracing in specific Integration Objects that you have built.

Note: You **must** select one or more options from each group to enable any Server tracing. For example, to enable all trace types in Host Publisher Server, select API, Entry/Exit, and Miscellaneous trace types, then select Server trace source.

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 the server.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 server.properties, and for information about enabling multiple log or trace files, see “Editing the server.properties file” on page 73.

Balancing the server’s load

Host Publisher Version 2 includes IBM SecureWay Network Dispatcher, which enables you to balance client request loads across multiple Host Publisher Servers. Network Dispatcher forwards page requests from clients to available

Host Publisher Servers in a round-robin fashion, meaning that each Host Publisher Server is dispatched a client request in turn. Clients include the Network Dispatcher's host name in the URL request for the initial Web application page. Network Dispatcher then replaces its own host name with the appropriate destination Host Publisher Server's host name in the URL and forwards the request to that server.

Since most Web applications include more than one page and at least one Integration Object, it is important that Network Dispatcher ensure a particular client's ability to access the same Host Publisher Server for a period of time. This is because Web applications can introduce the concept of a *state*, where a second page assumes specific information was entered on the first page. You would probably not want to be routed to Host Publisher Server A for the first page request, then to server B for the second page request; server B does not know what Server A provided on the page.

Maintaining the relationship between client and server for the duration of the Web application is called *session affinity*. Network Dispatcher achieves session affinity by enabling you to set *port stickytime*. This is a period of time in which a client is always routed to the same server. The timer starts on the first request and restarts on each subsequent request. After the timer expires (no request is received from the client during that time), the next client request will not necessarily be routed to the same server.

When using Network Dispatcher to balance request loads, you probably want to complete the Web application and not allow a long period of time to pass between page requests to avoid disruption of the application, resulting in lost information. Therefore, the administrator should configure port stickytime value large enough to allow for long Web applications or possible pauses or delays between page requests.

IBM SecureWay Network Dispatcher does not necessarily require an additional server machine; however, we recommend that Network Dispatcher not be installed on the same machine as Host Publisher, as it will require resources that might better serve Host Publisher, such as memory and processor time.

For instructions on installing and setting up Network Dispatcher, refer to the User's Guide on the IBM SecureWay Network Dispatcher CD in the documentation*LANG*\htm\ directory (where *LANG* is the appropriate language). Refer to the section on **Server Directed Affinity API** for information on how to set the appropriate port stickytime for session affinity.

Application administration

Open Host Publisher Server Administration and click Administration > Application Administration to:

List Applications

View a list of all the applications on the Server, both in the staging and production areas. These applications are identified as Deployed and Not Deployed. An application can appear as both Deployed and Not Deployed if a new version of a deployed application has been transferred to the server, but not yet deployed.

Deploy Applications

View a list of all applications transferred to the server and select one or more for deployment. The deployed applications move to the Server's production area and become available for use. If any of the applications you want to deploy have user pools that require strong encryption, you must enter the Host Publisher encryption key to deploy these applications. This key is defined in the Host Publisher Studio and allows Host Publisher Server to encrypt and decrypt data.

Changes to existing pools or connections will not take place until the Host Publisher Server restarts. When the server is running, you can deploy applications that use newly-defined connection pools, and the application will be activated immediately.

Delete Applications

View a list of all deployed applications, and select one or more of them to delete from the Server. You must stop the server before you can delete applications. If you delete an application accidentally, you cannot undo the deletion; however, if the application is still available in the Host Publisher Studio, you can publish the application again.

Delete Unused Files

View files in the production area of the Server that do not belong to any application, and select one or more of them for deletion. If these files belonged to an older version of an application, you can safely delete them. If, however, they were manually placed on the Server (not using transfer), it may not be safe to delete them. Therefore, carefully check that the files are not in use before you attempt to delete them.

You cannot restore files you delete using this panel.

Chapter 3. Advanced features

This chapter discusses how to use Host Publisher's advanced features: object chaining and security.

Using Integration Object chaining

Integration Object chaining enables multiple Integration Objects to execute in sequence, each using the same connection. Object chaining only refers to Host Access Integration Objects, which have connections to a screen based application, such as a 3270 application. An Integration Object in an object chain leaves the connection in a state (at a particular screen) suitable for the next Integration Object in the chain to use. Object chaining requires Host Publisher Server and is not supported when invoking Integration Objects from custom Java applications.

You can use chaining to break up a complex application into multiple tasks, each task represented by an Integration Object. Host Publisher Studio ensures that the order in which Integration Objects are invoked is correct; however, if you hand-edit the jsp files, the order of the Integration Objects is very important.

For example, if you have three Integration Objects in a object 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 in a specific state, the connection will not be available and the Integration Object will fail. Host Publisher Studio ensures the order for you;

A particular state within a object chain is identified by the current screen of the connection and the browser currently using that connection. Therefore, when a browser requests a page which accesses the next Integration Object in an object chain, that Integration Object is invoked using the connection reserved for that client in the specific state. A browser can use several object chains concurrently, but two browsers cannot use the same instance of an object chain. Only the connection that invoked the first object in a chain can invoke the next. This prevents other users from inadvertently picking up where your application has left off.

There are three types of Integration Objects in an object chain: first, middle, and last. When you create the first Integration Object, enable connection pooling, then record the connect and disconnect macros, which are part of the pool. When you create the middle and last Integration Object, the full connect

and disconnect macros are displayed in the macro tree; however they don't necessarily run for that object. The connect macro runs before the first object and disconnect runs after the last object.

To build an object chain within Host Publisher Studio, you must first build the Integration Objects within the Host Access application, and then import these Integration Objects into Host Publisher Studio and build Web pages around them.

To build the first Integration Object in the object chain:

1. Use the wizard within the Host Access application to define a connection as you normally would (see "Using the wizard" on page 12 for information about defining connections).
2. Select **Create a New Pool** on the Host Configuration panel.
3. Record your connect macro and your data macro (see "Recording interactions with a host" on page 13 for information).
4. You will probably not end your data loop 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 loop macro, and click **Stop Recording** on the toolbar.
5. Navigate back to the point where you can disconnect from the host.
6. Highlight the disconnect macro node in the macro tree, and click **Record**.
7. Record your disconnect macro.
8. Before you start building the Integration Object, select Options > Configure Object Chaining. On the window that appears, specify a stop state label for this Integration Object and specify that this object should be **first**. Host Publisher uses start and stop state labels to identify the Integration Objects that 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.

To record a middle Integration Object in the chain:

1. Use the wizard within 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 object chain must use the same connection configuration.
2. Optionally, play the connect macro, or connect manually.
3. Use the terminal to navigate to the correct starting point for this Integration Object, and record the data loop macro.
4. Select Data Macro in the tree, then click Record.

5. Select Options > Configure Object Chaining. On the window that appears, 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 object should be **middle**.
6. Save the Integration Object.

To record the last Integration Object in the chain:

1. Use the wizard within 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 object chain must use the same connection configuration.
2. Optionally, play the connect macro, or connect manually.
3. Navigate to the correct starting point for this Integration Object, record the data loop macro, and be sure to end at the same point where the first Integration Object in the chain started.
4. Select Options > Configure Object Chaining. On the window that appears, 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 object should be **last**.

   Note: For the last in the chain, you will supply only the start state label.
5. Save the Integration Object.

The last Integration Object in your object chain should return the host screen to the same state from which the first Integration Object started. Once you have all of your Integration Objects defined, return to the Host Publisher Studio, and import them into a new application.

In Host Publisher Studio, the Available Objects tree, when expanded for an Integration Object, also specifies the logical next and previous Integration Objects for the selected object. This makes it easier for you to identify the order in which the Integration Objects can be used.

Within Host Publisher Studio, you can build Web pages to form a sequential chain, where one page leads to the next after an input form is submitted (you can also build a single page that contains an entire chain). As you build a chain of pages, you can add the Integration Objects to the pages only in the logical order in which they were created for the object chain.

For example, if you have three Integration Objects in a object chain: A, B, and C, then you must use A first, then B, then C. The Host Publisher Studio wizards that guide you through building the application work the same way

as for normal Integration Objects, but are more restrictive in how object chained Integration Objects can be used within the application project.

When you have finished building your object chain application, transfer it to a server, deploy the application, and test it (see "Transferring applications to a Host Publisher Server" on page 27 for information).

Configuring and using Secure Sockets Layer (SSL) support

Host Publisher uses Host On-Demand to provide connection support to 3270, 5250, and VT applications using Telnet protocols. Host On-Demand also provides SSL support for securing these connections. A secure connection over SSL causes data flowing over the connection to be encrypted and therefore protected 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.

Host Publisher will verify that the certificate is signed by a well-known certificate authority. If it is not, Host Publisher is required to have its own version of the server's certificate for verification purposes. (See information about self-signed certificates in the following paragraphs.)

If you also select the server authentication option when configuring SSL, in addition to verifying the certificate itself, Host Publisher will perform 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.

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.

To use self-signed certificates, you must embed them within a Java class file and insert them into Host Publisher's CLASSPATH. Host Publisher Studio uses this class file to establish a secure connection while the Integration Object is being built. Host Publisher Server uses this file to establish the secure connection when the Integration Object is invoked. Host On-Demand provides a utility, included with Host Publisher Studio, to convert the certificate file to a Java class file. To use it, type the following command:

c:\HostPublisher\Studio\gencert.bat *certificate_file*

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

The result will be a .class file called CustomizedCAs.class, which you need to keep on the Studio and copy onto the Server. On the Studio, copy this file into the Studio subdirectory (c:\HostPublisher\Studio, for example). On the Server, copy this class file into the Host Publisher Server's common directory, which should be in WebSphere's CLASSPATH. You can place it anywhere in the CLASSPATH, so long as WebSphere picks up the .class file when processing a request for an Integration Object configured for SSL support.

Securing access to Host Publisher Administration

You can control access to your system resources by managing users and groups and associated permissions.

To enable WebSphere user ID and password security:

1. Load the WebSphere Application Server in your browser (http://*yourlocalhost*:9527).
2. Click the triangle next to **Security** in the navigation frame.
3. Click Users to add and delete users and control access permissions.
   a. To add a user, click Add, then type the user name and associated password in the Add User dialog box.
   b. To delete a user, highlight the name in the Defined Users list, then click Remove.
   c. To change a password, highlight the associated user name, then click Password. Type the new password in the Change Password dialog box.
4. To efficiently manage a large number of users, create a group. Click Groups in the navigation frame.
   a. To add a group, click Add, then type the name of the group in the Add Group dialog box.
   b. To add users to a group, highlight the user name in the Non-Members list, then click Add.

c. To remove users from a group, highlight the user name in the Members list, then click Remove.

d. To remove a group, highlight the group name in the Defined Groups list, then click Remove.

5. To control who can access your Web server resources, click Access Control Lists in the navigation frame

   a. To add an access control list (ACL) to the defaultRealm, click Add, then type the new ACL name in the ACL dialog box and click Add.

      Note: The default realm setting is **defaultRealm**.

   b. Highlight the new ACL in the Defined Access Control Lists, then click Add next to the bottom box.

   c. Make sure **Assign Permission for:** is set on **Files and Folders**.

   d. Highlight your user group.

   e. In the check boxes at the bottom of the box, check Get, Put, and Post.

   f. Click OK.

   .

6. To protect resources by establishing an access control list for each resource, click Resources in the navigation frame.

   a. To add a resource to the defaultRealm, click Add. (The default realm setting is **defaultRealm**.)

   b. In the Protect a Resource dialog box:

      1) Make sure Scheme is set to Basic.

      2) Select the appropriate ACL.

      3) Click the Servlet option, then select HPAdmin from the pull-down list.

      4) Click OK.

After you complete these steps and load http://*yourlocalhost*/HPAdmin/main.jsp in your browser, a logon prompt will appear. To run the servlet, type the user ID and password that you created using the previous procedure.

Chapter 4. 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. Determine whether the error occurred in the 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. Refer to “Common problems” on page 44. If the symptoms of your problem are listed, follow the recommended actions to resolve the problem.
4. Refer to the Host Publisher service page at:

   `http://www.ibm.com/software/network/hostpublisher/service`

   for hints and tips and fixes.
5. Check the Host Publisher logs for error messages that indicate the cause of the problem. Refer to “Logging and tracing components” on page 30 for information about accessing the Host Publisher Server log. Host Publisher Studio error output appears in the console window for the Studio application you are using. Refer to “Enabling tracing” on page 28 for information about enabling tracing in the Host Publisher Studio. If output has been scrolled off the console screen, open your own console screen and increase the buffer length using the console properties. Start the application from the console screen using the batch file (.bat) file for the application located in the Studio directory of the Host Publisher installation directory.
6. If the problem still appears when starting WebSphere or the Host Publisher Server or when accessing a Host Publisher application, check the WebSphere error logs for information about the problem. For Host Publisher applications, also check the Host Publisher error logs. Refer to “Gathering WebSphere and Web server problem data” on page 49.
7. 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 application being run.

8. Contact IBM service to resolve the problem. Refer to “Contacting IBM for service” on page 50.

Common problems

The following sections document common problems found while using Host Publisher. For each symptom, the probable cause and suggested action is documented.

Making WebSphere handle 50 or more requests

Cause: The WebSphere plugin has a hardcoded thread pool size of 50; therefore, even if your Web server has 200 threads and you send it 200 concurrent HTTP requests, WebSphere will only process 50 of them and queue the other 150.

Solution: You can increase the thread pool size by editing a property setting in the WebSphere configuration file bootstrap.properties. Add the following statement, where **200** is the number of desired requests.

```
ose.outofproc.threadpool.size=200
```

Note: Communication between the Web server and the plugin is handled by a named pipe. On Windows NT, there is a Win32–specific limit of approximately 128 sockets per named pipe. Because of this, even if you raise the thread pool size, you might encounter other operating system-specific limits and WebSphere might not process the expected number of requests.

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

This problem has two possible causes:

Cause: DB2 was not set up correctly.

Solution: Ensure that **SQLLIB\bin** is defined in your system search path.

Cause: WebSphere is configured to load the debug Java JVM. The JVM searches for debug libraries owned by the DB2 JDBC driver in the system, but the libraries do not exist in a normal DB2 installation.

Solution: Modify the WebSphere bootstrap.properties file under the properties subdirectory of the WebSphere installation directory. Set the java.debug property to **false**.

Macros fail in Host Access Integration Objects

Cause: 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 to be able to process the content of the screens they encounter, but also be precise in identifying the screens processed.

