IBM

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

Version 3 Release 5

GC31-8728-02

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

Version 3 Release 5

GC31-8728-02

Note

Before using this information and the product it supports, be sure to read the general information under "Appendix E. Notices" on page 181.

3rd Edition (June, 2001)

Contents

Chapter 1. About Host Publisher

About this information

This information is designed to help you, the Web Application Developer, plan for, configure, and start using IBM WebSphere Host Publisher. This book includes information about how to record interactions with data sources to produce Integration Objects, 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 8).

Notes:

1. Consult the product README and the Host Publisher Web site, http://www.ibm.com/software/webservers/hostpublisher, for corrections and additions to this information.
2. Throughout this book, **iSeries** refers to both iSeries and AS/400.

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 zSeries, iSeries, and RS/6000. Making this information available to new users and using it in new ways across intranets, extranets and the Internet enables you to reduce costs, improve services, generate new sources of revenue, and establish a competitive advantage.

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

Host Publisher enables you to create Integration Objects that contain logic to perform host data access and retrieval tasks. You can use these Integration Objects in WebSphere applications; for example, JavaServer Pages (JSPs), servlets, or Enterprise JavaBean (EJB) based applications. When you publish your application to a Web application server, 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 uses IBM WebSphere Application Server to provide a consistent, reliable execution environment for WebSphere applications; for example, servlets or EJB-based applications, and HTML/JSPs across platforms. Applications created in Host Publisher Studio can be accessed with all standard Web browsers, with or without Java.

Host Publisher supports terminal-oriented applications that use 3270, 5250, and VT data streams, as well as relational 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 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

Host Publisher provides Integration Object chaining. Integration Object chaining involves constructing a set of Integration Objects that depend on each other to drive a connection through several states. Chaining can increase performance and reduce the administration of creating complex applications. For example, you might use chaining in a typical 3270 application that uses multi-level menus. A corporate phone directory might have several menus to 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 task into steps, each of which will be represented as a single Integration Object, so that the end user does not have to navigate back down through several menus to reach the department list again.

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

Host Publisher provides support for executing Integration Objects remotely. Host Publisher Studio provides an option to create a Remote Integration Object (RIO) and associated support files when you create an Integration Object. You can invoke your RIO from an applet that runs in a Web browser or a Java application, which will in turn remotely execute your Integration Object, which is running on WebSphere. This allows you to execute Integration Objects from a client that is not executing in a WebSphere environment. In addition, you can develop an XML application, written in Java, Perl, C++ and other languages, to access the RIO servlet to execute your Integration Objects.

Host Publisher provides support for executing Integration Objects in EJB containers to take advantage of the server-side characteristics provided by the EJB architecture. The Host Publisher Studio provides an option to create a Host Publisher EJB and its support files when you create an Integration Object. The Host Publisher EJB is a stateful session EJB capable of running Integration Objects in an EJB environment. You can use Host Publisher Studio to build EJB-based applications.

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

For a technical overview of Host Publisher Integration Object execution, see "Appendix A. Technical overview" on page 113.

Comparing Host Publisher and WebSphere Application Server

Although Host Publisher and WebSphere Application Server 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 business logic. In contrast, WebSphere Application Server provides a robust Java infrastructure for the development and execution of Web applications and EJBs. WebSphere Application Server focuses on adding new business 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 Web applications that include Integration Objects created by Host Publisher. You can reuse Integration Objects within new servlets or EJB-based applications using WebSphere Application Server and your favorite Java interactive development environment (IDE), such as Visual Age for Java. For information about using Integration Objects in Java servlets and EJB-based applications, refer to the *IBM WebSphere Host Publisher Programmer's Guide and Reference*.

Host Publisher requires WebSphere Application Server Advanced Edition. However, if you need or already use the advanced features of WebSphere Application Server Enterprise Edition, you can substitute that product to support the Host Publisher runtime environment.

Comparing Host Publisher and IBM 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 to build applications for end users who are not familiar with typical host screens or how to navigate through legacy applications, and for whom a new, easy-to-use graphical interface is critical. Host Publisher also addresses the needs of those who are familiar with host applications, but who do not have Java-enabled browsers and therefore require HTML, or who prefer not to use the green-screen interface. As with Web self-service applications, these users 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.

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 WebSphere applications created outside Host Publisher Studio. 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 Page tags. You can use any HTML editor to enhance and customize the HTML or JSP tags to meet your design guidelines and personal preferences.

Host Publisher Server provides enterprise-class performance, scalability and availability through several key features, such as chaining, connection pooling, load balancing, hot standby, and cross-platform portability. Since the Host Publisher Server runs on AIX™, Windows 2000, Windows NT®, Solaris, and OS/400®, applications created with the common Host Publisher Studio will run unchanged in all environments.

Host Publisher can be used with WebSphere Edge Server's load-balancing capabilities to balance workload across a group of Host Publisher Servers. This provides predictable performance, easy scalability, and hot backup. The ability to move from one operating system platform to another will allow you to move your workload to a higher capacity platform as demands increase.

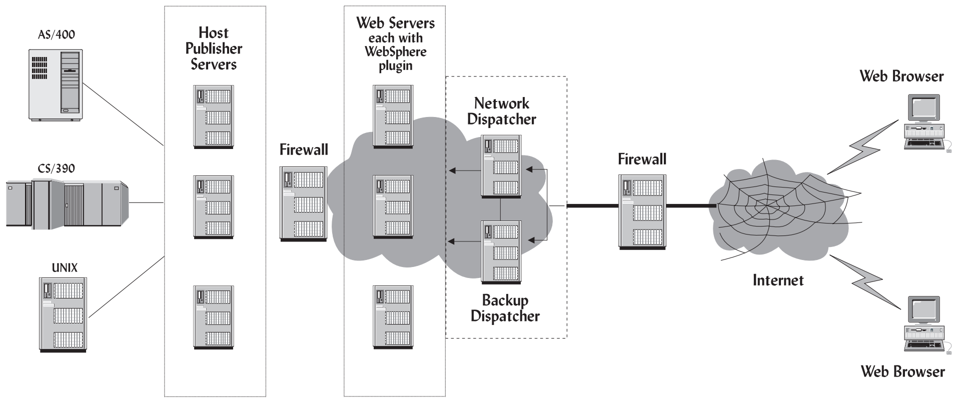


Figure 1. Host Publisher working in a network

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 applications. It uses task-oriented prompts to guide the user through the creation process—including recording host and database interactions, identifying desired data, and labeling that data for retrieval. Host Publisher Studio automatically generates Java beans called Integration Objects, which encapsulate the interactions and data retrieval logic. You can use Host Publisher Studio to generate fully-customizable Web pages for modeling interactions with the Integration Objects and rendering the resulting data. You can enhance the generated Web pages with your favorite Web authoring tool, such as WebSphere's page designer, to meet corporate guidelines on style and image. Once the Web pages are completed, you publish them to a Host Publisher Server for production access by end users.

Host Publisher Studio runs on the Windows 95®, Windows 98, Windows NT, and Windows 2000 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 Application Server and other run-time components such as connection management, license monitoring, run-time administration, express logon, Remote Integration Objects, EJBs, XML Gateway, and log and trace management.

Host Publisher Server is supported on OS/400, AIX, Windows 2000, Windows NT, and Sun Solaris operating environments. For information about supported Web servers, refer to the *Host Publisher Planning and Installation Guide* for your platform.

What's new in Host Publisher Version 3.5?

Host Publisher Version 3 Release 5 includes the following new function and features:

- **Distributed administration and support for WebSphere's load balancing**

  Host Publisher has been enhanced to support WebSphere Application Server's load balancing capabilities (cloning of JVMs on one machine and across multiple machines). You can now use one browser to administer multiple cloned Host Publisher instances.
- **Enterprise JavaBeans (EJB) support**

  Host Publisher delivery of EJB support provides those who are building EJB-based applications a locatable, scalable, distributed, and manageable server-side Java component with Host Publisher Integration Object function.
- **Express logon feature**

  Express logon allows you to log on to a 3270 application using a standard X509 V3 client certificate received through WebSphere Application Server from a browser. Use of the client certificate results in reduced password and user ID administration costs.
- **JavaServer Pages (JSP) 1.0 support**

  Host Publisher Studio now supports only JSP 1.0 pages. A migration tool is provided to help you convert .91 JSP tags (created prior to Host Publisher Version 3.5) to JSP 1.0 tags.
- **Host Access advanced screen recognition**

  In addition to the ability to recognize screens by text area, cursor position, and number of fields, the following recognition capabilities are now provided.
  - Comparison of text case (text area recognition)
  - Comparison of text color to a specified color
  - Comparison of a region to a value
  - Comparison of multiple region values
  - Multiple screen recognition criteria can be combined to form a logical expression for a single screen
  - Logical NOT
- **Keyboard settings**

Using Host Access, you can configure your host connection to use the keyboard settings you prefer. You can also add additional host functions to our predefined list. (Keyboard settings can be configured only in Host Access.)

- **Recording a macro for a chained Integration Object**

  Using Host Access to record a macro for an Integration Object that uses chaining is now easier. If you are recording an Integration Object that is middle or last in chain, you no longer have to manually navigate your terminal screen to your starting point before you begin recording your data macro. Instead, Host Access has a new function called Play Another Macro. Play Another Macro allows you to select another macro, for example, the data macro of the Integration Object that is first in the chain, and play that macro. Your terminal screen will be navigated through your selected macro.

Migrating from previous versions of Host Publisher to Version 3.5

When migrating from previous versions of Host Publisher to Version 3.5, you must install Version 3.5 in the same directory path to ensure access to your existing applications. Migration from previous versions of Host Publisher to Version 3.5 is not backwards compatible; that is, Integration Objects created with Host Publisher Studio Version 3.5 will not run on a previous version of the Host Publisher Server; however, Host Publisher Integration Objects created by a previous version of Host Publisher Studio will continue to run on any level of Server greater than or equal to the Studio level used to build the Integration Objects.

Note that applications containing chained Integrations Objects built by a previous version of Host Publisher will not run in WebSphere's load-balancing environment; however, you can modify your application to run in this environment. For more information, refer to the *IBM WebSphere Host Publisher Programmer's Guide and Reference*.

Host Publisher Studio Version 3.5 creates JSP 1.0 pages, and Host Publisher Server Version 3.5 supports only JSP 1.0 pages; therefore, Host Publisher provides a migration tool called JSPMigrator for migrating 0.91 JSPs to 1.0 JSPs. See "Migrating JavaServer Pages (JSPs)" on page 34 for detailed information on JSP migration.

For more information

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

For the *IBM WebSphere Host Publisher Administrator's and User's Guide*

AIX	/usr/lpp/HostPublisher/Common/doc/*lang*/guide.htm

OS/400	/QIBM/ProdData/Hostpublisher/doc/guide/guide.htm
Solaris	/opt/HostPublisher/Common/doc/*lang*/guide.htm
Windows NT and Windows 2000	*install_dir*\Common\doc\guide\guide.htm where *install_dir* is the directory in which Host Publisher is installed.

For the *IBM WebSphere Host Publisher Planning and Installation Guide*

AIX	/usr/lpp/HostPublisher/Common/doc/*lang*/instgd.htm
OS/400	/QIBM/ProdData/Hostpublisher/doc/install/as4inst.htm
Solaris	/opt/HostPublisher/Common/doc/*lang*/instgd.htm
Windows NT and Windows 2000	*install_dir*\Common\doc\install\instgd.htm where *install_dir* is the directory in which Host Publisher is installed.