Solution: Some tips for recognizing a screen in a macro are:

1. Never recognize a screen by text that could change over time, such as a connection identifier, date, time stamp, or user ID.
2. Recognize a screen from minimal text on the screen, simplifying the recognition process and reducing the possibility for error.
3. Recognize a unique string located anywhere on the screen, eliminating the possibility of the string being in a different position.
4. Most screens are drawn in sections, depending on the complexity of the screen. Recognize the screen using text that is drawn last, so Host Publisher will wait until the screen is complete before continuing with the macro.
5. Define global screens and associated actions for these screens that might appear at any time, such as monthly reminders or messages from colleagues. Global screens allow you to define a generic action to take (such as a clear screen command) if such a screen is encountered, allowing the macro to return to the normal flow.

Problems publishing applications to a Solaris or AIX Host Publisher Server

Cause: WebSphere 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 staging and production 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.

Solution: When you specify a Sun Solaris or AIX FTP server (Preferences > Options) in Host Publisher Studio, or when configuring a server in the Host Publisher Studio's wizard, specify the nobody user ID to use when transferring application files to that server using FTP.

You will have to use this new user ID when transferring applications to this Host Publisher Server.

Connect timeout in a database Integration Object has no effect

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

Solution: 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.

An Integration Object input is also being shown as an output

Cause: When building Integration Objects, Host Publisher Studio creates an associated output method (*getter*) for any input 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 for 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.

Solution: None needed.

PluginTester servlet for debugging WebSphere problems

The servlet PluginTester (http://*yourhost*/servlet/PluginTester) provides useful information about which JAR files have been opened to load classes, as well as other information for debugging problems.

Latest version of page with Internet Explorer 5.0

Cause: If you are using Internet Explorer 5.0, it is possible you will not always receive the latest version of a page when you do a refresh or when you return to a previously-viewed page. Internet Explorer 5.0 has set **Check for newer versions of stored pages** to Automatically. This setting overrides instructions to force the browser to the new server every time.

Solution: Continue to refresh until the latest version of the page appears.

Solution: Change the setting to something other than Automatically.

1. From the Internet Explorer menu bar, select Tools.
2. Select Internet Explorer Options.
3. In the Temporary Internet Files box, click Settings.
4. Click either **Every time you start Internet Explorer** or **Every visit to the page**.

Pages are not returned

Cause: The error log indicates that Host Publisher ran out of time while trying to set up a connection.

Solution: Increase the Connecttimeout parameter in the .connspec file, and the timeout attribute of the <HAScript> tag in all the macros.

Shutting down Host Publisher Server

Cause: If you shut down Host Publisher Server by shutting down the Web server or WebSphere Application Server, host connections may not be properly cleaned up.

Solution: Stop Host Publisher Server before you stop WebSphere.

Host Publisher Server and other WebSphere applications shut down after an application generated in Host Publisher Studio is deployed

Cause: WebSphere automatically restarts all Java servlets when it detects that applications have been deployed that use Java resources already in use, such as JAR files. This happens to allow Java servlets that require these JAR files to pick up the changes. This causes Host Publisher Server and any other servlet in WebSphere to stop and restart, which disrupts use of the applications.

You may not notice an interruption with the servlets, since they are restarted as quickly as possible; however, applications in progress at the time of the shutdown might not be restored to the starting state, such as a terminal log in screen. The applications have to be manually returned to starting states.

Solution: Deploy new or existing applications only when you know that the system is not being utilized. This reduces the likelihood that the deployment will adversely affect other applications or users.

Failures creating Integration Objects

Cause: Integration Objects are Java programs that are generated from the information you provide in Host Access or Database Access. Host Publisher builds Java source code and compiles it into an executable file. When saving and generating an Integration Object, you may receive a message that the Integration Object could not be created. This message indicates that the compilation of the generated Java source contained errors.

Solution: If you are an advanced user, you may find useful information about the cause of the failure in the iofailed.txt file located in the *yourinstalldir*\Studio directory.

If you have manually modified the host macro files or the .hpi files, use the messages in iofailed.txt to determine if your changes caused the error. If you have not customized any of the Host Publisher files, contact IBM and report the problem.

Out of memory error starting 20 sessions

Cause: 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.

Solution: When you configure per process thread limits for a platform, remember that the process is typically used to handle threads of the Web server, as well as threads of WebSphere.

Problems associated with replicating Host Publisher Servers

You can replicate Host publisher Server Version 2.1 using one of two methods:

1. Install Host Publisher Server on multiple machines and transfer identical applications to each machine. Then place SecureWay Network Dispatcher in front of these clones, which are all running the same Host Publisher (Studio-generated) applications. See“Balancing the server’s load” on page 33 for more information on setting up SecureWay Network Dispatcher.
2. Install Host Publisher Server on the S/390. Then use the scalable Web server facility on the S/390 (works in conjunction with the S/390 Work Load Manager (WLM)) to create multiple copies of the Web server. Each copy potentially serves different URLs; for example, /HostPublisher/callup/index.html and /HostPublisher/puborder/index.html would go to different Web server instances on the S/390.

You can run multiple Web server clones on the S/390 using WLM; however, only one of them can have WebSphere V1.1 running. It is therefore not possible to create replicas and clones of Host Publisher server on the S/390 using WLM. WebSphere V1.2 solves this problem; however, at the time of this writing, WebSphere V1.2 is not supported by Host Publisher Server. Check SecureWay Host Publisher’s home page (http://www.ibm.com/software/network/hostpublisher) for up-to-date information on what levels of WebSphere are supported.

When you are running identical Host Publisher servers and deployment environments on multiple machines with Network Displatcher in front as a load-balancing mechanism, there is a potential Host Publisher Server replication issue. The main points are:

- Everything works fine if the Host Publisher applications deployed.

  - Do not use chained Integration Objects.
  - Do not use user lists for hosts supporting a single logon per user ID.
  - Do not use connection pools where it is important to manage connection pool limits across clones.
- For chained Integration Objects, set the Network Dispatcher sticky port timer to be greater than the HTTP session timer in each WebSphere configuration to provide HTTP session affinity.
- The administration of each Host Publisher Server must be done from behind Network Dispatcher, not by accessing a common URL through Network Dispatcher. Network Dispatcher cannot guarantee which server you end up managing.
- Global user lists and connection pools are not supported. Each clone creates a pool that can grow as large as the maximum connections configured in the poolspec file. For user lists for hosts supporting a single logon per user ID, publish a separate userpool file to each clone with different user IDs. If these files have different names, adjust the poolspec file on each clone to point to its respective userpool file.

  As an alternative, transfer the same userpool file to ech clone; however, make sure there are no duplicate user IDs. Currently, there is no Studio support for this method.

Gathering WebSphere and Web server problem data

WebSphere maintains the following types of error logs:

1. Standard output and standard error output logs from the Java virtual machine
2. Error logs redirected from Web servers
3. Error logs for problems and events discovered by WebSphere while trying to build or run a servlet

For information on how to obtain problem determination data from WebSphere, access the WebSphere Web page at

`http://www.ibm.com/software/webservers`

Standard output and standard error output logs

WebSphere uses a Java virtual machine to run Java servlets, Host Publisher Server, and Integration Objects. WebSphere creates two files that contain the standard output and standard error output from the Java virtual machine: jvm_stdout.txt and jvm_stderr.txt.

To access these error log files, navigate to the directory path containing the logs for your operating system:

MVS /usr/lpp/WebSphere/AppServer/logs

AIX /usr/lpp/IBMWebAS or /usr/WebSphere/AppServer/logs

Sun Solaris
/usr/WebSphere/AppServer/logs

Windows NT
(drive):\WebSphere\AppServer\logs

Error logs redirected from Web servers

WebSphere is configured to use Web servers. WebSphere redirects error logs from Web servers into the WebSphere directory structure. The redirected error logs are sequentially numbered. The most recent of these files contain information related to events and errors detected by the Web server. These errors usually indicate unsatisfied page requests or Web server-specific errors. If the WebSphere directory structure does not contain Web server logs, check your Web server documentation for information on gathering problem data. You might be able to access Web server documentation from the Web server main page by entering the TCP/IP hostname or address as the URL.

To access the redirected Web server error log files (ncf.log), navigate to the directory path containing the logs for your operating system:

MVS /usr/lpp/WebSphere/AppServer/logs

AIX /usr/lpp/IBMWebAS or /usr/WebSphere/AppServer/logs

Sun Solaris
/usr/WebSphere/AppServer/logs

Windows NT
(drive):\WebSphere\AppServer\logs

Error logs for problems and events discovered by WebSphere

WebSphere creates three log files to record problems and events discovered while attempting to build or run a servlet. The log files are located in the servlet/servletservice directory. The names of the files are:

- access_log
- error_log
- event_log

Contacting IBM for service

This section lists a number of ways you can reach IBM. Depending on the nature of your problem or concern, we will ask you to be prepared to provide us with information to allow us to serve you better.

If you have a technical problem, please 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 the 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 occurred in the Host Publisher Studio, a copy of the output in the console window of the application. If the error occurred in the Host Publisher Server, copies of the messages.txt, jvm_stderr.txt, jvm_stdout.txt, error_log, event_log, and access_log files from WebSphere.
3. If the error occurred in the Host Publisher Server while attempting to access an application, the files for the application might be needed. You can find these files under the Server\production directory of the Host Publisher installation directory.
4. The level of Host Publisher Studio or Host Publisher Server. To determine the version of the Host Publisher Studio, Select **About...** from the **Help** menu. For Host Publisher Server, the version of the Host Publisher Server is located on the welcome screen for Host Publisher administration.

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/network/hostpublisher/`
- Access the IBM Personal Software Services Web page, which links to the IBM Software Support Handbook, at:

  `http://ps.software.ibm.com/`

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. Editing files

Host Publisher produces applications using standard open formats—such as HTML pages, Java class files, and XML files. This makes it simple to make changes to a Host Publisher application after it has been published to a Host Publisher Server. You don't have to keep returning to the Host Publisher Studio to make small changes to your application.

Note: If you make changes to applications on the server without updating the files in the Host Publisher Studio, you could lose the changes in the server version when you next publish your application. Be sure to update the version you keep in the Host Publisher Studio before you publish those files. There is no automatic way to synchronize the two versions.

A Host Publisher application is made up of several types of files. Some of the files are only used in the Host Publisher Studio. The sections below describe each file, explain how it is used, and provide the file format.

Integration Object project file (.hpi)

Host Access and Database Access store project information into .hpi files. These files describe the details you defined while creating an Integration Object. The following is a sample .hpi project file generated by Host Access.

```
<?xml version="1.0" standalone="yes"?>
<!DOCTYPE com.ibm.HostPublisher.IntegrationObject SYSTEM "io.dtd" []>
<com.ibm.HostPublisher.IntegrationObject name = "callup"   type="hod">
     <Package name = "IntegrationObject"/>
      <Session>
   <PoolName>callup<PoolName>
     </Session>

     <OutputVariable name="callupResults" type ="simple">
           <ScreenCoordinates x="2" y="14" dx="71" dy="6"/>
                <SubVariable name="column1" type ="array">
                     <RelativeCoordinates x="0" y="0" dx="26" dy="5"/>
                </SubVariable>
                <SubVariable name="column2" type ="array">
                     <RelativeCoordinates x="26" y="0" dx="4" dy="5"/>
                </SubVariable>
                <SubVariable name="column3" type ="array">
                     <RelativeCoordinates x="30" y="0" dx="9" dy="5"/>
                </SubVariable>
                <SubVariable name="column4" type ="array">
                     <RelativeCoordinates x="39" y="0" dx="9" dy="5"/>
                </SubVariable>
```

```
                <SubVariable name="column5" type ="array">
                     <RelativeCoordinates x="48" y="0" dx="9" dy="5"/>
                <SubVariable name="column6" type ="array">
                     <RelativeCoordinates x="57" y="0" dx="15" dy="5"/>
                </SubVariable>
     </OutputVariable>

     <HODMacro filename = "callup.macro"/>
     <SessionChain>
          <StartState name = "Start Label"/>
          <EndState name = "The End Label"/>
          <Position>middle</Position>
      </SessionChain>
<com.ibm.HostPublisher.IntegrationObject>
```

Tag descriptions:

com.ibm.HostPublisher.IntegrationObject

name Specifies the name of this Integration Object. This name must match the name of the file.

type The type of Integration Object described in this file. Valid values are **hod** or **db**.

Package

name Specifies the Java package name used when generating the Java source code for this object.

Session

PoolName
Specifies the connection pool name used by this Integration Object.

OutputVariable

Output variables define data that will be extracted during the macro execution.

name Specifies the variable name provided by the user during macro recording.

type Specifies the type of variable described by the tag. Valid values are **simple** and **array**. Simple variables are stored and displayed as single blocks of text. Array variables are stored as individual lines that can be displayed individually using the REPEAT tag on a JSP page.

ScreenCoordinates
Identifies a rectangular region on the application screen that defines the data to extract.

x Identifies the starting column number. The first column begins at 1.

y Identifies the starting row number. The first row begins at 1.

dx Identifies the total number of columns to include in this variable.

dy Identifies the total number of rows to include in this variable.

SubVariable Subvariables define further detail of the format of data defined by an output variable. Subvariables are generated when the user specifies data to be extracted as a table in Host Access. Each column identified by the user is defined using the SubVariable tag.

name Specifies the variable name provided by the user during macro recording.

type Specifies the type of variable described by the tag. Valid values are **simple** and **array**. Simple variables are stored and displayed as single blocks of text. Array variables are stored as individual lines that can be displayed individually using the REPEAT tag on a JSP page.

RelativeCoordinates

Identifies a rectangular region on the application screen that defines the data to extract.

x Identifies the starting column number. The first column begins at 1.

y Identifies the starting row number. The first row begins at 1.

dx Identifies the total number of columns to include in this variable.

dy Identifies the total number of rows to include in this variable.

HODMacro

filename

Identifies the filename contain the Host On-Demand macro recorded by the user for this Integration Object.

SessionChain

StartState name
The connection start state label given by the user.

EndState name
The connection end state label given by the user.

Position
The position of this Integration Object in the object chain. Valid values are **first**, **middle**, or **last**.

SQL Identifies the SQL statement to execute for database Integration Objects.

JDBCUrl

name The JDBC URL to connect to when executing this Integration Object.

JDBCDriver

name The JDBC driver name to use for the specified URL.

Host Publisher application (.hpa)

This XML file organizes all of the parts that make up a Host Publisher application, including Integration Objects and the Web pages that refer to them. Host Publisher applications are published to Host Publisher Servers for access by your customers. When Host Publisher Studio is used to load an existing application, it is this file that you actually open.