For the *IBM WebSphere Host Publisher Programmer's Guide and Reference*

AIX	/usr/lpp/HostPublisher/Common/doc/*lang*/proggd.htm
OS/400	/QIBM/ProdData/Hostpublisher/doc/proggd/proggd.htm
Solaris	/opt/HostPublisher/Common/doc/*lang*/proggd.htm
Windows NT and Windows 2000	*install_dir*\Common\doc\proggd\proggd.htm where *install_dir* is the directory in which Host Publisher is installed.

For the IBM Host Publisher Readme

AIX	/usr/lpp/HostPublisher/Common/doc/*lang*/readme.htm
OS/400	/QIBM/ProdData/Hostpublisher/doc/readme.htm
Solaris	/opt/HostPublisher/Common/doc/*lang*/readme.htm
Windows NT and Windows 2000	*install_dir*\common\doc\readme.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
tr_TR	Turkish
zh_CN	Simplified Chinese
zh_TW	Traditional Chinese

To open the book (PDF) versions, substitute the following file names for the .htm file names:

Administrator's and User's Guide
guide.pdf

Planning and Installation Guide
instgd.pdf

Programmer's Guide and Reference
progguid.pdf

Online help, including the HTML version of this book, is available from the product's graphical 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/webservers/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 common tasks necessary to build Host Publisher applications.

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 14 and “Creating an Integration Object for database access” on page 28.
2. Create Web pages that use the Integration Objects to generate Web content. See “Creating Web pages for Integration Objects” on page 31.
3. Put the Web application onto a Host Publisher Server. See “Transferring applications to a Host Publisher Server” on page 43.
4. Make the pages available over the Web. See “Using the Server” on page 49.

Overview

Host Publisher enables you to create Java beans called *Integration Objects*. Integration Objects are Java beans that encapsulate interactions with a data source, such as a database or a 3270 application, and return specific data from that source for use as output to Web pages or in EJB applications. Integration Objects connect to the data source, navigate to the desired data, extract the data, and store the data in properties of the Integration Object, which can then be accessed.

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 JSPs that reference Integration Objects together form a Web application. After you transfer your application to the Server, use Host Publisher Server Administration to deploy your application to the Server to complete the publishing step. WebSphere Application Server parses each JSP that is accessed by a browser and converts the pages into a Java servlet, if this has not been done during a prior access. These servlets are Java programs that run under WebSphere Application Server. 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 Application Server 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 using 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.

There are two ways you can run Integration Objects remotely:

- Using the Studio, you can create EJB support for Integration Objects. The resulting EJB Access Beans can be copied to a remote system where they can be invoked by a servlet, JSP, or Java application to access the Host Publisher EJB and run Integration Objects located on the Host Publisher Server system.
- You can use Remote Integration Objects (RIOs) to access Integration Object data from a Java program (applet or application) running on a remote machine.

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. Host Access is used to access data from a 3270, 5250, or VT application, while Database Access is used to access any relational database for which a JDBC driver is available. 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 servlets and EJBs, or you can use Host Publisher Studio to transfer other Java objects and beans.

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

About macros

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

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

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

Connect
Includes the information Host Publisher needs to connect to the host. The connect macro should contain the steps necessary for logging on to a system from as many initial states as possible. It should take into account different paths that might occur because the previous connection failed or was left in an unknown state. Host Publisher Server uses the connect macro to attempt to connect to a system and to recover from a previously-failed connection. Refer to “Recording conditionals” on page 21 for more information on recording alternate paths.

Data Includes information about how to navigate, extract, and organize 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. As you navigate through your application using a terminal screen, you define:

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

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

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

Host Publisher uses IBM Host On-Demand's macro facility to provide an interface where you can interact with a host and record the actions you take. You might want to edit a macro script after it has been created using Host Access. Manually editing macro scripts should only be performed by advanced users. For detailed macro script syntax information, see the *IBM WebSphere Host Publisher Programmer's Guide and Reference*.

Creating an Integration Object for host access

To create Integration Objects that collect data from applications on the terminal-oriented 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 by using macros. (See "About macros" on page 13.)

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

- Using the wizard, page 14
- Defining host connections, page 15
- Creating connection pools for host access, page 15
- Recording interactions with a host, page 17
- Defining a screen, page 18
- Using input variables, conditionals, and looping, page 20
- Identifying data to extract, page 24
- Verifying a macro, page 25
- Editing a macro, page 25
- Defining global screens, page 25
- Generating an Integration Object, page 26
- Securing passwords and other sensitive data in macros, page 27
- About user lists, page 27

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 Integration 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 LU pool name, if any, to use.

Creating connection pools for host access

For each Integration Object, Host Access allows you to create a new connection pool, select to use a default connection pool, or select to use a connection pool that you have already defined.

If you create a new connection pool, by default your connection pool will be set to use connection pooling. You can select to disable connection pooling for each connection pool. If you do not use connection pooling, your Integration Object connects to the host each time you request a connection. Without connection pooling:

1. The Integration Object requests a connection from the pool
2. Host Publisher Server creates a new connection to the TN server.
3. The connect macro runs to log on to the host terminal
4. The Integration Object extracts the information
5. The Integration Object returns the connection to the pool
6. The disconnect macro runs to log off the host terminal
7. The connection ends.

Connection pooling keeps one or more connections to the host initialized, thus reducing the response time between when a client with a browser requests information and when the information displays on the page. You specify how many connections you want to remain active in the pool and ready for use, and when to remove connections from the pool. With connection pooling:

1. The Integration Object requests a connection from the pool
2. If an initialized connection is available, a new connection is not created and the connect macro does not run
3. The Integration Object extracts the information
4. The Integration Object returns the connection to the pool

5. The disconnect macro is not run
6. The connection is returned to the pool in its initialized state

You may specify for each connection pool to use one of several user accounts defined in the associated user list. See "About user lists" on page 27.

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

To create a new pool:
1. Select Create a new Integration Object, then click Next.
2. Select Create a new pool, then specify a pool name.
3. Click Advanced to bring up the Connection Pool Configuration window where you can enable connection pooling and specify connection timeouts and limits.
4. 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, select a name from the list.
   - If you want to create a new connection configuration, select New and provide the information to establish a connection to the specified host.
5. Click Advanced to bring up the New Connection Information window where you can define connection and security information.
6. Specify the host server information.
7. If you enable express logon, when you choose to insert a user ID or password, you must define an application ID and a user ID and password. (See "Express logon" on page 80 for a detailed description of express logon.) The application ID is used when you run your Integration Object on Host Publisher Server. It is not used when recording or playing your macro in Host Publisher Studio. The user ID and password are only for recording the macro in Host Publisher Studio. When you play the macro in Host Access, you must supply the user ID and password. When you run the macro on Host Publisher Server, the user ID and password are retrieved from the Digital Certificate Access Server (DCAS).

If you create a new pool and you do not enable express logon, you continue to the User List Configuration window, where you can define a list of users.

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

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

- Configure time-outs and connection limits
- Change connection configuration information
- Add users

Note: When WebSphere's cloning capabilities are used to create multiple instances of Host Publisher Server that share the same application files, the following restriction applies:

- Each JVM running Host Publisher Server enforces connection pooling restrictions only within the scope of that JVM. For example, if an Integration Object uses a host connection pool with a maximum of 100 connections, but there are three JVM clones running the same application, then there could be a maximum of 300 connections. You cannot use connection pooling to restrict the total number of connections Host Publisher Server creates when the Server is running cloned configurations.

For more information about cloning, refer to "JVM cloning and load balancing in WebSphere" on page 75.

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

At this point you can edit keyboard settings to assign keys or key combinations as shortcuts to functions. If you want to edit keyboard settings, from the menu bar, select Terminal > Keyboard > Edit Keyboard Settings. Each keyboard configuration is saved as *filename*.hpk in *Install_dir*\Studio\Preferences. If you want to delete a keyboard configuration, delete *filename*.hpk for that configuration.

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 screen just as you normally would, and Host Publisher records your interactions as a macro. The macro steps are displayed in the left pane in the window in a tree view.

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

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

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

If you select a screen node with a definition that does not match the currently-displayed screen and then select Record, a jump screen is inserted after the currently-selected screen node and you begin recording after the jump screen.

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
4. Define a screen

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

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

Defining a screen

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

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

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

- Comparison of text case

  Case sensitive is selected by default.
- Comparison of text color to a specified color

  You can change the coordinates of the color region and select different background and foreground colors.
- Comparison of a region to a value

  You can change the start and end position coordinates of the defined region on the terminal screen.
- Comparison of two region values

  You can change the start and end position coordinates of the first and second regions you defined on the terminal screen.
- Multiple screen recognition criteria can be combined to form a logical expression for a single screen

  You can choose whether each criterion is optional or non-optional.
- Logical NOT

  You can match a specified criterion or not match a specified criterion.

Some tips for defining a screen in a macro are:

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

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

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

Note: Screen segmentation is controlled by the host application and not by SNA parameters such as RU sizes or DLC MTU sizes. In SNA, a screen segment corresponds to a chain.

5. If you cannot determine if a screen is segmented, and you want to reliably recognize the screen in a macro, you can adjust the macro's *pause* parameter. Pause is the time delay, in milliseconds, between when actions of a screen description are performed and when valid next screens of the screen description are registered to the Host On-Demand screen recognition logic. The pause default is 200 milliseconds. This attribute must be modified using a text editor. For example:

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

   Notes:

   a. If the host application sends various screen segments in response to inbound keystrokes sent during actions, this delay can be used to ensure that all segments of the screen update arrive before the next screen match is attempted. Pause introduces a delay in the macro execution which affects performance and response time. If no application screens seen by a macro are segmented, setting pause to zero might result in improved Host Publisher runtime performance and scalability.
   b. There is a global pausetime attribute in the first tag in the macro that sets the default pause time for all screens in that macro. You can override this attribute by modifying the pause parameter.
6. Define global screens and associated actions for those screens that might appear at any time, such as monthly reminders or messages from colleagues. Global screens 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. See "Defining global screens" on page 25.

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

Using input variables, conditionals, and looping

The macros you record using Host Access can be simple or complex. Simple macros follow a straightforward path; each step leads to only one next step. Most often, the macros you record will be complex; they will include steps that lead to a choice of several steps, or they will include sequences of steps that repeat. When you have an input variable, for example, the user could enter information that leads to another prompt or to a loop of repeated actions. "Using input variables" on page 21, "Recording conditionals" on page 21, and "Recording loops" on page 23, describe how you can use commands on the toolbar to add complexity to a simple macro.

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

To insert an input variable:

1. Start recording a macro.
2. When you reach the point where the user will enter information, click the **Insert input variable** button on the toolbar.
3. On the window that appears, type a name for the variable and the 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.

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

For example, Figure 2 on page 22 shows an example of a conditional in a connect macro. When the connection is established, the terminal screen can display either a logged-in screen or a welcome screen that must be cleared before the logged-in screen will display. You can use **Insert Conditional** to define how Host Publisher handles either screen.

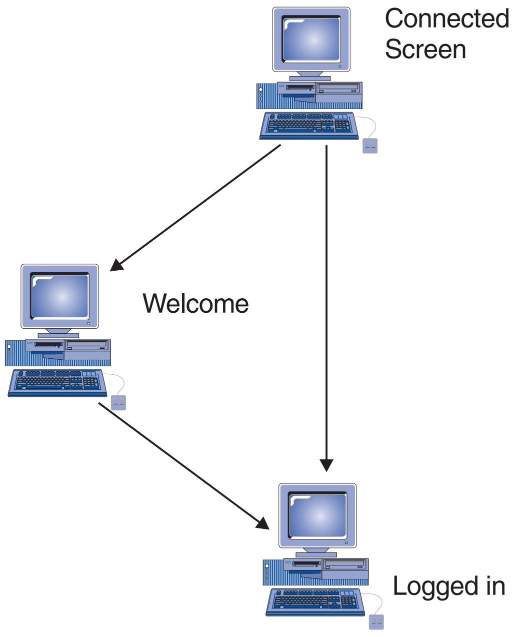


Figure 2. Host Publisher conditionals

To insert a conditional in a macro:

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

   Note that you can have several conditions for one step of a macro; for example, one screen might have several next screens associated with it in the macro tree. Each screen that you define has actions associated with it.