Here is a sample of a typical application file:

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE application SYSTEM 'WebBridge.dtd'>
<application>
<appl_name>testdb</appl_name>
<integration_object>
     <obj_name>D:\HostPublisher\Studio\IntegrationObjects\EmployeeQuery.jar
     </obj_name>
     <input_properties>
          <input>setLastName</input>
     </input_properties>
     <output_properties>
          <output>getEmployeesEmployeeIDResult</output>
          <output>getEmployeesFirstNameResult</output>
          <output>getEmployeesExtensionResult</output>
          <output>getEmployeesLastNameResult</output>
          <output>getLastName</output>
     </output_properties>
     <execution_method>doHPTransaction</execution_method>
```

```
</integration_object>
<page>d:\hostpublisher2\Studio\testdb\output.jsp</page>
<page>d:\hostpublisher2\Studio\testdb\input.jsp</page>
</application>
```

Tag descriptions:

appl_name
Names the Host Publisher application. This name must match the name of the file. It is also the name Host Publisher Server uses to track this application.

integration_object
Specifies the beginning of an Integration Object definition.

obj_name
Specifies the full path to the Integration Object source within the file system. If the Integration Object was generated by Host Publisher Studio, then this file refers to a .jar file containing the Integration Object Java Bean and its related files. Integration Objects can also be normal Java class files imported from outside Host Publisher Studio.

input_properties
Specifies the beginning of the list of inputs for this Integration Object. Inputs generally must be satisfied with data before the Integration Object can be executed. Each input is specified by a separate input tag under this tag.

input Specifies the Java method used to set an input value. For a Java Bean, this is typically the writer method for a Java Bean property.

output_properties
Specifies the beginning of the list of outputs for this Integration Object. Outputs are used to render Integration Object data within a Web page. Each output is specified by a separate output tag under this tag.

output
Specifies the Java method used to get data values from an Integration Object. For a Java Bean, this is typically the reader method for a Java Bean property.

execution_method
Specifies the Java method for invoking the Integration Object once the inputs are satisfied with data. After the execution method completes, the Integration Object's resulting data can be accessed using its output methods, if there are any.

page Specifies a Web page that is used, either directly or indirectly, to access Integration Objects.

Integration Object source (.java)

Integration Objects created by Host Publisher Studio are JavaBeans. The JavaBeans files are contained within a jar file and are generally made up of two files, the JavaBeans class and the JavaBeans Info class. These class files are generated based on a template maintained by the Host Publisher Studio and information provided by you through one of the Access applications.

Since the template is not available to you for customization and since any time the Integration Object is modified through Host Publisher Studio it is regenerated and compiled, do not customize the source files for the Integration Objects in any way. Instead, if you require custom logic to make use of Integration Object data, use JSP tags and additional Java code to include the logic in the Web pages, or develop another Java class to customize Integration Object results.

JavaServer Pages™ (JSP) Web pages

Host Publisher Studio generates JSP pages to manipulate Integration Objects and their output. JSP tags are similar to HTML tags, but their purpose is to instantiate Java objects, execute methods, and access the object's properties (inputs and outputs). JSP tags enable you to interact with Java objects using standard Web pages.

There are many JSP tags available. The pages Host Publisher Studio generates only use a few of the tags. You can use any tag from the Version .91 JSP specification (available from http://java.sun.com/products/jsp/techinfo.html) within a page.

The following is a sample JSP page, followed by a description of how the tags are being used.

```
<HTML>
<BEAN NAME="EmployeeQuery" TYPE="IntegrationObject.EmployeeQuery"
     INTROSPECT="yes" CREATE="yes" SCOPE="request">
<BEAN>
<% EmployeeQuery.setHPubStartPoolName("NorthwindLocal"); %>
<% EmployeeQuery.doHPTransaction(request, response); %>
<BODY>
<P>Table:
<TABLE BORDER>
<th>EmployeesEmployeeID</th>
<th>EmployeesFirstName</th>
<th>EmployeesExtension</th>
<REPEAT INDEX=idx1>
<tr>
<td>
<INSERT BEAN =EmployeeQuery PROPERTY  =EmployeesEmployeeIDResult>
```

```
<INSERT>
</td>
<td>
<INSERT BEAN =EmployeeQuery PROPERTY  =EmployeesFirstNameResult>
</INSERT>
</td>
<td>
<INSERT BEAN =EmployeeQuery PROPERTY  =EmployeesExtensionResult>
</INSERT>
</td>
</REPEAT>
</TABLE>
</BODY>
</HTML>
```

This page references an Integration Object called EmployeeQuery. After invoking the object, it renders the object's output in an HTML table with three columns. The following JSP tags are used on this page:

BEAN The BEAN tag identifies a Java object to be instantiated on the page. It contains a number of parameters:

NAME
This is the identifier that is used to reference the instantiated EmployeeQuery object on the page. For convenience, it has the same value as the base name of the Integration Object's full class-name (IntegrationObject.EmployeeQuery), but it does not have to.

TYPE This refers to the Integration Object's full class-package name.

INTROSPECT
This turns on automatic introspection for this Integration Object. This allows a Web parameter, such as one passed into this page via another input form page, to automatically have its value set as an input to the Integration Object if the parameter has the same name as an input property. Valid values are yes and no. For example, if the Web parameter were **name=Pam** and the EmployeeQuery Integration Object had a property called name, then WebSphere would automatically set the name property of EmployeeQuery to **Pam**.

CREATE
Specifies whether a new instance of this class is to actually be created or not. There could exist a Java class that can be manipulated statically without the need for an instance of that class, or there could be another JSP page that created this Integration Object instance with a SCOPE value of **session** (see below). Valid values are **yes** and **no**.

SCOPE

This specifies how long the instance of the Integration Object is to last. A value of request means that the life of the Integration Object is as long as the request for this page is processed. It is discarded when a new page is requested. A value of session means that the instance of this class is to be maintained past the current request, allowing other JSP pages, for example, to access this object again. Valid values are **request** and **session**.

Inline Java tag (<% %>)

These tags specify the beginning and end of actual Java code segments that are to be invoked as they are written. These segments may reference variables specified within other inline Java tags before these on the same page. As shown in the example above, these tags can be used to access or execute Java objects explicitly.

REPEAT

The REPEAT and end REPEAT (</REPEAT>) tags denote a section of the page that is to be repeated during output until WebSphere receives an ArrayIndexOutOfBounds exception. The only way to get this exception is through the use of indexed properties using the INSERT JSP tag (see below) or by throwing the exception yourself using the inline Java code tags. Therefore, REPEAT tags should not be used without JSP tags between them.

INDEX

This parameter to the REPEAT tag specifies an optional index variable that can be used within inline Java code to access the current index value of the repeat loop.

INSERT

This tag inserts an output property value from an Integration Object at the specified location. For this table, the insert position is a cell within a row of the table. Notice how the REPEAT tags enclose the creation of a row of data for the table. The REPEAT will continue until there are no more rows of data to extract. The INSERT tag can only be used on JavaBeans.

Normal Java objects require inline Java code to access the output method using the index variable on the REPEAT tag. For example, if there was a Java object on the page called myObject with an indexed output method called getMyData(), then an inline Java statement might look like: `<= myObject.getMyData(idx1); %>` The equal sign after the inline Java tag (<%=) specifies that the return value from the function should be displayed. Remember, in order for the REPEAT to be terminated, getMyData(idx1) must throw an ArrayIndexOutOfBounds exception when there is no more data to extract, or the REPEAT will continue indefinitely.

Connection and application configuration file formats

This section describes the format of configuration files used by the administration servlet to configure Host Publisher. The files use XML tags to structure their content. Host Publisher generates these files along with Integration Objects and publishes them to the server as part of an application. The XML syntax is as follows:

Connection pool specification (.poolspec)

Defines how to create a pool of connections to a host or database. These files specify parameters for pools of connections to data sources, such as 3270 applications or databases. They also serve as the main coordinating file for a complete connection pool definition (including connection, users, and connect and disconnect macros, if appropriate).

User pool specification (.userspec)

These files list the users and any associated user-specific information necessary for accessing a data source. It is this list of users and the connection definition that define a pool of connections.

Connection specification (.connspec)

These files specify the parameters necessary for establishing a connection to a data source, such as a 3270 application or a database.

Logon specification files (.logonspec)

These files specify the connect and disconnect macros for Host Access Integration Objects. They also specify the screen description for the screen that Host Publisher Server expects on the host session before the Integration Object can check the connection back into the pool.

Checkin screen description (.screen)

This Host On-Demand screen description identifies the host screen that should be active for a connection to be considered ready to be returned to the connection pool. If a connection is not in that state, then it is discarded or recycled in attempt to return the connection to that state.

Application manifest (.application)

Describes all Web documents, JavaBeans (and classes), and configuration files that an application requires.

This file is generated by Host Publisher Studio when transferring an application to Host Publisher Server. When the application is deployed into production by the server, this file ensures all parts of an application are moved into the production area.

Macro files (.macro)

For Host Access Integration Objects only, these files specify IBM SecureWay Host On-Demand keyboard and screen recognition macros. They are used for replaying sequences of keystrokes for performing

certain tasks for the Integration Object, such as logging on to a system or accessing a data screen on an application. See “Appendix B. Macro script syntax” on page 79 for more information on the format of this XML file.

Note: Configuration descriptions might refer to other files that can be referenced using relative path names. The forward slash (/) is used as a file name separator, and it is replaced by the platform-specific filename separator character when the administration servlet processes file names.

Format of XML-based connection pool specifications

XML tag conventions

The following sections describe the XML syntax used to define Host Publisher Server connections and connection pools, using examples. The following conventions have been used:

- Each file contains a set of tags that correspond to a single instance of the connection or connection pool that the file describes.
- A single top level tag identifies the type of connection being described, such as <poolconfig> or <connconfig>.
- A tag that describes an instance always has a name attribute.
- Different object-specific tags are used to distinguish the connections that they are instances of. For example, a connection pool specification configuration file can have a <hodpoolspec> or a <dbpoolspec> tag.
- Within the tag describing the object of interest are nested tags defining that object’s properties. Each nested tag within a connection-specific tag is an empty XML tag, and the value of the property that it represents is specified by an attribute.
  - If the property is a “simple” type (integer, string, etc.), the value is specified using a value attribute.
  - If the property is a reference to one of the above four record types, a refname attribute is used to reference that connection’s definition. Another file with that name and a fixed extension (in the production/poolspecs directory) contains that connection definition.
  - If the property is a reference to another Java object such as java.util.Properties, whose string representation can be quite big, the value is represented using a nonempty nested tag. An example of this is the sessionprops attribute of the hodconnspec tag.
  - If the property is a reference to an object that also has an XML representation (such as an HOD macro), then the object is stored in a separate file, and an empty tag with a filename attribute is used to reference that file.

Note: The file can be in a subdirectory relative to the *c:/HostPublisher* / Server / production / poolspecs directory, where *c:/HostPublisher* is the directory where Host Publisher is installed. As in all other cases, relative pathnames are specified using the separator character '/'.

- If a property value is not specified in the XML tag, the default value is used during execution.

Notes:

1. All timeout values are integers (32 bit), and the unit of time is seconds.
2. A timeout value of 0 indicates no waiting. A timeout value of -1 indicates an infinite wait. For counters, the upper limit is always the maximum value of the primitive integer type in Java (2,147,483,647).
3. While all attribute values in XML are strings, type information is provided for each property that the attribute represents since that will limit the string values that can appear (for example, if boolean, valid values are true and false).

XML tags for pool specifications

Each file is used to define an instance of the PoolSpec record. A pool specification is nested within a single <poolconfig> tag. The <hodpoolspec> tag is used to describe an HodPoolSpec record, and the <dbpoolspec> tag is used to describe an DbPoolSpec record. Both objects have the same set of properties with values defined using the set of nested tags described below:

maxidletime

The time, in seconds, after which a connection that is idle is removed from the pool, if the number of connections in the pool exceeds minconnections.

The value is an integer, either -1 or 60 or greater. The default is –1.

If maxidletime is set to -1, an idle connection is never removed from the pool.

maxbusytime

The time, in seconds, after which a connection that has been checked out of the pool is reclaimed, given the assumption that the Integration Object that acquired the connection is never going to release it.

The value is an integer, either -1 or 60 or greater. The default is -1.

If maxbusytime is set to -1, a busy connection is never reclaimed.

connecttimeout

The time, in seconds, for which a requester of a pooled connection is made to wait to acquire a connection from the pool if no connections are available.

The value is an integer, either -1 or 0 or greater. The default is 120.

If connecttimeout is set to -1, the requester will wait forever.

minconnections

The minimum number of idle connections that a pool will have. This does not imply that the administration servlet will create that many connections during initialization. The pool is populated on demand only.

The value is an integer. The default is 0.

maxconnections

This is the maximum size of the pool. Once this many connections have been created and all connections have been checked out, the next requester will wait unless overflowallowed is set to true. In that case, a new non-pooled connection is created.

The value is an integer. The default is 1.

overflowallowed

If set to true, when a request is received for a pooled connection and none is available (because the maxconnections limit has been reached), a nonpooled connection is used. When this connection is released, it is ended.

The value is boolean. The default is false.

poolingenabled

If set to false, connection pooling is disabled and requests to acquire connections from this pool result in non-pooled connections being handed back to the requester.

The value is boolean. The default is true.

hodconnspec

A reference to a HodConnSpec specification in another file. This tag (with different attributes) is also used in .connspec files to define HodConnSpec records.

dbconnspec

A reference to a DbConnSpec specification in another file. This tag (with different attributes) is used in .connspec files to define DbConnSpec records.

Examples: callup.poolspec

```
<?xml version="1.0" encoding="UTF8" ?>
<!DOCTYPE poolconfig SYSTEM "poolconfig.dtd">
<poolconfig>
<hodpoolspec name="callup">
    <hodconnspec refname="vm6conn"/>
    <hodlogonspec refname="vm6"/>
```

```
    <localuserpool refname="vm6users"/>
    <maxidletime value="600"/>
    <minconnections value="10"/>
    <maxconnections value="20"/>
 <connecttimeout value="30"/>
 <overflowallowed value="true"/>
 </hodpoolspec>
</poolconfig>
```

puborder.poolspec

```
<?xml version="1.0" encoding="UTF8" ?>
<!DOCTYPE poolconfig SYSTEM "poolconfig.dtd">
<poolconfig>
<hodpoolspec name="puborder">
    <hodconnspec  refname="vm6conn"/>
    <hodlogonspec  refname="vm6"/>
    <localuserpool  refname="vm6users"/>
    <connecttimeout value="30">
    <minconnections value="30"/>
    <maxconnections value="40"/>
</hodpoolspec>
</poolconfig>
```

empdb.poolspec