When you play your macro and Host Access encounters one of the possible screens you specify, it performs the actions associated with that screen.

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

Recording loops: Looping enables you to define an action that should be repeated until a condition is met.

For example, Figure 3 shows an example of a loop in a data macro. When the user requests data, the connection returns 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.

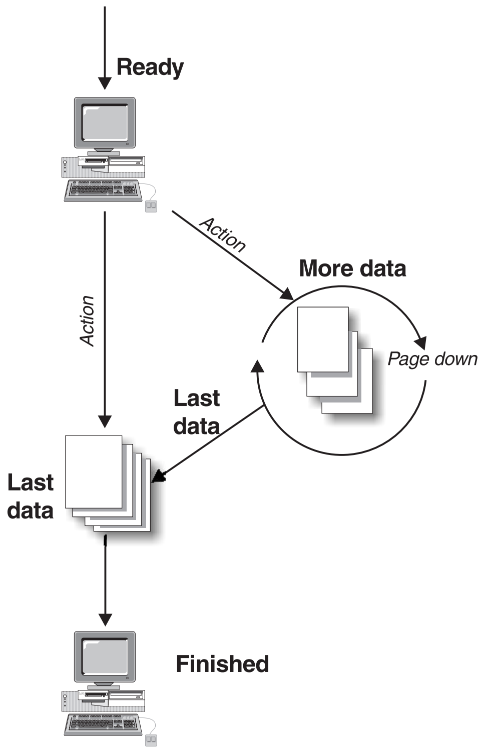


Figure 3. Host Publisher looping

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 about gathering data.
5. Click Next to continue.
6. Type the keystrokes that will cause the terminal to advance to the second screen in the loop. When the macro is executed, the keystrokes 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. You can now choose to define a unique screen where the loop will stop, or to stop the loop after a fixed number of iterations.
8. If you are defining a unique screen, Host Publisher stops recording while you navigate to the final screen of the loop. Host Access does not record the steps you take to navigate to the last screen because when the macro is running, this screen appears after the loop repeats. After you reach the screen that indicates the end of the loop, click Next to define the screen. In your definition, identify something on the screen that is displayed only when the loop is complete; for example, use words like *Bottom* or *Finished*, or use empty screen lines that do not appear until the last screen of the loop.

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

Identifying data to extract

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 screen with the host you defined. The wizard will ask you if you want to extract data in the Integration Object. If you do not, continue to use the terminal to record your navigation as a navigational macro. If you do want to extract data, use the terminal to navigate to the desired data, and select the data for extraction. You can:

- Select a table of data
- Select a region of text data
- Select multiple regions on a screen by choosing **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. Using Host Publisher Studio, 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 and inspect the captured data.
4. Click **Play** again to play the disconnect macro.

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

Editing a macro

If necessary, you can edit your recorded macro in a text editor. To do this:

1. In the Host Access Macro tree, stop recording, then right-click either the Connect Macro, Data Macro, or Disconnect Macro node and select Modify.
2. Click Edit macro... in the Global Macro Information window.
3. Make the necessary changes to the text in the Edit Macro window, then click OK.
4. Click OK on the Global Macro Information window to save your changes.

Defining global screens

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

For example, if you are recording a 3270 VM application logon sequence as a macro, you might notice that messages sometimes display after you type the user ID and password. Each time you see one of these messages, you 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.

Generating 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 Integration Object to avoid overwriting existing objects on the Server when you use Host Publisher Studio to transfer your application to the Server.

Securing passwords and other sensitive data in macros

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

- They can be hard-coded in the macro; for example, for an iSeries or VT host where the same user ID and password can be used several times to log on to an application simultaneously. When you record a macro using Host Access, you may notice that hidden fields on the Host On-Demand terminal are placed in the Macro Tree as asterisks; however, when the macro is written out to the file-system, these fields are plain text and your sensitive data is visible. When it is read back in, it again becomes asterisks.
- The user ID and password parameters can be provided externally (for example, by using an HTML form in the Web environment), and used to set the Integration Object's input properties that represent those macro parameters. To set up this option, select Insert Input Variable in Host Access.
- The user ID and password can be defined in a user list, which is a list of user IDs and passwords, defined in a configuration file and managed as an in-memory list by Host Publisher Server. To define a user list in Host Access, refer to the following section, "About user lists" on page 27.

- The user ID and password parameters can be provided using express logon. Express logon uses a Digital Certificate Authentication Server (DCAS) to dynamically supply a user ID and one-time-use-only passticket, for example, password. To enable express logon, select Express Logon Enabled in Host Access. See "Express logon" on page 80 for more information.

About 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 various user accounts it can access to connect. The accounts have associated user IDs and passwords.

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

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

ID	Password
pam	pampw
sarkar	sarkarpw
villari	villaripw

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

To define a user list, select the Connection Pools tab in the left pane and the User List Configuration tabs in the right pane. This list of user IDs and password pairs provide values for macros to use at runtime when connecting to the data source. This data can be extended to include other sensitive data, such as PIN codes and other access keys.

To include the user ID in your connect macro, select the Macro tab and begin recording your macro. The Integration Object you are recording must use a connection pool that has the user list defined with the user ID you want to insert. Navigate to the place where you want to enter the user ID, select Insert User ID, then select the user ID you want to use for this recording session.

The macro tree will show the entry as _userid. To access a previously-defined user in a user list from the connect macro, this Integration Object must use a pool that is associated with this user list.

To include the password, navigate to the place where you want to enter the password, then select Insert Password. The password associated with the user ID you previously selected will be automatically inserted. The macro tree will show the entry as _password.

Note: When WebSphere's cloning capabilities are used to create multiple instances of WebSphere JVMs running Host Publisher Server that share the same application files, the following restriction applies:

- You cannot use user lists. Each JVM running Host Publisher Server reads the user lists associated with deployed applications and assumes it has each user list for its exclusive use, but that is not true for cloned configurations.

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

Creating an Integration Object for database access

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

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

- Using the wizard ("Using the wizard")
- Defining database connections ("Defining database connections" on page 29)
- Retrieving information from a database ("Retrieving information from a database" on page 29)
- Generating an Integration Object ("Generating an Integration Object" on page 29)
- Verifying a SQL statement ("Verifying a SQL statement" on page 30)
- Creating connection pools for Database Access ("Creating connection pools for Database Access" on page 30)

Using the wizard

When you use Database Access to create an Integration Object, it launches a wizard to guide you. To start Database Access, open the Host Publisher Studio application, then click **Create** > **Database Access** > **Integration Object**. The wizard starts automatically. 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 Integration Object is created when you save the file or when you select **Finish**.

Defining database connections

Before you can build your SQL statement to access the data you want to publish, you need to connect to your database. Select a database driver, then replace the [ ] with the name of your database, and type any required user ID and password. Select Connect to connect to your database.

Notes:

1. Host Publisher does not install or set up JDBC drivers. You must ensure that the drivers are installed on the machine running the Host Publisher Studio.
2. If you cannot connect to your database, define the paths to the JDBC drivers on the —classpath statement of the dbaccess.bat file. The dbaccess.bat file is located in the *install_dir*\Studio directory, where *install_dir* is the directory in which Host Publisher is installed.
3. If you launch Database Access from the Host Publisher Studio, define the paths to the JDBC drivers on the —classpath statement of the webbridge.bat file. The webbridge.bat file is located in the *install_dir*\Studio directory, where *install_dir* is the directory in which Host Publisher is installed.

Retrieving information from a database

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

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

The SQL statement types are:

- Select
- Select Unique
- Insert
- Update
- Delete

After Database Access guides you through creating an Integration Object, you can use the 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 use the SQL statement you created to access the data that will be returned to the Web browser.

Verifying a SQL statement

To verify your generated SQL statement, click Run SQL... on the SQL tab. If successful, a result window displays your results with a maximum number of records. You can specify the maximum number of records to display by clicking Run SQL Settings... This maximum is only used by Database Access to run a sample of your SQL statement. The maximum will not be used when your Integration Object executes on the Server; all of your records will be returned when executing on the Server. Setting a maximum number of records to display using Database Access will avoid some out of memory conditions because your Server is likely to have more memory than your Studio machine.

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

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

Creating connection pools for Database Access

You can use the Connection Pools tab in Database Access to define connection pools. You can also use the Connection Pools tab to modify attributes of the pools, including connection configurations and user lists.

For each Integration Object, Database Access allows you to create a new connection pool, or to select to use an existing connection pool that you have already defined.

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

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

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

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 create a page and imports the Integration Objects for you. You can also use Host Publisher Studio to import EJB Access Beans and other Java Beans or objects.

This section provides information to help you with the following tasks:
- Using wizards ("Using the wizards")
- Specifying Integration Objects to publish on the Web ("Specifying Integration Objects to publish to the application server" on page 33)
- Choosing properties to render and control appearance ("Choosing properties to control appearance" on page 34)
- Previewing a page ("Previewing a page" on page 34)

For information on using the JSPMigration utility to migrate applications to 1.0 JSPs, see "Migrating JavaServer Pages (JSPs)" on page 34.

Using the 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 and other Java objects into the application, and creating pages for publishing data.

You can select to create a new application or modify an existing application. To create a new application, specify an application name and follow the wizards using the Next and Back buttons at the bottom of the wizard windows. You can create an entire working application using the New Application wizard. Use the buttons at the bottom of the windows to move from one wizard to another. During creation of Web pages in Host Publisher Studio, some of the windows contain multiple layers of buttons. Carefully review the instructions presented at the top of each Studio Wizard window. The required steps are listed in chronological order. Select Next only after you complete all of the actions required on the current window and are ready to move to the next step in the Web page creation process. Select Back when you want to return to the previous step in the Web page creation process. Select Finish only when you are ready to leave the Host Publisher Studio wizard.

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

The Studio wizards are:

New Application wizard
This wizard guides you to name the new application and select the pages and Integration Objects to add to the application. You can use this wizard to create an entire functioning application, or click the Finish button at any time to exit the wizard. If you exit before completing the wizard, use the pull-down menus on the Main Panel to complete your application. This wizard is launched by the New Application item on the File pull-down menu.

New Integration Object Wizard
This wizard guides you to add an Integration Object to a Web page, resolve the Integration Object inputs, and render the outputs. This wizard is launched by the New Application wizard or by the Insert Integration Object... item on the Insert pull-down menu.

New Page Wizard
This wizard guides you to create a new page, name the page, and specify its function. You identify the resources to add to the page, and whether the resources provide input to an Integration Object or render the output of an Integration Object. This wizard is launched by the New Application wizard or by the HTML Page item on the Create pull-down menu.

Insert Output Control Wizard
This wizard guides you to add an output control to the current page. As part of creating the output control, you specify the Integration Object output to be displayed with the control. This wizard is launched by the New Page wizard, the New Integration Object wizard, and by various items on the Insert pull-down menu.

Insert an Input Wizard
This wizard guides you to add an input control to the current page. When creating the input control, identify the form that contains the new input control and the destination of the submit action for that form (if you are creating a new form). This wizard is launched by the New Page wizard, the New Integration Object wizard, and by various items on the Insert pull-down menu.

New Error Page Wizard
This wizard guides you to create an error page to display to the client

when an Integration Object error is encountered. If you do not create an error page, a default error page is provided for you on Host Publisher Server. Use the error page to inform the client that an error occurred and how to address the problem, such as calling a service telephone number. When you create an error page, all the Web pages in your application containing Integration Objects are updated with the error page name as an input parameter to the Integration Objects. If you add Integration Objects to new or existing pages, the error page name is added as an input parameter to those Integration Objects, also. If an error page already exists in the application when you create a new error page, the new error page name referenced by the Integration Objects replaces the previous error page name. The previous error page remains in the application unless you remove it. You can use more than one error page in your application, but the last error page you create is referenced by default unless you change the reference for a specific Integration Object. This wizard is launched by the New Application wizard or Error page item on the Create pull-down menu.

Transfer to Server Wizard
To launch this wizard, select File > Transfer to Server. This wizard guides you through the process of transferring application parts to Host Publisher Servers. All Host Publisher Servers to which you want to transfer your application must be configured using the Host Publisher Server Definition window. To get to this window, select Options > Preferences, then select the Servers tab, or click Server Info on the first page of the Transfer to Server wizard.

Specifying Integration Objects to publish to the application server

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

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

JSP tags are designed for any Java object or bean. You can use the Host Publisher Studio to import other Java classes or beans, in addition to Host Publisher Integration Objects, into your application 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 (JSPs), 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.

Previewing a page

When you are creating an HTML or JavaServer page (JSP) 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, Java code, and associated Java objects are not shown.

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.

Showing Integration Object input also as an output

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 to use as input to an Integration Object on another page. The second page searches for a telephone number for the person and uses the output method for the name provided as input. You could construct a sentence on the page like this:

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

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

Migrating JavaServer Pages (JSPs)

Because Host Publisher Version 3.5 no longer supports 0.91 JSPs, migration of pages created by a previous version of Host Publisher is required; however, you can choose when and how you migrate:

- When Host Publisher Server is installed.
- Using the command line on Host Publisher Server.
- When an application using 0.91 JSPs is opened in the Host Publisher Studio.
- When a 0.91 JSP is imported into the Host Publisher Studio.
- Using the command line on Host Publisher Studio.

Host Publisher provides a migration utility called JSPMigrator.

Migration Tool: JSPMigrator

JSPMigrator has the following capabilities and options.

- Saves the original JSP file in the same directory with an .old extension.
- Creates JSPMigrator.log (or a specified file name provided on the command line), in the directory from which JSPMigrator is invoked. This log file lists:
  - The name and location of all the converted JSP files
  - The line number and name of each modified tag within the converted JSP files
- Can migrate:
  - A page
  - All pages in a directory
  - All pages in a specified directory, including subdirectories
- Allows modification of the source code to add function for unsupported tags. Point your browser to *install_dir*\SDK\JSPCustomMigrator\Introduction.html for instructions on how to modify the JSPCustomMigrator.java file.

  Note: If you choose to modify the JSPMigrator source code, the modified code is not available during Host Publisher Server installation or by Host Publisher Studio. The modified version of the code can be executed only from the command line.

The JSPMigrator file is located in the following directories:

On the Host Publisher Studio machine
`install_dir\Studio\`, where *install_dir* is the directory in which Host Publisher Studio is installed.

On the Host Publisher Server machine

Windows platforms
`install_dir\Server\`, where *install_dir* is the directory in which Host Publisher Server is installed.

AIX `/usr/lpp/HostPublisher`

Solaris
`/opt/HostPublisher`

OS/400
`/QIBM/ProdData/HostPublisher/Install`

The JSPs and .jar files are located in the following default directories:

Windows platforms

.jsp files

install_dir\Server\production\documents\MyApplication\

where *install_dir* is the directory in which Host Publisher Server is installed, and *MyApplication* is the name of your application.

.jar files

install_dir\Server\production\beans\

AIX and Solaris

.jsp files

/var/HostPublisher/Server/production/documents/*MyApplication*/

where *MyApplication* is the name of your application.

.jar files

/var/HostPublisher/Server/production/beans/

Migration of tags and page directives

Following is a list of 0.91 JSP tags and how they will be migrated.

<BEAN>

Replaced with the JSP 1.0 <jsp:useBean> tag

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

<INSERT></INSERT>

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

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

<REPEAT></REPEAT>

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

<REPEAT> is replaced with:

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

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

</REPEAT> is replaced with:

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

```
  {
   break;
  }
 }
%>
```

<%@ content_type=″text/html;charset=ISO-8859–1″ %>
Replaced with the following JSP 1.0 syntax:

```
<%@ page contentType="text/html;charset=ISO-8859–1" />
```

In Host Publisher-created error pages:

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

If you have added 0.91 JSP tags to Web pages in your applications other than those described here, those tags are not migrated to 1.0 JSP tags.

Note: The log file will contain informational messages concerning tags that you might need to migrate manually. You must determine if these tags need to be migrated.

Running JSPMigrator from the command line

To run JSPMigrator from the command line:

Windows platforms

```
JSPMigrator -s[r] "jsp files/paths" -j "jar files/paths" -l log file
```

AIX

```
./AIXjspmigrator.sh -s[r] "jsp files/paths" -j "jar files/paths" -l log file
```

Solaris

```
./SUNjspmigrator.sh -s[r] "jsp files/paths" -j "jar files/paths" -l log file
```

OS/400

```
./jspmigrator.sh -s[r] "jsp files/paths" -j "jar files/paths" -l log file
```

- Where *jsp files/paths* is a list of JSP files and paths to convert
  - **-s** means that all JSPs in the path, excluding subdirectories, are migrated.
  - **-sr** means that all JSPs in the path and all subdirectories are migrated.
  - If you specify a list of files and paths, you must enclose them in double quotation marks and separate them with semi-colons; for example:

    ```
    JSPMigrator -s "jsp91.jsp;jsp91_2.jsp"
    ./AIXjspmigrator -s "jsp91.jsp;jsp91_2.jsp"
    ./SUNjspmigrator -s "jsp91.jsp;jsp91_2.jsp"
    ```
- Where *jar files/paths* is a list of .jar files and paths to search

- The .jar file is searched for method names required to convert the <insert> tag. If only a path is specified, a search is done for all the .jar files in that path. If no .jar file is specified, a search is done for all the .jar files in the current path.
- If you specify a path, the path and all subdirectories are searched.
- If you do not specify a path, the current directory and subdirectories are searched. It is best to specify directories that contain .jar files. If you are not sure, specify the drive name where Host Publisher is located.
- If you specify a list of files and paths, you must enclose them in double quotation marks and separate them with semi-colons; for example:

```
JSPMigrator -s "jsp91.jsp;jsp91_2.jsp" -j "my_JAR_path;my_JAR_path_2"
./AIXjspmigrator -s "jsp91.jsp;jsp91_2.jsp" -j "my_JAR_path;my_JAR_path_2"
./SUNjspmigrator -s "jsp91.jsp;jsp91_2.jsp" -j "my_JAR_path;my_JAR_path_2"
JSPMigrator -s "jsp91.jsp;jsp91_2.jsp" -j "my_JAR_path;my_JAR_path_2"
```

- Where ***log file*** is the log file
  - If you do not specify a path, the log file is saved in the same directory as JSPMigrator.
  - If you specify an existing file, log information is appended to the existing file.
  - If you omit the log file parameter, JSPMigrator.log is created or appended to in the current directory.

To get command-line help for JSPMigrator, type

```
JSPMigrator -?
./AIXjspmigrator -?
./SUNjspmigrator -?
JSPMigrator.sh -?
```

Host Publisher includes the sample source code for the migration tool in the *install_dir*\SDK\JSPCustomMigrator\ directory (where *install_dir* is the directory in which Host Publisher is installed). HTML instructions for modifying the JSPCustomMigrator.java file are included in the *install_dir*\SDK\JSPCustomMigrator\ directory. To view the instructions, point your Web browser to *install_dir*\SDK\JSPCustomMigrator\Introduction.html.

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

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

Sequencing Integration Objects on a single page

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

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

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

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

Sequencing Integration Objects between non-adjacent pages

You can create a composite application similar to the one described in "Sequencing Integration Objects on a single page" on page 39, but the Integration Objects reside on different pages and they do not require user input. The output of one Integration Object is stored within the Web session and retrieved by another Integration Object.

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

Integration Object. This method requires no interaction with the end user and does not demand that the two pages be adjacent to each other in the logical page hierarchy in your web site.

To create this type of composite application 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. To create a <FORM> tag and to enable session transfer to take place, you must render at least one variable as a control that will serve as both output and input; for example, list box, password field, and so forth. This excludes tables and normal text, which do not allow selection. The created form must specify that the expected page of the second Integration Object (execute2) will appear when the form is submitted. This links the pages as parent/child, makes the data from the first Integration Object available to that page, and enables the expected button for the next step.
4. For the second Integration Object, create a new page execute2, then define an input of this Integration Object that will be satisfied by an Integration Object output. Select an output variable of the first Integration Object.

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

Importing Java objects

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

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

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

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 contained in user lists, such as passwords. Any data in a user list except the user ID can be encrypted. You can choose to:

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

Encryption requires that you specify an encryption password. You are prompted for a password to be used for strong encryption. When a Web application containing a strongly-encrypted user list is deployed on the Server, the password you specified for strong encryption must be specified when the Server is restarted. Without this password, the Server cannot be restarted. This password is also required by Host Publisher Server Administration when an administrator modifies the strongly-encrypted user list. If another user list is created with a property that requires strong encryption, the same password must be specified to encrypt that user list. The Host Publisher Server can only have one startup password, so the same password must be used by all strongly-encrypted user lists. If weak encryption is used, no password is required.

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.

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

Here are some examples of target directories by platform:

AIX	/var/HostPublisher
Windows NT	C:\HostPub Note that FTP does not allow the use of a drive letter as a destination. You must configure the FTP service you are using on Windows NT to have an alias (such as /HostPublisher) point to C:\YourInstallDirectory. You then specify /HostPublisher as your target directory for this server.
Solaris	/var/HostPublisher
OS/400	/QIBM/UserData/HostPublisher

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

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

If you publish an application to a Server that contains a file with the same name as an existing file, a warning message displays. You then have the options to:
- Continue the file transfer and overwrite the existing file.
- Stop the file transfer and keep the existing file.
- Give general permission for overwrites for the current and any future existing files found during this transfer to the server.

Application transfer places the application files in a staging directory, which is cleared when an administrator deploys the application. The situation occurs only when two applications with the same name, or applications with shared parts, are transferred before one of them is deployed. Continue with the transfer and overwrite the existing files *only* if you are sure the application files are consistent with each other and can later be successfully deployed.

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.

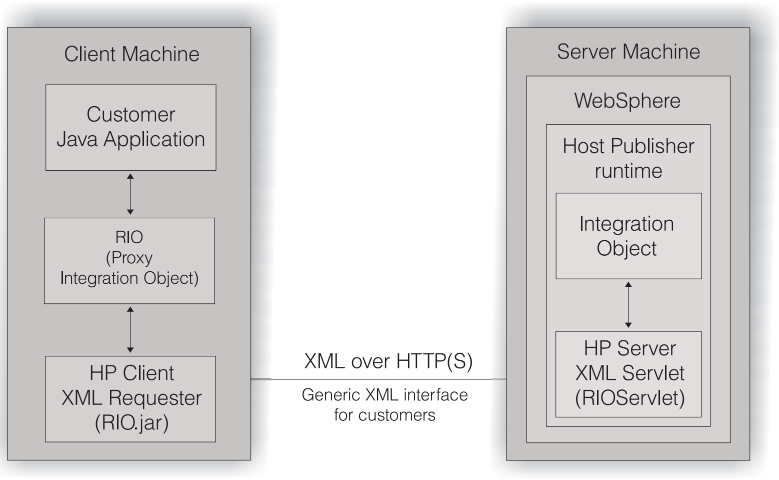
Trace information is written to separate files for each component:

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

The trace files will be located in the Studio directory.