```
<?xml version="1.0" encoding="UTF8" ?>
 <!DOCTYPE poolconfig SYSTEM "poolconfig.dtd">
 <dbpoolspec name="empdb">
     <dbconnspec refname="empdb"/>
     <localuserpool refname="empdbusers"/>
     <connecttimeout value="20">
     <minconnections value="5"/>
   <maxconnections value="10"/>
 </dbpoolspec>
 </poolconfig>
```

XML Tags for connection specifications

Each file is used to define an instance of the ConnSpec record. A connection specification is nested within a single <connconfig> tag. The <hodconnspec> tag is used to describe an HodConnSpec record, and the <dbconnspec> tag is used to describe a DbConnSpec record. These records have different sets of properties, and the nested tags used to set their values are described in separate sections.

Host On-Demand connections: The following XML tags correspond to properties of an HodConnSpec record.

singlelogon
: If true, a single user ID and password cannot be used for multiple connections, and, if a LocalUserPool object is referenced, the user

ID/password pairs in that user pool are locked to prevent use by simultaneous connections. If this value is true and a user ID is not available from the pool, then the caller waits the time specified by the connecttimeout property.

The value is boolean. The default is false.

sessionprops
Contains Host On-Demand connection properties.

connecttimeout
The time, in seconds, that Host Publisher Server will wait while creating a host connection using Host On-Demand APIs, and priming it by running a connect macro.

The value is an integer, either -1 or 1 or greater. The default is 120.

disconnecttimeout
The time, in seconds, that Host Publisher Server will wait while running a disconnect macro and disconnecting a host connection using Host On-Demand APIs.

The value is an integer, either -1 or 1 or greater. The default is 120.

JDBC connections: The following XML tags correspond to properties of a DbConnSpec record.

drivername
The name of a JDBC driver (class) that can be used by Host Publisher Server to load the driver.

This string value is mandatory.

urlname
This URL must identify the database to which a connection is created.

This string value is mandatory.

connecttimeout
The time, in seconds, that Host Publisher Server will wait to create a database connection using JDBC APIs.

The value is an integer, either -1 or 1 or greater. The default is 120.

Examples: vm3.connspec

```
<?xml version="1.0"?>
<!DOCTYPE connconfig SYSTEM "connconfig.dtd">
<connconfig>
<hodconnspec name="vm3">
<singlelogon value="false"/>
<sessionprops>
autoFontSize=false
screensize=2
```

```
sessionType=1
OIAVisible=true
LUName=
codePage=037
codePageKey=KEY_US
SSLServerAuthentication=false
fontSize=10
fontSizeBounded=true
host=ralvms
SSL=false
TNEnhanced=false
port=23
</sessionprops>
<connecttimeout value="120"/>
</hodconnspec>
</connconfig>
```

empdb.connspec

```
<?xml version="1.0"?>
<!DOCTYPE connconfig SYSTEM "connconfig.dtd">
<connconfig>
<dbconnspec name="empdb">
<drivername value="com.ibm.db2.jdbcdvr"/>
<urlname value="jdbc://myserver.ibm.com/employeedb"/>
<connecttimeout value="60"/>
</dbconnspec>
</connconfig>
```

XML Tags to specify information for running HOD connect and disconnect macros

Each file is used to define an HodLogonSpec record that contains Host On-Demand-specific information used to prime connections to a specific host. The <hodlogonspec> tag is used to describe an HodLogonSpec record, and is nested within a single <logonconfig> tag. The following XML tags correspond to properties of an HodLogonLogoff record.

logonmacro

Points to a file containing the Host On-Demand connect macro (in Host On-Demand 4.0-defined XML format). A connect macro may not be needed if the Host Access Integration Object's data macro includes connect actions or if certain public domain hosts do not need a connect step.

This file reference value is optional.

logoffmacro

Points to a file containing the Host On-Demand disconnect macro (in Host On-Demand 4.0-defined XML format). A disconnect macro may not be needed if the Integration Object's data macro includes disconnect actions or if certain public domain hosts do not need a disconnect step.

This file reference value is optional.

checkinscreendesc

This is a string representation of a com.ibm.eNetwork.ECL.ECLScreenDesc object that is constructed in the Host Publisher Studio. When a connection is returned to Host Publisher Server, it checks the current screen against this screen description, and, if it does not match, connection recovery is initiated if necessary.

This string value is mandatory.

Example: This is the file vm6.logon. In the example, the file names have been derived from the record name by adding a standard suffix.

```
<?xml version="1.0"?>
<!DOCTYPE logonconfig SYSTEM "logonconfig.dtd">
<logonconfig>
<hdlogonspec name="vm6">
<logonmacro filename="vm6_logon.macro"/>
<logoffmacro filename="vm6_logoff.macro"/>
  <checkinscreendesc value="vm6_checkin.screen"/>
</hdlogonspec>
</logonconfig>
```

XML Tags to Define User ID/Password Pools

A LocalUserPool record defines a database of user ID/password pairs that are used to connect to a pool of connections. For hosts that allow one user ID to be used simultaneously to connect multiple times (for example, AS/400s) as well as databases, this database will typically have one entry only (if more than one entry is specified, Host Publisher Server will ignore the other entries). However, for hosts that do not allow simultaneous connections using one user ID (for example, 3270 hosts running VM), the database manages these user ID/password pairs by locking user IDs that are in use.

The user pool can be used to store more than just sets of user IDs and passwords. If the user ID is treated as a key in the database, then the password (to log on to the host), as well as other properties (another password to log on to another application as part of the session priming process), can be associated with each user ID. User pool properties other than the user ID can be encrypted, and different properties in each configuration can be encrypted differently.

A <schema> tag is used to define the properties that should appear in each database entry, and what the encryption level of each property should be. The <localuserpool> tag is used to describe a LocalUserPool record, and this tag is

nested within a single <userconfig> tag. Multiple <entry> tags are used to define the database entries, one for each user ID, password, and any other properties, using <property> tags.

The tag has an optional session attribute. If this attribute is present, and has a nonnull value, then the passwords in this file are encrypted. The session attribute is set by the RTE code if the designer using the Host Publisher Studio requests password encryption, and depending on its value, a determination is made about whether weak or strong encryption schemes are to be used. If strong encryption is to be used, the value of this attribute is based on an RTE startup password supplied at design time. The same RTE startup password must be supplied by the administrator at runtime to start the Server or to modify the user pool from the Host Publisher Server administration user interface.

Examples of User Pool Definitions: vm6users.userpool

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE userconfig SYSTEM "userconfig.dtd">
<userconfig>
<schema>
    <defineproperty encrypt="0" name="_userid"/>
    <defineproperty encrypt="0" name="app_password"/>
    <defineproperty encrypt="0" name="_password"/>
</schema>
<localuserpool name="nm01users">
   <entry  key="vm6Userid01">
      <property name="userid" value="vm6Userid01"/>
      <property name="_password" value="vm6Password01"/>
      <property name="app_password" value="apppw1"/>
   </entry>
   <entry  key="vm6Userid02">
      <property name="userid" value="vm6Userid02"/>
      <property name="_password" value="vm6Password02"/>
      <property name="app_password" value="apppw2"/>
   </entry>
</localuserpool>
</userconfig>
```

empdbusers.userpool

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE userconfig SYSTEM "userconfig.dtd">
<userconfig>
<userconfig>
<schema>
   <defineproperty encrypt="0" name="_userid"/>
   <defineproperty encrypt="2" name="_password"/>
</schema>
<localuserpool name="empdbusers"
session="AAEU2JptbII7I3UDqtdMHyu590xGde+p+oU8nxtjdMEisfJp0CsSOMd4gUdpbU5y7kmRe">
   <entry key="UserName5">
      <property name="_userid" value="UserName5"/>
```

```
        <property name="_password" value="o0+n5w1W3mhej7KWpb6SEw==">
    </entry>
    <entry key="UserName4">
        <property name="_userid" value="UserName4"/>
        <property name="_password" value="h57hlX=hMrl13baAuFhk9Q==">
    </entry>
</localuserpool>
</userconfig>
```

The application manifest

This file describes, using XML, all the resources that a Host Publisher Studio-generated application depends on, which the deployment tool understands. The application manifest lists the pool configuration files, Integration Objects, and JSP / HTML pages used by the application. No specific deployment support is provided for other dependencies, such as native executables. The manifest is used by Host Publisher Server during deployment to move files from the staging directory to the production directory and is also used to remove applications and provide administrative information.

There is one manifest file per application. Its name is derived from the application name and should match the subdirectory name under the directory where application-specific Web document files are transferred; for example, the *c:/HostPublisher* / Server / staging / applications subdirectory, where *c:/HostPublisher* is the directory where Host Publisher is installed. The file must have the extension .application, and it must be located in the application-specific subdirectory. During deployment the manifest file is moved to the *c:/HostPublisher* / Server / production / appmanifests subdirectory.

The application manifest tags

There are three main categories of tags in this file, which are used to reference the parts of an application:

- The <hodpoolspec> and <dbpoolspec> tags are used to identify one or more connection pools used by the application. These tags identify a connection pool specification file that has been placed in the staging / shared directory during the transfer step. The connection pool specification file is used to identify all other files that together define the pool characteristics.
- The <bean> and <beandir> tags are used to reference all the Java classes (for example, Integration Objects) that are used by the application. Either a specific class or JAR file, or an entire directory of class files, can be specified. All directory names are relative to the staging / shared directory where Java files are transferred.
- The <document> and <documentdir> tags are used to reference all the Web document parts (HTML, JSP, GIF, JPEG, and other files) that are used by the application. Either a specific file, or an entire directory of files, can be

specified. All directory names are relative to the application-specific subdirectory of the staging / applications directory where the application's Web document files are placed during the transfer step.

Note: All Web document files in the application-specific subdirectory referenced in the manifest are moved to the *c:/HostPublisher* / Server / production / documents subdirectory during deployment, where *c:/HostPublisher* is the directory where Host Publisher is installed. During this file / directory copy, the application-specific subdirectory is also copied. The URLs used to access the Web pages must therefore include the application name.

A sample application manifest

The following example illustrates the contents of the application manifest for a (host-based) application called callup. This file is called callup.application, and will be in the *c:/HostPublisher* / Server / staging / application / callup directory, where *c:/HostPublisher* is the directory where Host Publisher is installed, after the publishing step has been completed.

```
<?xml version="1.0"?>
<!DOCTYPE applconfig SYSTEM "applconfig.dtd">
<applconfig>
<hodpoolspec refname="callup">
<bean filename="com/xyzcorp/callup/callup1.class"/>
<bean filename="com/xyzcorp/callup/callup2.class"/>
<bean filename="xyzcorpmisc.jar"/>
<beandir name="com/xyzcorp/utils"/>
<!-- .java files too, for debugging on the server ? -->
<document filename="callup1.jsp/>
<document filename="callup2.jsp"/>
<document filename="startpage.html"/>
<document filename="errors/errorpage.html"/>
<documentdir name="images"/>
<!-- Entire directory contents should be published/deployed -->
</applconfig>
```

File locations before deployment

The following are the contents of the **/var/HostPublisher/Server/staging** directory (for AIX) after the callup application is transferred to the server.

- The **shared** subdirectory contains the following directories / files:
  - com / xyzcorp / callup / callup1.class
  - com / xyzcorp / callup / callup2.class
  - xyzcorpmisc.jar
  - com / xyzcorp / utils / util1.class
  - com / xyzcorp / utils / util2.class

  - Configuration files that together define the connection poolspec for the callup application.
    - callup.poolspec
    - vm6conn.connspec
    - vm6.logonspec
    - vm6_logon.macro
    - vm6_logoff.macro
    - vm6users.userpool
    - vm6_checkin.screen
- The **applications/callup** directory contains the following directories/files:
  - callup1.jsp, callup2.jsp
  - startpage.html
  - errors/errorpage.html
  - images/logo.gif, images/pic1.jpg, images/pic2.jpg
  - callup.application

Directory contents after deployment

The following are the contents of the **/var/HostPublisher/Server/production** directory (for AIX) after the administration servlet has copied the files from the staging directory during the deployment step. The production directory structure is now primed for RTE initialization to begin.

- The **beans** subdirectory contains the following directories/files:
  - com/xyzcorp/callup/callup1.class
  - com/xyzcorp/callup/callup2.class
  - xyzcorpmisc.jar
  - com/xyzcorp/utils/util1.class
  - com/xyzcorp/utils/util2.class
- The **documents/callup** subdirectory contains the following directories/files:
  - callup1.jsp, callup2.jsp
  - startpage.html
  - errors/errorpage.html
  - images/logo.gif, images/pic1.jpg, images/pic2.jpg
- The **documents/poolspecs** directory contains the following files:
  - callup.poolspec
  - vm6conn.connspec
  - vm6.logonspec
  - vm6_logon.macro
  - vm6_logoff.macro

- vm6users.userpool
- vm6_checkin.screen

- The appmanifests directory contains the file callup.application.

Editing the server.properties file

The server.properties file is a Java properties file that contains Host Publisher server settings. The file is created when you start the Host Publisher server for the first time. The server.properties file is 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 file.

Host Publisher uses the default value for any property without a specified value.

You can also edit the server.properties file manually to make changes to the server settings. If you manually edit the file, you should restart the server to ensure that the changes you make take effect.

The server.properties file contains the following properties:

num_licenses
Specifies the number of licenses you purchased.

The value is an integer. The default is –1.

Specify num_licenses = –1 if you purchased an unlimited license.

licenseTracking
Specifies whether HostPublisher tracks license usage or not.

Note: You can only modify this property by editing the server.properties file.

0 HostPublisher does not track license usage.

1 HostPublisher tracks license usage. The Host Publisher Server tracks the number of requests per minute and logs a message when the value exceeds the number of licenses purchased. The license usage information is written to a file named licenseX.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 512KB 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 512KB and is renamed, the old license2.txt is deleted.

The license usage files contain the following information, arranged in rows, with one row per hour. The values are separated by a comma (,).

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

Specifies the path and file name of the file to which messages are written. A backslash in the path should be specified with a double slash (\ \).

maxLogFiles

Specifies the maximum number of files for messages.

Note: You can modify this property only by editing the server.properties file.

In the server.properties file:

1. Set maxLogFiles to the desired number.
2. Optionally, set maxLogFileSize in kilobytes. The default is 512 KB.

To increase the number of log files, Host Publisher Server begins by adding a 1 to the end of the file name; for example, messages.txt becomes messages1.txt. As the size of the file grows, the following takes place:

1. Host Publisher writes to messages1.txt until it reaches maxLogFileSize.
2. Host Publisher closes and renames the file to messages2.txt, and opens a new messages1.txt.
3. When messages1.txt is full, Host Publisher renames it to messages2.txt, renames messages2.txt to messages3.txt, and opens a new messages1.txt.
4. When the maxLogFile number is exceeded, Host Publisher deletes the oldest file, which is the file with the highest number.

There are never more files than the number set for maxLogFile. For example, if maxLogFiles is 3 and maxLogFileSize is 1000, Host Publisher Server eventually creates three log files named messages1.txt, messages2.txt, and messages3.txt. The most recent log entries appear in messages1.txt, and its size is less than 1000 kilobytes. The older log entries appear in messages2.txt and messages3.txt, and each of these files is approximately 1000 kilobytes in size. All log entries older than those contained in messages3.txt have been discarded.

The value is an integer.

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 server.properties file.

The value is an integer. The default is 512 KB.

%traceFile

Specifies the path and file name of the file to which traces are written. A backslash in the path should be specified with a double slash (\ \).

maxTraceFiles

Specifies the maximum number of files for trace information.

Note: You can only modify this property by editing the server.properties file.

In the server.properties file:

1. Set maxTraceFiles to the desired number.
2. Optionally, set maxTraceFileSize in kilobytes. The default is 512.

To increase the number of trace files, Host Publisher Server begins by adding a 1 to the end of the file name; for example, trace.txt becomes trace1.txt. As the size of the file grows, the following takes place:

1. Host Publisher writes to trace1.txt until it reaches maxTraceFileSize.
2. Host Publisher closes and renames the file to trace2.txt, and opens a new trace1.txt.
3. When trace1.txt is full, Host Publisher renames it to trace2.txt, renames trace2.txt to trace3.txt, and opens a new trace1.txt.

4. When the maxTraceFile number is exceeded, Host Publisher deletes the oldest file, which is the file with the highest number. There are never more files than the number set for maxTraceFile.

The value is an integer.

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 server.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 logs 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:

1 API traces

4 Entry and exit traces

16 Miscellaneous traces

The value is a decimal integer. The default is 0.

%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.

%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.

%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.

%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.

%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.

%HPAdminFile
Specifies the path and file name of the file that defines the Host Publisher Server Administration main page. You should never change this value.

autoDeploy
Specifies whether Host Publisher deploys applications that have been transferred to the server when the server is started or not.

0 You must deploy published applications manually using Host Publisher Server Administration.

1 Host Publisher deploys applications automatically when the server is started.

The value is binary. The default is 0.

Appendix B. Macro script syntax

This section is excerpted from the *Host On-Demand Bean Reference*. The rest of this book is available on the Web, at http://www.ibm.com/software/network/hostondemand/library under "Host Access Beans for Java".

After a macro script has been created using the Macro bean, you might want to manually edit it. Manually editing macro scripts should only be performed by advanced users.

Introduction

The IBM Host On-Demand V4 Macro and MacroManager beans now use XML because a macro is better suited to the state machine model (the main reason for the move: XML is tailor made for a state machine).

The idea of a state machine may be fairly new to you. The idea behind a state machine, especially in the IBM Host On-Demand macro context, is simple. Think of how you use a host system from a terminal or a terminal emulator (like IBM Host On-Demand). The process you follow when you interact with a host system is illustrated in these steps:

1. The host sends an expected screen down to you at your terminal.
2. You look at and understand which screen is presented to you.
3. You take the required actions based on your understanding (type keystrokes, and so forth).
4. Another screen is presented after these actions.
5. If you see the screen you expected, you do this all over again.
6. If you do not see the screen you expected, call the help desk or handle the error.

This is the idea behind a state machine in the Macro context (although the Macro can't call the help desk for you). The states are the screens you expect to see, and you take actions on those screens to change from one state, or screen, to another. That's it, see a screen, perform the action, see the next screen. It is easier to understand (and program) a macro with this approach than having several if-then-else and do-while programming statements. Remember, see a screen, perform the action, see the next screen.

Now that you understand that a macro is a series of screens with their actions associated with them, take a look at how well suited XML is to coding a macro. Here is an example of how to specify a connect macro:

```
<HAScript>
<screen name="Logon" startscreen="true">
     <description>
     <string value="Please Log on" casesense="true"/>
     <cursor row="12" col="10"/>
     <description>
     <actions>
         <prompt name="ID" row="12" col="10" len="8"/>
         <prompt name="Password" row="13" col="10" len="8"/>
         <input value="[enter]"/>
     </actions>
     <nextscreens>
          <nextscreen name="Logon.Complete"/>
     </nextscreens>
</screen>
<HAScript>
```

These few lines of code demonstrate the power of this new syntax. All the screens you expect to see for a task (like connecting) are coded within <screen> tags in XML. You describe the screen in a <description>tag, specify the actions for the screen in an <actions> tag, and specify the screen you want to see next in a <nextscreens> tag.

With the above example, keep in mind that the actions happen in sequence. The screen tag describes a logon screen with the text **Please Log on** on the screen and the screen's cursor position at row 12, column 10. If the macro logic sees a screen matching this description, it prompts the user for an ID and password, places the prompt results at the specified row and column positions, sends the ENTER key, effectively connecting the user to the host. The nextscreens tag specifies the name of another screen tag that appears later in the macro. If the next screen does not appear, the macro logic returns an error.

Although there are a large number of valid XML tags, XML is not complicated. A screen is specified with a description, actions, and the next screens. When a macro is played and a screen matching the description appears, the actions are executed for that screen and the macro logic monitors the host for any next screens specified.

Macro Syntax

The following details each valid macro element:

```
<HAScript>
<screen>
<comment>
<description>
<oia>
<cursor>
<numfields>
<numinputfields>
```

```
<string>
<attrib>
<customreco>
<actions>
<prompt>
<input>
<extract>
<message>
<trace>
<custom>
<nextscreens>
<nextscreen>
<recolimit>
```

The following XML tags and their attributes are valid in the IBM Host On-Demand Macro XML namespace. This description of the tags is structured like an actual macro file.

Note: The tag and attribute values are not case sensitive.

Attention: All characters in a macro must be Unicode characters. Most text editors support this by default, because they use the ASCII character set, which is at the lower end of the Unicode character set.

<HAScript> tag

The <HAScript> tag is the main enclosing tag for the macro. All other tags at this level that are not HAScript are ignored by the parser.

The attributes of the <HAScript> tag are:

name The name of the macro. This attribute is optional. The name can contain any valid Unicode character.

description
The description of the macro. This attribute is optional. The description can contain any valid Unicode character.

author The creator of the macro. This attribute is optional. The author can contain any valid Unicode character.

creationdate
The date the macro was created. This attribute is optional. The creationdate can contain any valid Unicode character. The date format is not checked.

promptall
This launches all prompts at the beginning of the macro. This attribute is optional. The default is true. The value must be true or false.

pausetime

The sleep time in milliseconds initiated after a screen is matched. This is used to let the host quiet down. This attribute is optional. The default is no pause. The value must be a number.

Note: The maximum pause time is limited to the platform on which the macro is running.

timeout

The allowable time in milliseconds between recognition events. If time expires, the macro goes into the error state. You can override this value in the <nextscreens> element. The default is no timeout. The value must be a number.

Note: The maximum pause time is limited to the platform on which the macro is running.

Example

```
<HAScript name="Logon Macro" description="Logs me on" author="btwebb"
creationdate="12/29/1998" promptall="true" pausetime="500" timeout="10000" >
 ...
</HAScript>
```

<screen> tag

The <screen> tag is the enclosing tag for the screen.

The attributes of the <screen> tag are:

name The unique identifier for the screen. This attribute is mandatory and **must be a unique string among the other screen IDs**. The name can contain any valid Unicode character.

startscreen

If true, the screen should be the first screen seen. Any other screen generates an error. This value must be true or false. This attribute is optional. The default is false.

Note: There can be only one screen with the startscreen attribute set to true.

stopscreen

If true, a match on the screen causes the macro to stop playing. You can have multiple screens with the stopscreen attribute set to true. This value must be true or false. This attribute is optional. The default is false.

transient

If true, the screen is handled as transient. Transient screens exist

outside the normal macro flow. **They are matched after nontransient screens. If you specify next screens in a transient screen, the next screens are ignored**. Use this attribute to specify a screen that can appear at any time in the screen flow. This value must be true or false. This attribute is optional. The default is false.

Example

```
<screen name="screen1" startscreen="true" stopscreen="false" transient="false"> ...
</screen>
```

<comment> tag

The comment tag for the screen. This can contain any valid Unicode character.

There are no attributes for the <comment> tag.

Example

```
<comment> ... </comment>
```

<description> tag

The <description> tag is the enclosing tag for the description associated with the screen.

There are no attributes for the <description> tag.

Example

```
<description> ... </description>
```

<numfields> tag

The <numfields> tag defines the total number of fields on the screen. This element is optional. The number of fields not used if not specified.

The attributes of the <numfields> tag are:

number
The field count. The value must be a number. This is a required tag.

optional
If true, recognition matching passes the element, if they and all other description elements that are not optional match; or, if there are no non-optional elements, and at least one optional element matches. The value must be true or false. This element is optional. The default is false.

invertmatch
If true, recognition matching passes only if the screen does not match

this description element (boolean not operation). The value must be true or false. These element is optional. The default is false.

Example

```
<numfields number="10" optional="false" invertmatch="false" />
```

<numinputfields> tag

The <numinputfields> tag defines the total number of input fields on the screen. This element is optional. The number of input fields is not used if not specified.

The attributes of the <numinputfields> tag are:

number
The field count. The value must be a number. This is a required tag.

optional
If true, recognition matching passes the element, if they and all other description elements that are not optional match; or, if there are no non-optional elements, and at least one optional element matches. The value must be true or false. This element is optional. The default is false.

invertmatch
If true, recognition matching passes only if the screen does not match this description element (boolean not operation). The value must be true or false. These element is optional. The default is false.

Example

```
<numinputfields number="10" optional="false" invertmatch="false" />
```

<oia> tag

The <oia> tag specifies an operator information area (OIA) condition to match. This element is optional. The default is to wait for inhibit status.

The attributes of the <oia> tag are:

status If NOTINHIBITED, the OIA must be uninhibited for a match to occur. If DONTCARE, the OIA inhibit status is ignored. This has the same effect as not specifying OIA at all. Valid values are NOTINHIBITED and DONTCARE. This is a required tag.

optional
If true, recognition matching passes the element, if they and all other description elements that are not optional match; or, if there are no

non-optional elements, and at least one optional element matches. The value must be true or false. This element is optional. The default is false.

invertmatch
If true, recognition matching passes only if the screen does not match this description element (boolean not operation). The value must be true or false. These element is optional. The default is false.

Example

```
<oia status="NOTINHIBITED" optional="false" invertmatch="false" />
```

<string> tag

The <string> tag describes the screen based on a string.

The attributes of the <string> tag are:

value The string value. This value can contain any valid Unicode character. This is a required tag.

row The starting row position for a string at an absolute position or in a rectangle. The value must be a number. This value is optional. If not specified, Macro logic searches the entire screen for the string. If specified, col position is required. <erow> and <ecol> attributes can also be specified to specify a string in a rectangular area.

Note: Negative values are valid and are used to indicate relative position for the bottom of the screen (for example, -1 is the last row).

col The starting column position for the string at an absolute position or in a rectangle. The value must be a number. This element is optional.

erow The ending row position for string in a rectangle. The value must be a number. This element is optional. If both erow and ecol are specified, string is in a rectangle.

ecol The ending column position for string in a rectangle. The value must be a number. This element is optional. If both erow and ecol are specified, string is in a rectangle.

casesense
If true, string comparison is case sensitive. The value must be true or false. This element is optional. The default is false.

optional
If true, recognition matching passes the element, if they and all other description elements that are not optional match; or, if there are no

non-optional elements, and at least one optional element matches. The value must be true or false. This element is optional. The default is false.

invertmatch
If true, recognition matching passes only if the screen does not match this description element (boolean not operation). The value must be true or false. These element is optional. The default is false.

Examples

```
<string value="hello" row="1" col="1" optional="false" invertmatch="false" />
<string value="hello" row="1" col="1" erow="11" ecol="11" casesense="false" optional="false" invertmat
<string value="hello" />
```

<cursor> tag

The <cursor> tag describes the screen based on the position of the cursor.

The attributes of the <cursor> tag are:

row The row position of the cursor. The value must be a number. This is a required tag.

col The column position of the cursor. The value must be a number. This is a required tag.

optional
If true, recognition matching passes the element, if they and all other description elements that are not optional match; or, if there are no non-optional elements, and at least one optional element matches. The value must be true or false. This element is optional. The default is false.

invertmatch
If true, recognition matching passes only if the screen does not match this description element (boolean not operation). The value must be true or false. These element is optional. The default is false.

Example

```
<cursor row="1" col="1" optional="false" invertmatch="false" />
```

<attrib> tag

The <attrib> tag describes the screen based on an attribute. This is an advanced feature and should only be used if needed. Usually all the other description elements are enough to describe a screen.

The attributes of the <attrib> tag are:

plane The plane value string that the attribute resides in. Valid values are COLOR_PLANE, FIELD_PLANE, and EXFIELD_PLANE. This is a required tag.

value The hex value string of the attribute. Example, value=″0xA0″. This is a required tag.

row The row position of the attribute. The value must be a number. This is a required tag.

col The column position of the attribute. The value must be a number. This is a required tag.

optional
If true, recognition matching passes the element, if they and all other description elements that are not optional match; or, if there are no non-optional elements, and at least one optional element matches. The value must be true or false. This element is optional. The default is false.

invertmatch
If true, recognition matching passes only if the screen does not match this description element (boolean not operation). The value must be true or false. These element is optional. The default is false.

Example

```
<attrib value="0x01" row="1" col="1" plane="COLOR_PLANE" optional="false" invertmatch="
```

<customreco> tag

The macro logic will call out to any custom recognition listeners for the custom tag to have the listener do its own custom screen recognition logic.

The attributes of the <customreco> tag are:

ID The unique identifier for the custom description element. Allows for multiple custom elements. This can be any valid Unicode character. This is a required tag.

Example

```
<customreco id="id1" />
```

<actions> tag

The <actions> tag is the enclosing tag for the actions associated with the screen.

The attributes of the <actions> tag are:

promptall

If this value is set to true, the macro bean will gather all prompts **within the current action tag** and launch them as one prompt event. The value must be true or false. This attribute is optional. The default is false.

Example