Remote Integration Objects

You may want to write a program that does not execute in WebSphere but that requires access to an Integration Object. You can use Remote Integration Objects (RIOs) to access Integration Object data from a Java program (applet or application) running on a remote machine. The remote machine requires lightweight RIO .jar files and network access to a Host Publisher Server; however, it does not require the Host Publisher Server or WebSphere Application Server.


Client Machine
Customer Java Application
RIO (Proxy Integration Object)
HP Client XML Requester (RIO.jar)
XML over HTTP(S)
Generic XML interface for customers
Server Machine
WebSphere
Host Publisher runtime
Integration Object
HP Server XML Servlet (RIOServlet)

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

The protocol used for communication between the proxy object and the servlet uses XML to encapsulate the method invocation and the input parameters and output parameters. The XML flows over an HTTP(S) connection. Since XML is used to interface with the RIO servlet, you can develop an XML application, written in Java, Perl, C++, and other languages, to access the servlet to execute your Integration Objects. These applications would not require any of the RIO proxy object or the RIO.jar file on the client, but would require HTTP access to the RIOServlet and an XML parser to process.

See the *IBM WebSphere Host Publisher Programmer's Guide and Reference* for more detailed information on RIOs.

RIO properties

The Studio Options menu contains a Remote Integration Objects Properties selection. A default prefix is provided; however, you can specify a different prefix for each Remote Integration Object you create. If you open a different Integration Object, the prefix defaults to the value you specified when you created it.

To change the default value, used by new Integration Objects, for this property, edit RIO_NAME_PREFIX in Studio.ini.

Using WebSphere Studio with Host Publisher Studio

Integration of Host Publisher with WebSphere Studio is flexible, in that you can choose how to divide work between Host Publisher Studio and WebSphere Studio. In general, Host Publisher Studio is fine-tuned for building Web-to-host applications, whereas WebSphere Studio is a more general-purpose studio for building and managing a wider variety of Web applications.

There are two things you can do using WebSphere Studio:

You can insert a Host Publisher Integration Object into a WebSphere Studio project.

To do this, update the WebSphere classpath to include the following files, located in the *Server_install-dir*\Common directory:

- HpRte.jar;
- log.jar;
- HPubCommon.jar;
- habeansnlv.jar;
- elf.jar;
- HostPubElf.class; (if your Integration Objects are express logon enabled)
- xmlLegacyPortal.jar;
- sslight-ex11-rsa-des.zip;

After you use Host Publisher Studio to create your basic Web-to-host application, you might want to use WebSphere Studio to manage Host Publisher Studio-generated JSPs.

- Create the application with Host Publisher Studio, then transfer it to Host Publisher Server.
- Import the JSPs into your WebSphere Studio project.
- Modify the JSPs within WebSphere Studio.
- Publish the modified JSPs from WebSphere Studio to Host Publisher Server.

For detailed information on how to perform these tasks, refer to:

```
http://www-4.ibm.com/software/webservers/hostpublisher/library
/whitepapers/wsstudio/index.html
```

You can build a Web application in WebSphere Studio using Host Publisher Integration Objects

Once you create your Host Publisher application, you can use WebSphere Studio to enhance the JSPs. You can use WebSphere Studio's Java bean wizard to build Web applications with Host Publisher Integration Objects.

- Create the Integration Objects with Host Publisher Studio, then transfer them to Host Publisher Server.
- Import the Integration Objects into your WebSphere Studio project.
- Create your Web application within WebSphere Studio using WebSphere's Java bean wizard.
- Publish the Web application to Host Publisher Server using WebSphere Studio.

For detailed information on how to perform these tasks, refer to:

```
http://www-4.ibm.com/software/webservers/hostpublisher/library
/whitepapers/wsstudio/index.html
```

Chapter 3. Administering Host Publisher

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.

Host Publisher Server administration is Web-based. You perform administration tasks using a Web browser, through a user interface provided by the Host Publisher Administration servlet, **HPAdminServlet**. Host Publisher Server administration allows you to monitor server status, manage licenses, administer connections and connection pools, manage user lists, administer applications, perform problem determination, and administer the XML Legacy Gateway.

Distributed administration

Distributed administration allows you to perform administration tasks on instances of Host Publisher Server running either in JVM clones defined using WebSphere, or in JVMs with an EJB container but no Web container.

In Host Publisher Version 2.X, the administration task required an instance of HPAdminServlet in every WebSphere JVM in which Host Publisher Server ran. The servlet managed the Host Publisher Server instance in that JVM by making local method calls to the server objects and classes. This is no longer possible in WebSphere 3.5, the current version of WebSphere, because:

- When cloned JVMs are defined within or across machines, an attempt to access the URL for the Host Publisher administration servlet cannot be directed to a specific JVM, because every JVM is identical. The request is not necessarily routed to the desired JVM.
- A Host Publisher user can run Integration Objects in a JVM that contains only an EJB container, with no Web container. However, HPAdminServlet cannot run in such a WebSphere JVM.

To overcome these limitations, the Host Publisher Server administration function in Host Publisher Version 3.5 consists of two components:

- A Java Remote Method Invocation (RMI)-based Host Publisher *administration server* that is created in every Host Publisher Server instance that is initialized in a WebSphere JVM.
- An *administration client*, HPAdminServlet, that can run in any WebSphere JVM and can manage a Host Publisher Server instance in another

WebSphere JVM by making RMI calls to its administration server. HPAdminServlet is configured within the Host Publisher Application Server JVM (HostPubServer) which is defined, using WebSphere configuration, when you install Host Publisher Server.

Most functional and user-interface aspects of Web-based administration have not changed significantly since Host Publisher Version 2. Before you can perform any administration tasks, however, you must select a Host Publisher Server instance to administer on a given server machine. The default is the Server instance running on the same JVM as HPAdminServlet. After you select the Server instance, all administration functions are assumed to apply to that Server instance until the selection is changed.

The following categories of administration tasks are allowed for the Host Publisher Server instance you are managing:

- Selection of a server, and an instance of Host Publisher Server in that server, to administer
- Monitoring server status: starting and stopping Host Publisher Server, and supplying passwords during startup
- License management: changing the allowed number of licenses on a server running one or more instances of Host Publisher Server (JVMs)
- Connection and connection pool administration: displaying pool definitions, pool status, and the status of active connections for Host Publisher Server
- Management of user lists
- Application administration: deploying applications, deleting applications, and deleting files that are no longer needed
- Problem determination: viewing trace and log files, and setting various options for tracing and logging
- XML Legacy Gateway administration: configuring the XML Legacy Gateway to access hosts and host applications, and creating a portal for accessing hosts and host applications.

Using Host Publisher Server Administration

Accessing 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***/***hp_alias***/HPAdmin/main.jsp** (where *server* is your server name and ***hp_alias*** is the alias chosen during installation of Host Publisher Server).

Naming an instance of Host Publisher Server

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

The JVM name consists of two parts, separated by an underscore (_) character.

- The first part is the value of the Java system property hostpublisher.server.name, defined as a JVM startup parameter (-D) in the WebSphere Application Server definition. If this property is not defined, the default value used is *HostPubServer*.
- The second part is the unique port number configured for the WebSphere OSE protocol for that JVM's servlet engine. If no servlet engine is configured, a random port number is generated, using a facility provided by the TCP/IP stack to ensure that the port number will not be used by any other program on that machine.

  Note: For more information about OSE, refer to "Using Host Publisher in a remote open systems environment (OSE)" on page 79

For example, in a WebSphere JVM where the value of the hostpublisher.server.name property defined for WebSphere Application Server is *HostPub*, and the OSE port number for the JVM's servlet engine is *8994*, the name of Host Publisher Server on that WebSphere JVM is *HostPub_8994*.

Select Server and JVM

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

Server Status

This window displays information about the current status of the server, including the number of licenses installed on this server. You can stop or restart the server from here. If a deployed application uses express logon, you must type the Host Publisher keyring password for express logon to start or restart the server. If you have user lists requiring strong encryption, you must type the Host Publisher encryption password to start or restart the server. For more information on the key ring password, see "Express logon" on page 80.

License Management

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

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

To enable the license tracking option, set the licenseTracking property in the server.properties file from 0 to 1. The file license1.txt is located in the Log subdirectory under your Host Publisher installation directory.

For detailed information about editing the server properties files, see "Appendix C. Server properties files" on page 161.

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

Note: When Host Publisher is used in the Web Access environment, Client Access licensing is used, which does not work as described above. You will not see license violation and tracking information in Host Publisher logs. Refer to WebSphere Application Server iSeries documentation for more information.

Connection Pools

Connection pools are collections of communication links to backend data sources, such as 3270 applications or databases. When an Integration Object is run on behalf of a client request, the Integration Object obtains an available connection from a pool, uses it for access to the data source, then returns the connection back to the pool for reuse by another Integration Object. Pooling of connections allows the overhead of establishing the connection to be absorbed in its first use. Each Integration Object reusing this connection benefits from the prior establishment of the connection and can run faster.

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

Pool Definitions

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

This window displays pool definition information for host and database connections.

User Lists

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

User List Members

User List Members provides server and password information about users in selected pools.

Administering applications

To publish an application, it is transferred the server, then deployed. When an application is transferred, it is copied from Host Publisher Studio to Host Publisher Server, but it must be deployed before it can be used.

List Applications

This window displays 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, and the list is sorted by application name, regardless of its status. 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

This window displays a sorted list of all applications transferred to the server from which you can select one or more for deployment. The deployed applications move to the Server’s production area and become available for use. By default, applications must be deployed manually. If you want an application automatically deployed when the server starts, you must change the autoDeploy property value in the server.properties file. See “Appendix C. Server properties files” on page 161 for information about the server properties files.

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 to the production area for immediate use. The staging area is then cleared.

Some examples of problems are: errors in connection pool configuration files, missing files, conflicting information, or applications already deployed. Refer to “Chapter 8. Troubleshooting” on page 95 for help in identifying problems and determining solutions.

If one or more of the applications selected for deployment contain strongly-encrypted user list data, you must enter the Host Publisher encryption password in the entry field provided. If you do not, only selected applications that do *not* contain strongly-encrypted user list data are deployed; however, you will receive a message informing you that one or more applications were not deployed because they require the encryption password.

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 pools will be activated immediately.

Notes:

1. The Host Publisher encryption password is defined during the application creation process in the Host Publisher Studio, and allows Host Publisher Server to encrypt and decrypt user lists.
2. 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.
3. If you deploy an application that has already been deployed, WebSphere Application Server will stop and restart all servlets in that JVM. This might cause a disruption in service.
4. In a cloned environment, if you want all clones to access an application that has been deployed to one clone, you must recycle the Host Publisher Server on all the JVMs. For cloning across machines, you must transfer to all machines.
5. If the application you want to deploy uses express logon, you must type the Host Publisher keyring password for express logon to deploy the application. Additional considerations for express logon are found in “Express logon” on page 80.

Remove 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 Studio's Transfer to Server), 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.

Logging and tracing components

The base log and trace file names in server.properties are used as templates to generate unique sets of log and trace files for each application server (JVM). The default base trace file name is trace.txt; the default base log file name is messages.txt. These names can be changed in server.properties. The JVM running Host Publisher is the concatenation of the following values:

- The value of the hostpublisher.server.name property defined to the JVM (the default is HostPubServer if the property is not defined)
- The underscore (_) character
- The unique port number configured for that JVM's servlet engine (a random port number is generated if there is no servlet engine configured)
- Another underscore (_) character

The JVM ID (for example: *SSSS_PPPP_*) is then appended to the base file name to generate the template for the log and trace files for a JVM; using the trace file as an example, this becomes trace_*SSSS_PPPP_*.txt. Finally an index is added to this name (1, 2, 3, and so forth) to distinguish multiple files. Therefore, if the port number for the application server running Host Publisher is 8120 (the configured default), and the application server name is HostPubServer (the configured default), the trace file for that application server is named trace_HostPubServer_8120_.txt. With multiple trace files configured, the trace file names for this JVM are trace_HostPubServer_8120_1.txt, trace_HostPubServer_8120_2.txt, and so forth.

When trace_*SSSS_PPPP*_1.txt reaches maxTraceFileSize, it is closed and renamed to trace_*SSSS_PPPP*_2.txt. A new trace_*SSSS_PPPP*_1.txt is opened.

When trace_*SSSS_PPPP*_1.txt reaches maxTraceFileSize again, it is renamed to trace_*SSSS_PPPP*_2.txt, trace_*SSSS_PPPP*_2.txt is renamed to trace_*SSSS_PPPP*_3.txt, and a new trace_*SSSS_PPPP*_1.txt is opened.

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

The log file

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 B. Server error messages and recovery actions" on page 117.)

The trace file

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

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

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.

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. Error events are always written to the log file. See the explanation for file naming under Trace File Name below. The same applies to the log file, which has a template name of messages.txt.

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

Trace File Name
The trace file name template is trace.txt and is created separately for each JVM. The JVM name is added as a suffix to each file name, in addition to the suffixes used to identify the number of files; for example, if the JVM name is HostPubServer_8120, and the trace file

name template defined in server.properties is f:\HP3\Log\trace.txt, and the number of trace files is set to 2, the trace files are named as follows.

f:\HP3\Log\trace_HostPubServer_8120_1.txt

f:\HP3\Log\trace_HostPubServer_8120_2.txt

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

You can edit server.properties to change only the template name of trace.txt.

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 updates to the terminal screen

Transport tracing
Traces the network connection

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
Enables tracing in Host Publisher Server for 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.

Setting up the Display Terminal

Host Publisher Server Administration includes a tracing function that enables you to view the host terminal screen (*green screen*) of any Host Access Integration Object as it runs in real time on the Server. You can enable and disable this function using the Display Terminal check box located in Server Administration. After the trace option is set, terminal settings are shown for only new connections.

For more information on the Display Terminal, see “Using the Display Terminal” on page 60.

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 the server properties files, see “Appendix C. Server properties files” on page 161.

Advanced topics

Securing access to Host Publisher Server Administration using WebSphere Application Server

You can protect a WebSphere Application Server administrative domain that consists of a cluster of servers using the same WebSphere Application Server administrative database. By doing this, certain Host Publisher administrative operations are also protected.

To enable WebSphere Application Server 3.5 user ID and password security:

1. On the WebSphere Administrative Console menu bar, click Console > Tasks > Configure Global Security Settings.
2. On the General tab of the Set Global Security Wizard, select Enable Security.

Refer to the WebSphere Application Server documentation for information on other security tasks.

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

When you use Host Publisher Server Administration to administer a host or JVM (running the Host Publisher server) on a protected WebSphere Application Server, you must provide a user ID and password. This user ID and password must match the user ID and password used for authentication on that WebSphere Application Server.

When WebSphere Application Server security is enabled, the following Host Publisher Server Administration operations require the WebSphere user ID and password.

- Start server
- Restart server
- Shut down server
- Deploy application
- Remove application
- Delete files
- Change the number of licenses
- Delete a connection
- Update passwords on a user list

Opening Host Publisher Server Administration in a new browser window

If you have a browser window open when you open Host Publisher Administration, by default the Host Publisher Administration utility opens in the browser window. If you want to open Host Publisher Administration in a new browser window, you must change your browser settings as follows:

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

Notes:

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

Using the Display Terminal

You can control whether terminal screens are created on the Server display. Display screens are created only for connections established while this option is turned on. They are not created for existing connections, or for connections that are idle. However, the Host Connections page in Host Publisher Server Administration includes a Toggle Display button that turns Display Terminal on and off for selected connections. You can turn the Display Terminal on for any host connection by checking the Display Terminal box on the Host Connection Tracing page; however, you can use Toggle Display to turn Display Terminal on and off at any time, regardless of the Host Connection Tracing setting.

Notes:

1. To use Display Terminal on OS/400, Remote Abstract Windowing Toolkit (RAWT) must be enabled.
2. For Windows NT systems, you must enable the IBM WebSphere Administration Server service to interact with the desktop.

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.

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

1. Configure your Windows NT/9x system to run the Java JDK Remote AWT server/listener.
2. Configure your iSeries WebSphere Application Server JVM, which also runs Host Publisher Server, to send all AWT requests and functions to the Windows server/listener.
3. When setup and configuration is complete, use OS/400 commands to stop and start WebSphere Application Server.

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

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

Chapter 4. Host Publisher Enterprise JavaBeans

Enterprise JavaBeans (EJBs) is a server-side component architecture that enables rapid development of versatile, reusable, portable applications.

EJB technology delivers benefits that address the concerns of enterprise development teams, such as:
- Reduced time to market for mission-critical applications
- Enhanced scalability and portability
- Reduced development cost of enterprise scale applications

Host Publisher Server now provides a Host Publisher EJB deployed in a WebSphere Application Server that is used to run any EJB-enabled Integration Object or Integration Object chain in an EJB environment.

Host Publisher Studio's Host Access and Database Access now include the option to generate EJB support for Host Publisher Integration Objects.

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

Overview

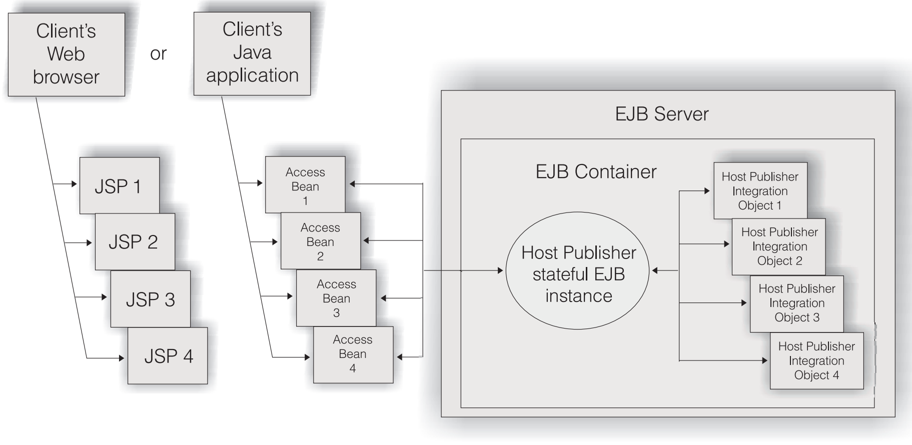
Host Publisher provides support for executing Integration Objects in EJB containers to take advantage of the server-side characteristics provided by the Enterprise JavaBeans architecture. This support consists of the following parts:
- The Host Publisher EJB, which is a stateful EJB capable of running Integration Objects in an EJB environment. This EJB is installed in the Host Publisher application server Default Container of WebSphere when Host Publisher Server is installed.
- Host Publisher Studio support for the generation of EJB support files for running Integration Objects. This includes the generation of an EJB Access Bean for each Integration Object, that provides the same signature as the ″real″ Integration Object.
- Host Publisher Studio support for building applications using the generated EJB support files.

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

- A JSP or a servlet that uses one or more EJB Access Beans in a Web application where the Integration Objects execute in an EJB container.
- A custom EJB that uses one or more EJB Access Beans to execute the Integration Objects in an EJB container.
- A Java application that runs outside a WebSphere execution environment.

Execution flow is as follows:

- The EJB access bean communicates with the Host Publisher stateful session EJB using the remote method invocation/Internet InterORB Protocol (RMI/IIOP), passing it the input properties, and invoking its execution method.
- The session EJB executes the ″real″ Integration Object and packages its output properties, passing it back to the access bean as the result of the execution method.
- These output properties are used to generate values for the output properties of the access bean, which can be subsequently used in a JSP, servlet, custom EJB, or a Java application.


Client's Web browser
or
Client's Java application
EJB Server
EJB Container
JSP 1
JSP 2
JSP 3
JSP 4
Access Bean 1
Access Bean 2
Access Bean 3
Access Bean 4
Host Publisher stateful EJB instance
Host Publisher Integration Object 1
Host Publisher Integration Object 2
Host Publisher Integration Object 3
Host Publisher Integration Object 4

This figure illustrates these concepts where the JSPs are executed in a Web container, the client Java application in a non-WebSphere JVM, and the Host Publisher EJB in a WebSphere EJB container.

Modifying the Host Publisher EJB

During the installation of Host Publisher Server, the Host Publisher EJB is installed in the default container of Host Publisher Application Server, and named HPubEJB or HPubEJBHome. This installed Host Publisher EJB is sufficient to process all EJB-enabled Integration Objects within an EJB environment. However, you might want to modify the Host Publisher EJB and deploy this modified EJB in other environments.

For example, you might want to:

- Prepare a Host Publisher EJB to be deployed in an EJB container other than the one located within Host Publisher Application Server.
- Change some of the default EJB values specified for the installed Host Publisher EJB.

During installation of Host Publisher Server an EJB input .jar file (HPubEJB.jar) and an .xml input file (HPubEJB.xml) are placed in the *install_dir*\Common\EJB\HPubEJB10 directory. The WebSphere Application Server jetace tool is used to customize and create a deployable EJB .jar file using these files. Refer to WebSphere documentation for details regarding the usage of the WebSphere jetace tool, as well as a description of deploying and creating an EJB within a WebSphere Advanced Edition container.

The following .jar files must be in the classpath when invoking the WebSphere jetace tool to process the Host Publisher EJB input .jar file:

1. *Hostpub_install_dir*\Common\HpRte.jar
2. *Hostpub_install_dir*\Common\HPubCommon.jar
3. *WebSphere_install_dir*\lib\servlet.jar

The following are Host Publisher attributes you might want to modify when preparing and creating your custom Host Publisher EJB:

1. **JNDI HomeName**

   This is the network name with which the Host Publisher EJB is registered in the JNDI directory and is used by the client (the Access bean) to locate the EJB. In most cases the default name is appropriate; however, if you want to separate the processing of various Host Publisher EJB requests, you can create multiple deployable Host Publisher EJBs, each with a different JNDI name. Each of these EJBs are separately deployed in different WebSphere Application Server containers on one or more workstations. The JNDI HomeName for the installed Host Publisher EJB is _com_ibm_HostPublisher_EJB_HPubEJB.
2. **Session Timeout**

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

The installed Host Publisher EJB is a stateful session bean. This is necessary to process chained Host Access Integration Objects; however, if your applications do not process chained Integration Objects, you can specify a stateless session bean to take advantage of the increased performance characteristics associated with stateless session beans.

Creating EJB support files for Integration Objects

Host Publisher Host Access and Host Publisher Database Access now include an option to create EJB Integration Object Support. In both Host Access and Database Access, select Options from the toolbar, then select Create EJB 1.0 Integration Object Support. This option must be checked to generate EJB support files for Integration Objects.