```
<actions promptall="true"> ... <actions>
```

<prompt> tag

The <prompt> tag specifies a prompt to be handled for the screen.

The attributes of the <prompt> tag are:

row The row to place the prompt. The value must be a number. This is a required tag.

col The column to place the prompt. The value must be a number. This is a required tag.

len The length of the prompt. The value must be a number. This is a required tag.

name The name of the prompt. This can be any valid Unicode character. This element is optional.

description

The description of the prompt. This can be any valid Unicode character. This element is optional.

default

The prompt's default value. This can be any valid Unicode character. This element is optional.

clearfield

This clears the host field on placement of prompt text. The value must be true or false. This element is optional. The default is false.

encrypted

Use a password echo character. The value must be true or false. This element is optional. The default is false.

xlatehostkeys

If true, host key mnemonics (example, [enter]) will be translated. For a list of key mnemonics, see the Host Access online help. The value must be true or false. This attribute is optional. The default is false. If you do not have this value set to true (which is normal because you wouldn't ask users to type key mnemonics), don't forget to code an input tag after the prompt(s) for the current actions to get the prompt data entered onto the host.

Example

```
<prompt name="ID" row="1" col="1" len="8" description="ID for Logon"
     default="btwebb" clearfield="true" encrypted="true"/>
```

<extract> tag

The <extract> tag specifies an extract to be handled for the screen.

The attributes of the <extract> tag are:

name The name of the extract. This can be any valid Unicode character. This element is optional.

srow Upper left row of the bounding extract rectangle. The value must be a number. This is a required tag.

scol The upper left column of the bounding extract rectangle. The value must be a number. This is a required tag.

erow The lower right row of the bounding extract rectangle. The value must be a number. This is a required tag.

ecol The lower right column of the bounding extract rectangle. The value must be a number. This is a required tag.

Example

```
<extract name="Get Data" srow="1" scol="1" erow="11" ecol="11" />
```

<input> tag

The <input> tag specifies keystrokes to be placed on the screen.

The attributes of the <input> tag are:

row The row position to send the keys. The value must be a number. This element is optional. This defaults to current cursor position.

col The column position to send the keys. The value must be a number. This element is optional. This defaults to current cursor position.

movecursor
The place the cursor at the end of the input string. The value must be true or false. This element is optional. This defaults to false.

value The text that is sent to the screen. This can be any valid Unicode character. This is a required tag.

xlatehostkeys
If true, host key mnemonics (example, [enter]) will be translated. The value must be true or false. This element is optional. The default is true.

Example

```
<input value="Your[tab]message[enter]" row="1" col="1" movecursor="true" xlatehostkeys="t
```

<message> tag

The <message> tag specifies a message to be sent to the user.

The attributes of the <message> tag are:

title The title to display in the message dialog. This can be any valid Unicode character. This element is optional. This defaults to macro name.

value The message to display in the dialog. This can be any valid Unicode character. This is a required tag.

Example

```
<message value="yourvalue" title="YourMessage" />
```

<trace> tag

The <trace> tag specifies a string to be sent to one of several trace facilities.

The attributes of the <trace> tag are:

type The type can either be sent to IBM Host On-Demand's trace facility, a user trace event, or to the command line. Respectively, the types are HODTRACE, USER, SYSOUT. This is a required tag.

value The text that is sent to trace. This can be any valid Unicode character. This is a required tag.

Example

```
<trace value="hello" type="HODTRACE" />
```

<custom> tag

The <custom> tag enables the user to have an exit to Java code, see Host On-Demand's Java documentation for the MacroActionCustom class.

The attributes of the <custom> tag are:

id The ID of the callout code that the Macro bean will use. This can be any valid Unicode character. This is a required tag.

args The argument string that can be passed to the callout. This can be any valid Unicode character. This element is optional.

Example

```
<custom id="custom1" args="YourArgument" />
```

<nextscreens> tag

The <nextscreens> tag contains all the valid next screens to be recognized after the current screen's actions have been executed. No text is allowed for the tag.

The attributes of the <nextscreens> tag are:

timeout
The allowable time in milliseconds that can elapse between current screen and any next screen before the macro bean will go into the error state. This overrides the timeout attribute for the entire macro. The value must be a number. This element is optional. The default is to use the overall macro timeout.

Example

```
<nextscreens> ... </nextscreens>
```

<nextscreen> tag

The <nextscreen> tag forces a next screen. Multiple nextscreen tags are allowed. If a screen appears that is in the macro but is not a next screen, the macro will go into an error state. If the next screen refers to a screen tag that doesn't exist, the macro will have a parse error.

The attributes of the <nextscreen> tag are:

name The name of the <screen> element that is the valid next screen. This can be any valid Unicode character. This is a required tag.

Example

```
<nextscreen name="screen1" />
```

<recolimit> tag

The <recolimit> tag is for advanced use only. It is used to enforce a limited amount of time a screen can be recognized **in a row** before it goes to the screen indicated in the goto attribute. This tag is useful for screen looping where you know exactly how many times you'll see a given screen in a row. It also is a safeguard against infinite screen recognition. No text is allowed for the tag.

The attributes of the <recolimit> tag are:

value The allowable number of times to recognize a screen. This value must be a number. This is a required tag.

Note: The actions will not be executed the last time the screen is recognized.

goto The name of the screen to go to when recognition limit has been reached. This can be any valid Unicode character but the screen must exist in the macro. This element is optional. If no goto screen is given, the macro terminates.

Example

```
<recolimit value="3" goto="endscreen"/>
```

A powerful macro

This section gets its name because the following example demonstrates how all of the tags and their attributes can be used in a macro.

```
<HAScript name="Logon Macro" description="Logs me on" author="btwebb"
     creationdate="12/29/1998" promptall="false" pausetime="500" timeout="10000">
     <screen name="Logon" startscreen="true">
     <comment>
     The screen description and actions for this screen demonstrate
     how to use everything. Normally, a screen tag would not be so full.
    </comment>
<description>
     <oia status="NOTINHIBITED" optional="false" invertmatch="false" />
     <string value="Please Log on" casesensitive="true"/>
     <cursor row="12" col="10"/>
     <numinputfields number="2" optional="false" invertmatch="false" />
     <numfields number="10" optional="false" invertmatch="false" />
     <string value="Welcome" row="1" col="1" optional="false" invertmatch="false"/>
     <string value="Enter ID" row="1" col="1" erow="11" ecol="11" casesense="false" optional="false" invertmatch=
     <string value="USERID" />
     <attrib value="0x01" row="1" col="1" plane="COLOR_PLANE" optional="false" invertmatch="false" />
     <customreco id="logon" />
</description>
<actions promptall="true">
     <prompt name="ID" row="11" col="10" len="8" description="ID for Logon"
          default="btwebb" clearfield="true" encrypted="true" />
     <prompt name="Password" row="13" col="10" len="8"/>
     <extract name="Get Data" srow="1" scol="1" erow="11" ecol="11" />
     <message value="YourMessage" title="Message" />
     <trace value="logging on" type="HODTRACE" />
     <custom id="logon" args="YourArgument" />
     <input value="[enter]" movecursor="true" xlatehostkeys="true"/>
</actions>
<nextscreens timeout="20000" />
     <nextscreen name="Logon.Complete"/>
</nextscreens>
</screen>
<screen name="Logon.Complete" stopscreen="true">
<comment>
     This screen just checks to see if we're logged on OK then hits the ENTER key.
     Because it is a stop screen we don't have to specify any nextscreens tag.
</comment>
<description>
    <oia status="NOTINHIBITED" optional="false" invertmatch="false" />
    <string value="Logon Successful, hit ENTER to continue" casesensitive="true"/>
</description>
<actions>
     <input value="[enter]"/>
</actions>
</screen>
<screen name="MessageReceived" transient="true">
<comment>
     This screen demonstrates the idea of a transient screen. Say our host system can send us
     asynchronous messages while we're logging on, we just want to clear them. This screen
     handles this and is valid anytime a message screen appears. No nextscreens needed here.
```

```
</comment>
<description>
     <oia status="NOTINHIBITED" optional="false" invertmatch="false" />
     <string value="Message received, hit CLEAR to continue"/>
</description>
<actions>
     <input value="[clear]"/>
</actions>
</screen>
</HAScript>
```

Recolimit Demonstration

This macro demonstrates how to extract a list that exists on multiple screens using the recolimit and invertmatch attributes. The behavior of this macro is as follows:

1. Assume we get to the ExtractScreen.Main screen (recolimit is 1)
2. Data is extracted and PF8 is sent to page down
3. The ExtractScreen.Main is matched again (recolimit is 2), and so on
4. If recolimit becomes 25, meaning that ExtractScreen.Main was recognized 25 times, the actions for ExtractScreen.Main will not be executed the 25th time. Instead, the actions for ExtractScreen.Complete will be executed. The actions are not executed for ExtractScreen.Main to keep from extracting twice if the ExtractScreen.Complete screen is reached before the recolimit is reached.
5. If ExtractScreen.Complete is matched before the recolimit reaches 25, then we'll just get all the data in the system.

```
<HAScript name="Extracter Macro" description="Gets me data from a list" author="btwebb"
     creationdate="12/29/1998" promptall="false" pausetime="500" timeout="10000">
<screen name="ExtractScreen.Main">
<comment>
     We'll assume there would be other screens that log us on and get us to this point.
     This screen is the main extraction screen, and extracts data only if there is no b
     line at the bottom of the screen indicating there isn't anymore data (invertmatch="
     To be safe we'll apply a recolimit of 25. This screen does an extract, then pages d
</comment>
<description>
<oia status="NOTINHIBITED" optional="false" invertmatch="false" />
<string value="Data Screen"/>
<string value="           " row="24" col="1" erow="24" ecol="11" invertmatch="true" />
</description>
<actions promptall="true">
     <extract name="Get Data" srow="2" scol="1" erow="24" ecol="80" />
     <input value="[pf8]"/>
</actions>
<nextscreens timeout="20000">
    <nextscreen name="ExtractScreen.Main"/>
    <nextscreen name="ExtractScreen.Complete"/>
</nextscreens>
<recolimit value="25" goto="ExtractScreen.Complete"/>
</screen>
<screen name="ExtractScreen.Complete">
```