The Studio Options menu contains an EJB Integration Object Properties selection. A default suffix is provided; however, you can specify a different suffix for each EJB Integration Object you create. If you open a different Integration Object, the suffix defaults to the value you assigned previously to that Integration Object.

To change the default value, used by new Integration Objects, for these properties, edit the following properties in Studio.ini:

- EJB_PROPERTIES_SUFFIX
- EJB_HELPER_SUFFIX
- EJB10_ACCESS_SUFFIX

When you change a suffix, you change the suffix of only the Integration Object on which you are working.

When you choose to create EJB Integration Object support, EJB support files are generated during creation of the Integration Object. An EJB Access Bean .jar file is created in the IntegrationObjects directory, and the Integration Object .jar file has additional EJB support class files added to it. The default name for the EJB Access Bean .jar file is *IOName*Access0.jar, where *IOName* is the name of the Integration Object.

Note: An EJB Access Bean properties file (for example, *IOName*Access0.properties) is included in the EJB Access Bean .jar file. This .properties file specifies parameter values used by the EJB Access Bean, including host name and port number of the JNDI server and the JNDI name of the Host Publisher EJB to connect to. The default values assume a local JNDI server and a default Host Publisher EJB JNDI name of _com_ibm_HostPublisher_EJB_HPubEJB.

To change the default values:

- Edit the *Hostpub_Install_Dir*\Studio\IntegrationObjects\EJB\AccessEJB.properties file prior to generating the Integration Object (this file is used as input to create the .properties file in the EJB Access Bean .jar file)

or

- Replace the .properties file in the EJB Access Bean .jar file with a modified copy after the .jar file is generated.

Creating a Host Publisher application using EJB Access Beans

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

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

Using EJB Access Beans to transfer, deploy, and process a Host Publisher application

Note: The following information applies only when the Host Publisher application is running on the same server where the Host Publisher EJB is deployed and active. To run the EJB on a different workstation, see "Running the EJB Access Bean on a workstation other than where the Host Publisher EJB is active" on page 68.

Transfer and deployment of Host Publisher applications that are created using EJB Access Beans is the same as transfer and deployment of Host Publisher applications using Integration Objects.

To process Host Publisher applications using EJB Access Beans, you must add the original Integration Object .jar files used by the application to the classpath of the application server where the Host Publisher EJB is deployed and active.

To do this:

1. In the WebSphere Application Server Administrative Console, highlight the application server entry.
2. In the Command line arguments field (located in the right pane), add a -classpath parameter specifying the fully-qualified Integration Object .jar file, or add the fully-qualified Integration Object .jar file to an existing -classpath parameter. For example, if Host Publisher Server is installed at

D:\Hostpub and you have an Integration Object named Fulist.jar, for Windows NT, type **-classpath D:\Hostpub\Server\production\beans\Fulist.jar;** in the input field, or add D:\Hostpub\Server\production\beans\Fulist.jar; to an existing -classpath parameter.

3. Click Apply.

Note: For WebSphere Application Server 3.5 Advanced Edition, changes to application server command line arguments do not become active until you stop and restart the application server.

Running the EJB Access Bean on a workstation other than where the Host Publisher EJB is active

To run the EJB Access Bean on a workstation other than where Host Publisher EJB is deployed and active, modify the EJB Access Bean .properties file to specify the location of the JNDI directory in the network, then copy the modified EJB Access Bean .jar file to the target workstation.

The following WebSphere Application Server 3.5 Advanced Edition and Host Publisher .jar files are required on the target workstation:

- *WebSphereInstallDir*\AppServer\lib\ujc.jar;   (for EJB client code)
- *WebSphereInstallDir*\AppServer\lib\ejs.jar;   (for EJB client code)
- *HostPublisherInstallDir*\Common\HPubCommon.jar

In addition, copy the deployed EJB .jar file to the target workstation; for example, for Windows NT:

```
WebSphere_install_dir\AppServer\deployedEJBs\_wlm_DeployedHPubEJB.jar
```

Note: These .jar files must be present in the classpath to run the EJB Access bean.

Chapter 5. Advanced features

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

Integration Object Chaining

Deciding when to use Integration Object chaining

An Integration Object handles a single task of your Web application. Integration Object chaining enables multiple Integration Objects to execute in sequence, each using the same connection. For Integration Objects created with Host Access, consider that a task is the macro that the Integration Object executes. To determine the number of macros or Integration Objects needed, first list all of the high-level tasks that will be performed. For example, imagine a host application called *fileview* that, when invoked, displays a list of files. From the list of files, a user can select any file by typing a **1** next to its name, press Enter, then view its contents. There are two tasks to perform here:

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

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

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

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

Using Integration Object chaining

Integration Object chaining can only be used with Host Access Integration Objects, which have connections to a terminal-based application, such as a

3270 application. An Integration Object in a chain leaves the connection in a state (at a particular screen) suitable for the next Integration Object in the chain to use from that browser.

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 edit the .jsp files without using the Studio, you must ensure that the order of the Integration Objects is correct.

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

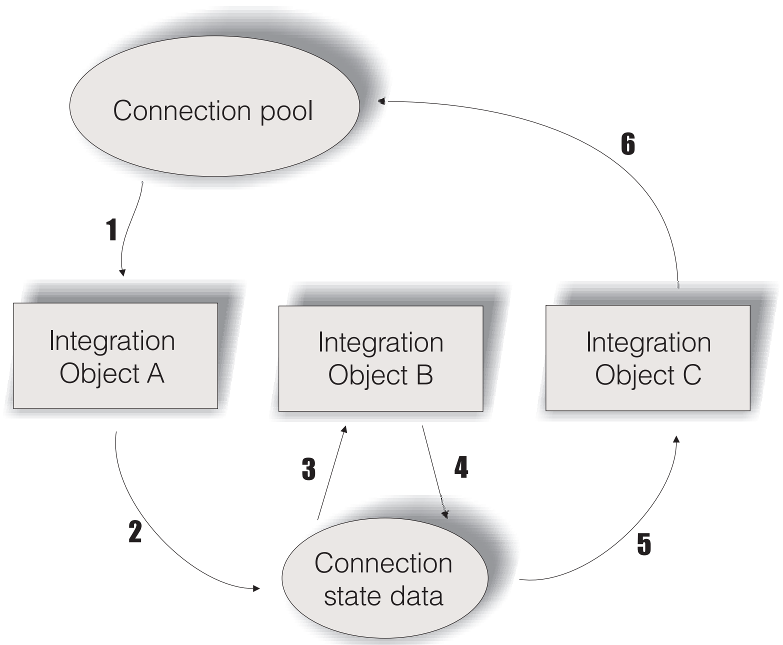


Figure 4. Connection lifetime with chaining

Figure 1 depicts the lifetime of a connection throughout the execution of three Integration Objects. Integration Object A is configured as first, Integration Object B as middle, and Integration Object C as last. Connection state data represents the connection and the state label of the last Integration Object executed.

1. When Integration Object A begins to execute, it retrieves a connection from the connection pool. If connection pooling is enabled and a connection is available, the connection is already logged on and ready. If connection pooling is enabled, but a connection is not available, the connection is created and the connect macro is executed. If connection pooling is disabled, a connection is created and the connect macro is executed.
2. Integration Object A runs the associated data macro, then saves the connection and its current state for the next invocation. You defined this state as the stop state label during Integration Object chaining configuration in Host Access.
3. When Integration Object B begins to execute, it retrieves a connection and its state from the connection state data because it is the middle Integration Object. You must have defined Integration Object B's start state label as Integration Object A's stop state label, which allows these Integration Objects to be chained.
4. When execution completes for Integration Object B, the connection and its state are saved in the connection state data. You defined this state as the stop state label during Integration Object chaining configuration in Host Access.
5. Integration Object C also retrieves the connection from the connection state data. When Integration Object C begins to execute, it retrieves a connection and its state from the connection state data because it is the last Integration Object. You must have defined Integration Object C's start state label as Integration Object B's stop state label, which allows these Integration Objects to be chained. Last-in-chain Integration Objects have no end labels because the connection is always returned to the connection pool.
6. When connection pooling is enabled, the connection returns to the pool; otherwise, the disconnect macro is executed and the connection ends.

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

To build the first Integration Object in the Integration 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 14 for information about defining connections).
2. Record the connect macro. (See "Recording interactions with a host" on page 17 for additional information.)
3. Record your data macro.

4. You will probably not end your data macro at the same point where you began (so that another Integration Object can pick up where you've stopped). Navigate to the desired ending point for the data macro, and click **Stop Recording** on the toolbar.
5. Navigate back to the point where you can disconnect from the host; that is, where the data macro of the last bean in the chain ends.
6. Highlight the Disconnect Macro node in the macro tree, and click **Record**.
7. Record your disconnect macro.
8. Before you create the Integration Object, select Options > Configure Integration Object chaining. On the window that appears, specify a stop state label for this Integration Object and specify that this Integration Object should be **first**. Host Publisher uses start and stop state labels to identify the Integration Objects that may precede or follow this one. For example, if this Integration Object has a stop state label of **connected**, only Integration Objects that have **connected** as a start state label can follow this Integration Object in a chain.
9. Save the Integration Object.

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

To 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 Integration Object chain must use the same connection configuration.
2. Optionally, play the connect macro, or connect manually.

   Note: When you create the middle Integration Object, the full connect and disconnect macros are displayed in the macro tree; however they don't necessarily run for that Integration Object when it is executed on the Server. The connect macro runs before the first Integration Object and disconnect runs after the last Integration Object.
3. You can play the data macros of preceding Integration Objects in the chain or use the terminal to navigate to the correct starting point for this Integration Object. To play the preceding macros, select Play Another Macro on the Options menu. Host Access plays the macros of preceding Integration Objects in the chain that you select. If the macro does not complete successfully, a window appears telling you why the macro did not play. You can play as many preceding macros as you want. Then record your new data macro by selecting Data Macro in the tree and clicking Record.

4. Select Options > Configure Integration 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 Integration Object should be **middle**.
5. 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 Integration Object chain must use the same connection configuration.
2. Optionally, play the connect macro, or connect manually.

   Note: When you create the last Integration Object, the full connect and disconnect macros are displayed in the macro tree; however they don't necessarily run for that Integration Object when it is executed on the Server. The connect macro runs before the first Integration Object and disconnect runs after the last Integration Object.
3. You can play the data macros of preceding Integration Objects in the chain or use the terminal to navigate to the correct starting point for this Integration Object. To play the preceding macros, select Play Another Macro on the Options menu. Host Access plays the macros of preceding Integration Objects in the chain that you select. If the macro does not complete successfully, a window appears telling you why the macro did not play. You can play as many preceding macros as you want. Then record your new data macro by selecting Data Macro in the tree and clicking Record. Be sure to end at the same point where the first Integration Object in the chain started.
4. Select Options > Configure Integration 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 Integration 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 Integration Object chain should return the host screen to the same state from which the first Integration Object started. To verify that this happens, play your disconnect macro.

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, specifies the logical next and previous Integration Objects for the selected Integration Object. This makes it easier for you to identify the order in which the Integration Objects can be used.

When you have finished building your Web application, transfer it to a server, deploy the application, and test it (see "Transferring applications to a Host Publisher Server" on page 43 for information).