```
<comment>
     This screen is the final extraction screen, and extracts data and sends an exit comm
     Note: It is only different from the main screen in that the blanks must be there.
     Assume there would be other screens to take care of logging off.
</comment>
<description>
     <oia status="NOTINHIBITED" optional="false" invertmatch="false" />
     <string value="Data Screen"/>
     <string value="           " row="24" col="1" erow="24" ecol="11"/>
</description>
<actions promptall="true">
     <extract name="Get Data" srow="2" scol="1" erow="24" ecol="80" />
     <input value="exit[enter]"/>
</actions>
<nextscreens timeout="20000">
     <nextscreen name="ExtractScreen.Complete"/>
</nextscreens>
</screen>
</HAScript
```

Appendix C. Error Messages and Recovery Actions

If you need to contact IBM, see the information in “Contacting IBM for service” on page 50.

(HPS5000) The Admin Servlet initializes the Host Publisher run-time environment.

Explanation
This is an informational message.

User Action
None.

(HPS5001) The value {0} of the attribute {1} in the XML element {2} is not a valid value.

Explanation
In a configuration file, the specified value is not acceptable.

User Action
Publish the application associated with this configuration file again. If the problem continues, review the settings in the Studio for this application.

(HPS5002) The bean filename has a file extension that is not .class, .jar, or .zip.

Explanation
Integration Objects (beans) are expected to be stored in jar or class files, but the specified bean filename does not appear to be a jar or class file.

User Action
If the specified file is a jar or class file, change its name so that it has a .jar or .class file extension, as appropriate. If the file is not a jar or class file, ignore this message.

(HPS5003) There was an unsuccessful attempt to deploy new applications while the server was running.

Explanation
An unexpected error occurred.

User Action
If the problem persists, turn on all Host Publisher Server traces and try to recreate the condition. Contact IBM service and provide the trace information.

(HPS5004) There was an error parsing the application manifest file {0}: {1}

Explanation

A problem occurred while reading the specified application manifest file.

User Action

Look at the error message given, and look up the detailed help for that error message.

(HPS5005) Failed to create a directory named {0}.

Explanation

An error occurred that prevented creating the named directory.

User Action

Ensure the directory path is valid and the user ID that is running the Host Publisher Server has the necessary privileges to create the directory.

(HPS5006) Failed to delete the directory named {0}.

Explanation

An error occurred that prevented deleting the named directory.

User Action

Ensure the directory path is valid and the user ID that is running the Host Publisher Server has the necessary privileges to delete the directory.

(HPS5007) Failed to delete the file named {0}.

Explanation

An error occurred that prevented deleting the named file.

User Action

Ensure the file path is valid and the user ID that is running the Host Publisher Server has the necessary privileges to delete the file.

(HPS5008) There was an error reading the file {0}.

Explanation

Some error occurred while reading the specified file.

User Action

Ensure the specified file exists and the user ID that is running the Host Publisher Server has the necessary privileges to read the file. If the file is part of a published application, publish and deploy it again.

(HPS5009) There was an error writing the file {0}.

Explanation

Some error occurred while writing the specified file.

User Action

Ensure the directory for the file exists, and the user ID that is running the Host Publisher Server has the necessary privileges to create and write files in that directory.

(HPS5010) The installation directory parameter {0} specifies a directory {1} that is not an absolute path.

Explanation

The Host Publisher install directory specified in the WebSphere servlets.properties file is not a complete unambiguous directory.

User Action

The simplest solution is to install Host Publisher again, taking care to provide a complete installation path. If you prefer, you can use the WebSphere Administration program to correct the install_dir property of the HPAdmin servlet.

(HPS5011) The installation directory parameter {0} specifies a path {1} that does not exist.

Explanation

The Host Publisher install directory specified in the WebSphere servlets.properties file does not exist.

User Action

The simplest solution is to install Host Publisher again, taking care to provide a complete installation path. If you prefer, you can use the WebSphere Administration program to correct the install_dir property of the HPAdmin servlet.

(HPS5012) {0} is not a directory.

Explanation

The named object was expected to be a directory but was not.

User Action

If the message was logged from init, the install_dir property is incorrect for the HPAdmin servlet in WebSphere. Install Host Publisher again, or correct the servlet property using the WebSphere Administration program. If the message was logged from deployApplication, look for other messages that identify the application being deployed, and publish the

application again. If the message was logged from start, there is a problem with the Host Publisher installation; install Host Publisher again.

(HPS5013) The XML element {0} is missing the required attribute {1}.

Explanation

Expected item was not found in a configuration file.

User Action

Look for other messages that identify the application. Publish and deploy the application again.

(HPS5014) The initialization parameter {0} is missing. It should specify the installation directory name.

Explanation

WebSphere is not correctly configured to provide Host Publisher with the name of its installation directory.

User Action

The simplest solution is to install Host Publisher again, taking care to provide a complete installation path. If you prefer, you can use the WebSphere Administration program to correct the install_dir property of the HPAdmin servlet.

(HPS5015) The XML element {0} is missing either the attribute {1} or the attribute {2}.

Explanation

One of two expected items is missing in a configuration file.

User Action

Look for other messages that identify the application. Publish and deploy the application again.

(HPS5016) The XML element {0} is not a valid element in the file {1}.

Explanation

An unexpected item was found in the specified configuration file.

User Action

Look for other messages that identify the application. Publish the application again.

(HPS5017) Unexpected error {0} occurred. {1}.

Explanation

An unexpected error occurred.

User Action

If the problem persists, turn on all Host Publisher Server

traces and try to recreate the condition. Contact IBM service and provide the trace information.

(HPS5018) An exception occurred. {0}.

Explanation

An unexpected error occurred.

User Action

If the problem persists, turn on all Host Publisher Server traces and try to recreate the condition. Contact IBM service and provide the trace information.

(HPS5019) There was an XML parse error while reading the file {0} or its DTD file at line {1} in column {2}. The error message is {3}.

Explanation

There was something wrong with a configuration file.

User Action

Look for other messages that identify the application. Publish the application again.

(HPS5020) An exception occurred while parsing XML file {0} : {1}.

Explanation

An unexpected error occurred.

User Action

If the problem persists, turn on all Host Publisher Server traces and try to recreate the condition. Contact IBM service and provide the trace information.

(HPS5021) Invalid value of attribute {0} in object {1}. {0} can only be greater than 0.

Explanation

An error was found creating the named object. An invalid value has been specified for the named attribute.

User Action

Examine the pool and connection XML files, and ensure you are providing a valid value for the named attribute.

(HPS5022) Invalid value of attribute {0} in object {1}. {0} cannot be equal to 0.

Explanation

An error was found creating the named object. An invalid value has been specified for the named attribute.

User Action

Examine the pool and connection XML files, and ensure you are providing a valid value for the named attribute.

(HPS5023) Invalid value of attribute {0} in object {1}. {0} can only be 0 or higher.

Explanation

An error was found creating the named object. An invalid value has been specified for the named attribute.

User Action

Examine the pool and connection XML files, and ensure you are providing a valid value for the named attribute.

(HPS5024) Invalid value of attribute {0} in object {1}. {0} can only be -1 or higher.

Explanation

An error was found creating the named object. An invalid value has been specified for the named attribute.

User Action

Examine the pool and connection XML files, and ensure you are providing a valid value for the named attribute.

(HPS5025) Missing mandatory attribute {0} in object {1}.

Explanation

An expected item was not found in the specified configuration file.

User Action

Look for other messages that identify the application. Publish the application again.

(HPS5026) Missing mandatory attribute {0} in object {1}.

Explanation

An error was found creating the named object. The mandatory named attribute is missing.

User Action

Examine the pool and connection XML files, and ensure you are providing the mandatory attribute for the named object.

(HPS5027) Error creating object {0}. Object {1} not defined.

Explanation

An error was found creating the named object. The second named object did not exist. An error must have occurred while creating the second named object.

User Action

Examine the log messages before this one and resolve the errors that occurred creating the second named object.

(HPS5028) Object {0} already defined.

Explanation

An attempt was made to create an object already defined.

User Action

Examine the pool and connection XML files, and remove the duplicate occurrence of the named object.

(HPS5029) The server is not running.

Explanation

The Host Publisher Server has not been started, but an operation was attempted that requires the Server to have been started.

User Action

Use the Host Publisher Administration program to start the Host Publisher Server.

(HPS5030) {0} expired. Cannot get a connection from {1}.

Explanation

The maximum waiting time defined for the named pool has expired. No connection is available to satisfy the request.

User Action

Based on how often this condition occurs you might want to increase the maximum number of connections defined for the named pool.

(HPS5032) Cannot load Pool Manager {0}.

Explanation

An unexpected error occurred.

User Action

If the problem persists, turn on all Host Publisher Server traces and try to recreate the condition. Contact IBM service and provide the trace information.

(HPS5033) Too little memory. {0} bytes available.

Explanation

There is too little memory left to proceed.

User Action

Free some memory by shutting down applications and freeing disk space. Retry the application request.

(HPS5034) No HttpSession available.

Explanation

An application bean called ConnMgr.acquireConnectionFromWebsession, but there is no web connection. Either the application bean that requested the connection or the bean that saved the object are miscoded, or the web connection has timed out, and the web connection (HttpConnection) has been deleted.

User Action

Examine the application beans for errors, and determine whether or not the timeout value for the web connection (HttpConnection) is too short.

(HPS5035) There is no data source object {0} in HttpSession.

Explanation

An application bean requested a nonexistent data source connection object. The web connection does not contain the requested object. Either the application bean that requested the object or the bean that saved the object are miscoded; or the web connection has timed out, and the object has been removed.

User Action

Examine the application beans for errors, and determine whether or not the timeout values for the connection object and for the web connection (HttpConnection) are too short.

(HPS5036) Attempt to create user pool with invalid type: {0}.

Explanation

An unexpected error occurred.

User Action

If the problem persists, turn on all Host Publisher Server traces and try to recreate the condition. Contact IBM service and provide the trace information.

(HPS5037) Attempt to create user pool with invalid name: {0}.

Explanation

An attempt was made to create a user pool with no name.

User Action

Give the user pool a name.

(HPS5038) Error creating object {0}. Object is empty.

Explanation

The named object has no contents. Perhaps the XML file that defines the object has been corrupted. For example, a user pool object has no users.

User Action

Examine the XML files that define the named object. You may need to republish or repair the XML files.

(HPS5039) Cannot get a User from user pool {0}.

Explanation

The named user pool has no free users available.

User Action

Retry the request later, or enlarge the size of the user pool.

(HPS5040) Invalid value of attribute {0} in object {1}. {0} can only be -1 or greater than 0.

Explanation

An error was found creating the named object. An invalid value has been specified for the named attribute.

User Action

Examine the pool and connection XML files, and ensure you are providing a valid value for the named attribute.

(HPS5043) Internal error, attribute {0} in Conn object is null.

Explanation

This is an internal error.

User Action

Turn on all Host Publisher Server traces and try to recreate the condition. Contact IBM service and provide the trace information.

(HPS5046) JDBC driver {0} not loadable, received exception: {1}

Explanation

The JDBC driver (Java class) could not be loaded.

User Action

The driver may be missing. The directory where the class resides may not be in WebSphere's application server or reloadable servlet classpath. Check both. Use the WebSphere PluginTester servlet, as described in "Contacting IBM for service" on page 50, to check accessibility of the driver.

(HPS5047) JDBC getConnection call failed for URL {0}, received exception: {1}

Explanation

Could not connect to database (URL) specified in the connection specification for this application.

User Action

Verify that the database URL is reachable using DB vendor-supplied tools.

(HPS5048) Call to method {0} not expected for JDBC connections.

Explanation

This is an internal error.

User Action

Turn on all Host Publisher Server traces and try to recreate the condition. Contact IBM service and provide the trace information.

(HPS5049) The following Session properties are not acceptable to HOD : {0}. Exception received: {1}.

Explanation

This is an unexpected error.

User Action

Turn on all Host Publisher Server traces and try to recreate the condition. Contact IBM service and provide the trace information.

(HPS5050) Cannot recover connection if HodLogonSpec is null.

Explanation

This connection requires recovery because the user ID and password are characterized as being single use only, and when the connection was returned to the connection pool, the screen was not what was expected. However, this application has no connect specification, and thus there is no connect or alternate connect macro that can be run to recover the user ID and password.

User Action

None, if the above scenario is true. If not, you can change the connection specification to define the user IDs to not be single use only, or define connect and alternate connect macros.

(HPS5051) Cannot recover connection if logon or alternate logon macros are null.

Explanation

This connection requires recovery because the user ID and password are characterized as being single use only, and when the connection was returned to the connection pool, the

screen was not what was expected. However, this application has no connect or alternate connect macro that can be run to recover the user ID and password.

User Action

None, if the above scenario is true. If not, you can change the connection specification to define the user IDs to not be single use only, or define connect and alternate connect macros.

(HPS5052) Cannot set up a connection to the host using the following session properties: {0}

Explanation

Cannot connect to the host (TN server) specified in the connection specification file for this application.

User Action

Examine the host name and port number for the TN server that is in the connection specification file for this application. Check that TCP/IP connectivity to the TN server is available using the ping program. If that succeeds, then use an emulator for the right terminal type and check if the TN server is operational.

(HPS5053) Received InterruptedException: {0}

Explanation

This is an unexpected error.

User Action

Turn on all Host Publisher Server traces and try to recreate the condition. Contact IBM service and provide the trace information.

(HPS5054) Macro state STATE_EMPTY not reached. Current macro state is {0}.

Explanation

This is an unexpected error.

User Action

Turn on all Host Publisher Server traces and try to recreate the condition. Contact IBM service and provide the trace information.

(HPS5055) Ran out of time while playing HOD macro: Macro file name = {0} Macro = {1}

Explanation

While playing a connect (or disconnect) macro, the connect (or disconnect) timeout specified in the connection specification for this application expired. The timeout value could be too

small in the context of how busy the TN server or network traffic is. It is also possible that a screen expected while running the macro did not arrive.

User Action

Increase the timeout value and look at the log to identify the screen.

(HPS5056) Problem encountered while playing macro in file {0} Current state is {0} Initial fragment of the macro: {1}

Explanation

This is an unexpected problem.

User Action

Turn on all Host Publisher Server traces and try to recreate the condition. Contact IBM service and provide the trace information.

(HPS5057) PropertyVetoException received from HOD: {0}

Explanation

This is an unexpected error.

User Action

Turn on all Host Publisher Server traces and try to recreate the condition. Contact IBM service and provide the trace information.

(HPS5058) MacroException received from HOD: {0}

Explanation

The Host On-Demand code threw an exception while executing a connect or disconnect macro. This is an unexpected error.

User Action

Turn on all Host Publisher Server traces and try to recreate the condition. Contact IBM service and provide the trace information.

(HPS5059) HOD extract event {0} not expected. Extracts are not permitted in logon or logoff macros.

Explanation

The connect or disconnect macro has extract commands, which are unexpected in such macros and are ignored.

User Action

Examine your macros. Extraction of host screens are not supported while running macros to connect to, or disconnect from, a host.

(HPS5060) Both UserPool and BeanRef are null - cannot provide value for prompt {0} to HOD macro.

Explanation

While trying to supply a parameter to a connect or disconnect macro, the connection manager could not get it from the user pool, nor was there an Integration Object bean that could be interrogated for the value.

User Action

This is an internal error and should be reported to IBM Service.

(HPS5061) Ran out of time while supplying prompt values to HOD macro in file {0}. Exception received: {1}

Explanation

While setting up a connect or disconnect macro for running, the connect (or disconnect) timeout expired. The timeout value could be too small in the context of how busy the TN server or the network traffic is. It is also possible that a screen expected while running the macro did not arrive. The amount of time elapsed in attempting to connect to the host did not allow enough time to run the macro.

User Action

Increase the timeout value and look at the log to identify the screen on which it failed.

(HPS5062) Macro prompt value could not be provided. Exception received: {0}

Explanation

An error was encountered while trying to provide one or more values corresponding to prompt statements in a connect or disconnect macro.

User Action

Examine the prompt statements in the macro (the filename is logged) and check that the value being accessed is in the user pool or is available using a getter method in the Integration Object bean.

(HPS5063) Received the following MacroErrorEvent while playing HOD macro: {0}

Explanation

The execution of a connect or disconnect macro resulted in an error.

User Action

Examine the additional information provided in the log for more information on the cause of the error.

(HPS5064) No get_userid method (with either 0 or 1 argument) in bean.

Explanation

Expected the Integration Object bean to supply the password using a get_userid method since there was no user pool. However, no such method was found.

User Action

Fix the application in the Studio.

(HPS5065) No get_password method in bean.

Explanation

Expected the Integration Object bean to supply the password using a get_password method since there was no user pool. However, no such method was found.

User Action

Fix the application in the Studio.

(HPS5066) No ″getter″ method {0} for parameter {1} in bean {2}.

Explanation

A parameter to be supplied to a prompt statement in a connect or disconnect macro was expected to be found using a getter method in the Integration Object bean, since it was not found in the user pool entry associated with this connection. However, no getter method was found.

User Action

Correct the application in the Host Publisher Studio.

(HPS5067) Exception received when calling getter method in bean: {0}

Explanation

When attempting to get values from the Integration Object bean to supply macro prompt parameters to a connect or disconnect macro, a bean getter method threw a Java exception. This indicates there was a problem with the Integration Object generated by the Studio.

User Action

Turn on all Host Publisher Server traces and try to recreate the condition. Contact IBM service and provide the trace information.

(HPS5068) Exception received: {0}

Explanation

Received an unexpected exception while attempting to supply macro prompt values to a connect or disconnect macro by invoking getter methods of the Integration Object bean.

User Action

Turn on all Host Publisher Server traces and try to recreate the condition. Contact IBM service and provide the trace information.

(HPS5069) Attempt to set connection state to {0} is not valid.

Explanation

This is an internal error.

User Action

Turn on all Host Publisher Server traces and try to recreate the condition. Contact IBM service and provide the trace information.

(HPS5070) Cannot change originating Pool. Old: {0}, New: {1}.

Explanation

This is an internal error.

User Action

Turn on all Host Publisher Server traces and try to recreate the condition. Contact IBM service and provide the trace information.

(HPS5071) Cannot change connection specification. Old: {0}, New: {1}.

Explanation

This is an internal error.

User Action

Turn on all Host Publisher Server traces and try to recreate the condition. Contact IBM service and provide the trace information.

(HPS5072) There is no Pool Manager.

Explanation

An unexpected error occurred.

User Action

If the problem persists, turn on all Host Publisher Server traces and try to recreate the condition. Contact IBM service and provide the trace information.

(HPS5073) The Host Publisher encryption key is needed to start the system.

Explanation

The Host Publisher Server was unable to start because a required encryption key was not provided. This case-sensitive encryption key is unique to Host Publisher. The user configures it in the Host Publisher Studio when publishing one or more applications, if at least one of the user ID pools is strongly encrypted.

User Action

Use the Host Publisher Administration program to start the Host Publisher Server and provide the Host Publisher encryption key.

(HPS5074) The Host Publisher encryption key is needed to start the system because user pool {0} is encrypted.

Explanation

The named user pool contains encrypted data that requires the Host Publisher encryption key.

User Action

Supply the required encryption key.

(HPS5075) Received STATE_PLAY_ERROR while playing macro in file {0}.

Explanation

An error was encountered by the Host On-Demand code while executing a connect or disconnect macro.

User Action

The log will contain additional information about the host screen that was seen by the macro engine when the macro failed, and the macro screen on which the failure occurred. Examine both to determine if this is the screen the macro expected.

(HPS5077) Session is in CONNECTION_ACTIVE state, but not CONNECTION_READY state. A possible reason could be that the TN server port specified does not support the data stream expected. Session properties being used: {0}

Explanation

This can occur if the port specified for the TN server, in the connection specification for this application, does not support the terminal type expected. There could be other reasons too.

User Action

Verify that the port specified does support the expected terminal type (3270, 5250, or VT). If it does, contact your TN server administrator.

(HPS5079) HOD macro {0} has one or more syntax errors.

Explanation

The connect or disconnect macro currently running (identified in the log file) has a syntax error.

User Action

This should not occur unless the macro was manually edited. If you have edited the macro, examine the additional information logged in server.properties to determine the line where an error is being detected, and correct it. If no error can be detected, rerecord the macro using the Host Publisher Studio.

(HPS5080) Invalid value of attribute {0} in object {1}. {0} can only be -1 or greater than {2}.

Explanation

An error was found creating the named object. An invalid value has been specified for the named attribute.

User Action

Examine the pool and connection XML files, and ensure you are providing a valid value for the named attribute.

(HPS5081) The value {0} assigned to the attribute {1} in the file {2} is not valid. It should be a number. A value of {3} will be used.

Explanation

A configuration file contains a value that was supposed to be a number but was not. A default value was used.

User Action

Modify the value in the specified configuration file to be a number.

(HPS5082) Missing property {0} in user pool {1}

Explanation

The password expected by a prompt statement in a connect or disconnect macro is missing. The password was expected to be in the user pool.

User Action

Examine the user pool being used by this application. If you do see entries that define values for the property _password, contact IBM service. Otherwise, the user pool needs to be rebuilt using the Studio.

(HPS5083) Failed to find the file named {0}.

Explanation

The specified file does not exist or could not be read.

User Action

Ensure the specified file exists and the user ID that is running the Host Publisher Server has the necessary privileges to read the file. If the file is part of a published application, publish it again, and deploy it.

(HPS5084) Could not set up a connection.

Explanation

The Host Publisher Server ran out of time while setting up a connection to a backend data source. The backend data source may not be currently available, or your connection timeout in the connection specification file may be too low.

User Action

Check the availability of the backend data source. If it is available, then it is possibly too busy, and you should consider increasing the timeout value in your connection specification and redeploying that application.

(HPS5085) Licenses used ({0}) exceeding licenses purchased ({1}).

Explanation

The number of licenses in use has exceeded the number of licenses purchased.

User Action

Contact your support organization to purchase more licenses.

(HPS5086) Invalid value of attribute {0} in object {2}. {0} cannot be greater than attribute {1}. The value of attribute {1} will be used.

Explanation

An inconsistency was found creating the named object. An invalid value has been specified for the first named attribute. The value of the first named attribute cannot be greater than the value of the second named attribute. The value of the second attribute will be used for the first attribute.

User Action

This message is informational only; no action is required.

(HPS5087) There was an error deploying the application {0}. See previous log messages for details.

Explanation

An error occurred that prevented successfully deploying the application.

User Action

Examine the log messages before this one, and resolve the errors that occurred.

(HPS5088) There was an error copying directory {0} to directory {1} while deploying application {2}.

Explanation

Due to an error, it was not possible to copy the files from one directory to another in order to deploy the application.

User Action

Ensure the source directory exists and is readable and the destination is writable by the user ID that is running the Host Publisher Server. Also ensure that the application has been successfully published.

(HPS5089) There was an error copying file {0} to file {1} while deploying application {2}.

Explanation

An error occurred that prevented copying a file in order to deploy an application.

User Action

Ensure the source file exists and is readable and the destination directory is writable by the user ID that is running the Host Publisher Server. Also ensure that the application has been successfully published.

(HPS5090) System start.

Explanation

This is an informational message.

User Action

The Host Publisher Server has been started. This message is informational only; no action is required.

(HPS5091) System shutdown.

Explanation

This is an informational message.

User Action

The Host Publisher Server has been stopped. This message is informational only; no action is required.

(HPS5092) {0} Security attribute {1} appears to be corrupted, or the Host Publisher encryption key is incorrect. {2}

Explanation

An attempt to decrypt user pool data failed. The supplied encryption key is incorrect or the user pool data has been corrupted.

User Action

Examine the user pool's XML file, and ensure you are supplying the correct encryption key. You may have to republish the user pool with the correct encryption key.

(HPS5093) {0} Invalid schema value for {1} property. {2}

Explanation

An invalid encryption level is specified for the named property.

User Action

Specify a valid encryption level in the schema for the user pool.

(HPS5094) {0} User property {1} conflicts with schema. {2}

Explanation

An unexpected error occurred.

User Action

If the problem persists, turn on all Host Publisher Server traces and try to recreate the condition. Contact IBM service and provide the trace information.

(HPS5095) User {0} not found in user pool {1}.

Explanation

The named user was not found in the named user pool.

User Action

Respecify the name of the user.

(HPS5096) Pool {0} exceeded its capacity. Overflow connection created.

Explanation

The maximum number of connections defined for the named pool has been reached. A non-pooled connection has been created to satisfy the request.

User Action

Based on how often this condition occurs you might want to increase the maximum number of connections defined for the named pool. This message is informational only.

(HPS5097) Pool {0} exceeded its capacity. Request failed.

Explanation

The maximum number of connections defined for the named pool has been reached. A new connection could not be created to satisfy the request.

User Action

Based on how often this condition occurs you might want to increase the maximum number of connections defined for the named pool.

(HPS5098) Ran out of time while playing HOD macro: Macro file name = {0} Macro = {1} Exception received: {2}

Explanation

While supplying parameters corresponding to prompt statements in a connect or disconnect macro, the connecttimeout or disconnecttimeout timer expired.

User Action

The timeout value could be too small in the context of how busy the TN server or the network traffic is. It is also possible that a screen expected while running the macro did not arrive, and the connect (or disconnect) timeout expired.

(HPS5099) Internal error, the following attribute or result is unexpected {0}

Explanation

This is an internal error.

User Action

Turn on all Host Publisher Server traces and try to recreate the condition. Contact IBM service and provide the trace information.

(HPS5100) User ID {0} from UserPool {1} was used to set up a session to host {2}, and must be recovered.

Explanation

During shutdown, a disconnect macro associated with a host connection could not be run because the connection was in use by the Integration Object bean. This message logs which user ID is potentially lost as a result.

User Action

You can manually log on to the host with the above user ID and perform the alternate connect steps to recover the user ID. Otherwise, Host Publisher will perform that recovery on a per connection basis, on restart. The manual recovery will result in more efficient operation after restart.

(HPS5101) The WebSphere servlet engine cannot create an HttpSession object to store the Host Publisher connection with the key {0}.

Explanation

This is a Servlet engine error.

User Action

Turn on all Host Publisher Server traces and try to recreate the condition. Contact IBM service and provide the trace information.

(HPS6001) Unable to acquire session from the Session Manager with Session Pool: {0}

Explanation

A connection to the backend resource, using the configured pool name, could not be established.

User Action

Examine the message log for other messages that should give more details about this problem.

(HPS6002) Conn exception while releasing session to the Session Manager

Explanation

A connection resource could not be appropriately freed. This may cause resources to accumulate and therefore degrade performance.

User Action

Use a resource-usage monitoring tool to check memory, threads, and handle usage. Increasing use of these resources indicates the server will soon lock up. Check preceding error messages for more information.

(HPS6003) Unable to acquire connection from the Session Manager

Explanation

A connection to the backend resource could not be acquired.

User Action

Examine the message log for other messages that should give more details about this problem.

(HPS6004) Exception while invoking a property's read method

Explanation

An exception occurred when trying to access a routine that returns the value of a program variable.

User Action

Contact application vendor if problem persists.

(HPS6005) Internal processing error

Explanation

An unexpected error occurred during program execution.

User Action

Contact application vendor if problem persists.

(HPS6006) Introspection exception

Explanation

An Exception occurred while the Java Bean was performing internal introspection.

User Action

Contact application vendor if problem persists.

(HPS6100) Unable to acquire session from the Session Manager

Explanation

A connection to the backend resource, using a configured pool or connection, could not be established.

User Action

Examine the message log for other messages that should give more details about this problem.

(HPS6101) Internal HOD macro processing error

Explanation

An error occurred during execution of the host macro. The host macro is a step-by-step, predefined interaction between the terminal connection and the host.

User Action

Examine the message log for other messages that should give more details about this problem. Contact application vendor if problem persists.

(HPS6102) Internal macro processing error, extraction coordinates out of bounds

Explanation

An error occurred during execution of the host macro. An attempt was made to extract data from the terminal connection, but the terminal connection coordinates of this data are not defined within this terminal connection.

User Action

Contact application vendor if problem persists.

(HPS6103) Internal macro processing error, macro empty

Explanation

An error occurred during execution of the host macro. The host macro that was to be executed was either not found, or was empty.

User Action

1. Check whether the macro file is empty. If it is, go back to Host Publisher Studio and attempt to recover or record the macro again.
2. If it is not empty, then there might have been a problem during transfer. Use Host Publisher Studio to transfer the application to the Host Publisher Server again. Then redeploy it.
3. If the problem persists, contact IBM service.

(HPS6105) Received unexpected interrupted exception while waiting

Explanation

The synchronization logic of the application received an unexpected error.

User Action

Contact application vendor if problem persists.

(HPS6106) Unable to communicate with the Session Manager

Explanation

The application was not able to establish communications with the program that is responsible for maintaining connections to the backend resources.

User Action

Contact application vendor if problem persists.

(HPS6107) Illegal access exception while invoking a property's read method

Explanation

An exception occurred when trying to access a routine that returns the value of a program variable.

User Action

Contact application vendor if problem persists.

(HPS6108) Invocation target exception while invoking a property's read method

Explanation

An exception occurred when trying to access a routine that returns the value of a program variable.

User Action

Contact application vendor if problem persists.

(HPS6109) Exception occurred, cannot retrieve current screen

Explanation

An exception occurred while attempting to gather information about a prior error condition.

User Action

This message can be ignored. Examine the message log for messages about the original error condition.

(HPS6200) Internal error: Unable to build SQL query statement

Explanation

Unable to build a valid SQL query statement from the program parameters.

User Action

Check for prior error messages. Contact application vendor if this error persists.

(HPS6201) Internal error: Unable to build SQL update/insert/delete statement

Explanation

Unable to build a valid SQL update/insert/delete statement from the program parameters.

User Action

Check for prior error messages. Contact application vendor if this error persists.

(HPS6202) Internal error: Unable to locate read method for property

Explanation

An attempt was made to access a method that returns a value for a user or application program defined variable. This method does not exist.

User Action

The application is in error or a problem is present in the Java Virtual Machine environment. If the problem persists, contact the application vendor.

(HPS6203) Internal error: Unable to locate marker for end-of-input-variable in SQL statement

Explanation

Unable to parse the internally generated SQL statement.

User Action

The application is in error or a problem is present in the Java Virtual Machine environment. If the problem persists, contact the application vendor.

(HPS6204) Internal error: Unable to locate ForSQL read method for property

Explanation

An attempt was made to access a method that returns a value for a user or application program defined variable. This method does not exist.

User Action

The application is in error or a problem is present in the Java Virtual Machine environment. If the problem persists, contact the application vendor.

(HPS6205) SQL error while opening statement

Explanation

An error was received for the SQL call: createStatement.

User Action

Investigate the cause of the SQL error. A more detailed description of the SQL exception should be in message log file.

(HPS6206) SQL error while executing query with SQL statement: {0}

Explanation

An error was received for the SQL call: executeQuery.

User Action

Investigate the cause of the SQL error. A more detailed description of the SQL exception should be in message log file.

(HPS6207) SQL error while getting result set

Explanation

An error was received for the SQL call: getString.

User Action

Investigate the cause of the SQL error. A more detailed description of the SQL exception should be in message log file.

(HPS6208) SQL error while closing statement

Explanation

An error was received for the SQL call: closeStatement.

User Action

Investigate the cause of the SQL error. A more detailed description of the SQL exception should be in message log file.

(HPS6209) SQL error while executing SQL statement: {0}

Explanation

An error was received for the SQL call: executeUpdate.

User Action

Investigate the cause of the SQL error. A more detailed description of the SQL exception should be in message log file.

Appendix D. 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 User's Guide contains information on intended programming interfaces that allow the customer to write programs to obtain the services of Host Publisher.

Appendix E. Trademarks

The following terms are trademarks of the IBM Corporation in the United States or other countries or both:

- AIX
- DB2 Universal Database
- IBM
- OS/390
- SecureWay
- WebSphere

Other company, product, and service names may be trademarks or service marks of others.

ActionMedia, LANDesk, MMX, Pentium and ProShare are trademarks of Intel Corporation in the United States, other countries, or both.

(For a complete list of Intel trademarks see
http://www.intel.com/tradmarx.htm)

Adobe is a trademark of Adobe Systems, Incorporated.

C-bus is a trademark of Corollary, Inc. in the United States, other countries, or both.

DIGITAL is a trademark of Digital Equipment Corporation.

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.

Microsoft, Windows, Windows NT, and FrontPage 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.

PC Direct is a trademark of Ziff Communications Company in the United States, other countries, or both and is used by IBM Corporation under license.

Sybase and its corresponding logo are property of Sybase, Inc.

UNIX is a registered trademark in the United States and other countries licensed exclusively through The Open Group.

SET and the SET Logo are trademarks owned by SET Secure Electronic Transaction LLC. For further information see http://www.setco.org/aboutmark.html.

Index

Special Characters

A

B

C

D

N

O

P

R

S

T

U

V

W

X

Readers' Comments — We'd Like to Hear from You

IBM® SecureWay® Host Publisher
User's Guide

Publication No. GC31-8728-00

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
GC31-8728-00

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
Information Development
Department CGMD / Bldg 500
P.O. Box 12195
Research Triangle Park, NC
27709-9990

Fold and Tape **Please do not staple** Fold and Tape

GC31-8728-00

Cut or Fold
Along Line

Part Number: CT7CUNA

Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.

CT7CUNA

GC31-8728-00