Debugging applications that use Integration Object chaining

While Integration Object chaining is a powerful tool for modeling the most complex host applications, take care when assembling your applications. The Host Publisher Studio helps you build Web applications using chained Integration Objects by ensuring that Integration Objects are invoked in the proper order. This does not prevent you from trying to put the Integration Objects in a different order, or from linking pages back to other pages that are earlier in the chain. Because it is easy to create a Web application with Integration Objects that are invoked out-of-order if you do not use Host Publisher Studio to create the application, you should become familiar with the errors you get on the Server when this happens.

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

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

HPS5075 Received STATE_PLAY_ERROR while playing macro in file zzz.macro

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

HPS5035 There is no data source object {0} in HttpSession. A possible cause is the use of multiple browsers from a single machine to a chained application. See documentation for more information.

This error occurs when an Integration Object with a start state label of last Integration Object state fails to retrieve the connection and its state because its start state label does not match the stop state label of the preceding Integration Object.

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

- Are you sure that another Integration Object (either a first or middle Integration Object) has already run and has placed its connection in the state labeled last Integration Object state (its stop state label)?
- Is it possible that the JSP running this Integration Object has been linked to out of order?
- Are you using your own instances of HttpSession objects (through the use of request.getSession(true))? If so, you may be destroying previous instances of the HttpSession object, as well as previous existing chains.
- Is there another Integration Object on the JSP that should have put the connection into the state labeled last Integration Object state and that failed? This will not always prevent other Integration Objects on the page from being invoked. You might be looking at a chain of errors.

In any case, check the message log for the Host Publisher Server and look for reasons for the problem. Refer to message HPS5035 on page 124 for details about this message.

JVM cloning and load balancing in WebSphere

WebSphere Application Server provides support for running multiple JVMs (referred to as application servers in the WebSphere documentation) on a single machine. Each JVM consists of two distinct execution environments: a *Web container* for executing Web applications such as servlets and JSPs, and an *EJB container* for executing Enterprise JavaBeans (EJBs). These JVMs can be *clones* of each other–that is, they consist of the same EJBs and Web applications. For Web applications, all JVM clones can process the same URL request. Alternatively, the JVMs can contain unrelated sets of applications and, for Web applications, they can process different sets of URLs.

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

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

Vertical cloning is also useful when configured on a machine which is powerful enough that a single JVM cannot effectively utilize its CPU power (such as on a multiprocessor machine).

WebSphere provides load balancing mechanisms for forwarding a client's requests across multiple cloned JVMs to distribute the load on individual JVMs evenly. There are two different load balancing mechanisms: one for cloned Web applications, and one for cloned EJBs.

Load balancing of cloned Web applications

HTTP requests directed to servlets or JavaServer Pages (JSPs) running in WebSphere are received by a Web server and forwarded to the WebSphere *plugin*. A plugin is a piece of code that "hooks into" a Web server using a standard Web server-specific programming interface. The WebSphere plugin forwards HTTP requests to a given application server (JVM) using a proprietary protocol called OSE.

OSE uses a variety of inter-process communication mechanisms such as pipes and TCP/IP. OSE is integrated with the load balancing feature of WebSphere. If multiple clones are defined to the plugin to respond to a particular URL, the OSE transport automatically distributes the load across the clones. When OSE runs over TCP/IP connections where the plugin and the JVM are in separate machines, it is referred to as remote OSE. Remote OSE is useful in configurations where Web server machines must be separated from machines running WebSphere Application Servers. For more information about OSE, refer to "Using Host Publisher in a remote open systems environment (OSE)" on page 79.

Load balancing of cloned EJBs

EJB requests directed from a client to an EJB are also load balanced. This type of load balancing is referred to as *Workload Management* (WLM). Load balancing of EJB requests is performed by the "smart" stub code that runs on the EJB client to perform the remote invocation of the EJB. This topic is discussed at length in the IBM Redbook *WebSphere Scalability: WLM and Clustering Using WebSphere Application Server Advanced Edition*, SG24–6153.

Cloning options for the Host Publisher JVM

Because the Host Publisher Server and Host Publisher Integration Objects can execute in any WebSphere JVM, either in a Web container or an EJB container, they exploit all the benefits of WebSphere load balancing and WLM. The Host

Publisher installation process configures a separate WebSphere Application Server in the machine where Host Publisher is installed. The WebSphere application server is called *HostPubServer* and includes two Web applications: *hostpublisher* runs applications generated by Host Publisher Studio, and *_IBM_HP_WebAdmin_* runs server-specific applications. Because WebSphere allows you to run multiple JVMs on a machine, you can create several variations of this basic configuration. You can:

- Vertically clone the basic HostPubServer JVM within a machine to utilize multiple CPUs.
- In addition to vertically cloning the HostPubServer JVM, add horizontal cloning across machines for scalability and fault tolerance.
- Partition the JVMs in a noncloned setup so that they process different URLs representing different Host Publisher applications. For example, if the two JVMs are called JVM1 and JVM2, the hostpublisher application in JVM1 can be configured to respond to the URL *.../HostPublisher/application1*, and the hostpublisher application in JVM2 can be configured to respond to the URL *.../HostPublisher/application2*. This provides a static form of load balancing based on the request pattern. It also protects each JVM from application-related problems in the other JVM.

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

Running chained Integration Objects in cloned JVMs

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

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

To enforce HTTP session affinity, the plugin maps the session ID to one of the JVM clones in such a way that a given session ID is always mapped to the same clone. Whenever a browser includes a given session ID on its HTTP request, the request is routed to the same clone. Thus, Websphere's session affinity feature creates an affinity between a browser with a given session ID and a given JVM among the clones that can handle that request.

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

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

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

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

The following section describes how to use remote OSE to set up cloned JVMs running Host Publisher, (that is, to separate the Web server from the machines running WebSphere and Host Publisher.)

Using Host Publisher in a remote open systems environment (OSE)

WebSphere Application Server uses a proprietary protocol called OSE to communicate between a plugin and the servlet engine. When you run both the Web server and WebSphere on the same machine, OSE is usually run over local pipes. If the Web server and WebSphere are running on different machines, however, you can use remote OSE over TCP/IP.

Remote OSE allows the Web server to run on a separate machine and send requests to one or more application servers running on remote machines. OSE supports clustering and workload management of application servers. This means that the Web server can send requests that require intensive processing to multiple application servers, freeing up the Web server to process more requests.

Refer to WebSphere documentation for detailed information about the administrative tasks required to configure a plugin for Remote OSE and remote application server machines.

In a remote OSE configuration, the Web server and application server run on different machines. When the WebSphere plugin receives a request from the Web server, it determines if the request should be forwarded to the application server, or if it should be passed to the Web server on the local machine. Requests for pages that contain dynamic content, such as JavaServer pages (JSPs), are forwarded to the application server. Requests for pages that contain only static content are passed to the Web server. Since Host Publisher applications can contain both static and dynamic pages, some manual configuration is required on the Web server in a remote OSE configuration.

To set up the Web Server and the default Host Publisher alias, do the following:

1. Create a directory for pages that contain static content, for example, on Windows platforms:.

   ```
   md c:\HPStatic
   ```

2. Add Host Publisher aliases; for example, on Windows platforms:

   ```
   Alias /_IBM_HP_WebAdmin_/ "C:/HPStatic/"
   Alias /HostPublisher/ "C:/HPStatic/"
   ```

3. Copy the Host Publisher application directories from the Host Publisher /Server/production/documents directory on the application server machine to the HPStatic directory on the Web server machine. For Windows platforms copy directories from c:/Hostpub/Server/production/documents to c:/HPStatic.

Host Publisher has two built-in applications (Host Publisher Server Administration, and XML Legacy Gateway) that require both static and dynamic pages. To enable these functions, copy the HPAdmin and xmlLegacyGateway directories from the /production/documents directory to the HPStatic directory. Remember that when new Host Publisher applications containing static pages are deployed, the directories for these applications must be copied to the HPStatic directory on the Web server machine.

Express logon

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

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

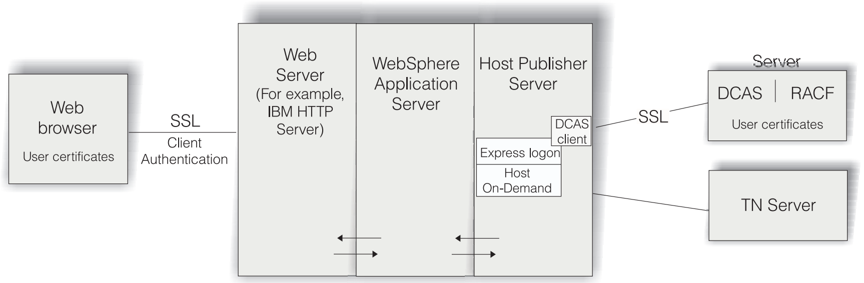


Figure 5. Express logon

The express logon architecture describes the following key components:

- A Digital Certificate Authentication Server (DCAS) component, provided by z/OS, that communicates with an enhanced version of the host-based RACF facility for user ID and password administration. DCAS dynamically supplies a user ID and a one-use-only passticket based on an X.509 certificate and the ID of the host application to which you want to log on.
- A DCAS client that communicates with the DCAS server over a client-authenticated SSL connection, performing a proprietary protocol to perform the function mentioned above.

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

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

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

The connection between the DCAS client and the DCAS server must be a client-authenticated SSL connection. Therefore, to execute express logon-enabled applications in Host Publisher Server, you must use IBM Key Management to create a keyring database, a password–protected Java class file that stores the X.509 certificate used for the client authentication of this connection. You must name the keyring database HostPubELF.class and save it in *install_dir*/Common. On non-Windows server platforms, you must create HostPubELF.class in Host Publisher Studio and transfer it to /Common in Host Publisher Server.

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

Configuring express logon in Host Publisher Studio

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

Configuring express logon in Host Publisher Server

Web browsers and Web servers

Your Web server requires client authentication from your Web browser. Refer to your Web browser's documentation for information on how to set up certificates for your Web browser. Refer to your Web server's documentation to set up client authentication over the SSL session.

IBM Key Management

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

DCAS server

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

RACF

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

WebSphere

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

Installing Host Publisher 3.5 using express logon on an existing installation of WebSphere Application Server 3.5

If you plan to use express logon and you are installing Host Publisher on an existing installation of WebSphere Application Server 3.5, the installation will complete successfully and both WebSphere Application Server and IBM HTTP Server (IHS) start properly. If, however, you then configure SSL with client authentication (required for express logon) through the IBM IHS Administration Server, the IHS loadmodule statements are incorrectly inserted in the httpd.conf file following the WebSphere loadmodule statements. This causes a syntax problem and results in a 1067 error.

To correct this problem, edit the httpd.conf file and move the WebSphere loadmodule statements so they appear after the IHS loadmodule statements.

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

Host Publisher uses Host On-Demand to provide connection support to 3270, 5250, and VT applications using Telnet protocols. Host Publisher uses the SSL support provided by Host On-Demand for securing these connections. Using a secure connection over SSL 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. If you 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.

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

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

To enable server authentication in Host Access, you must build a .class file called CustomizedCAs.class. You must have a copy of this file on the Studio and on the Server. On the Studio, copy this file into the Studio subdirectory (c:\HostPublisher\Studio, for example). For WebSphere Application Server 3.5, copy the file to the *Install_dir*\Server\production\beans directory.

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

To use this utility, type the following command:

c:\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.

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.

Chapter 6. Using the XML Legacy Gateway

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

The XML Legacy Gateway (XLGW) portal consists of:

- **hPubPortal servlet**, the portal servlet that enables access to host links created through Host Publisher Server Administration
- **xmlLegacyGateway servlet**, a Web-based terminal emulator that enables interaction with host terminal applications using XML data formats

Host Publisher XML Legacy Gateway portal page

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

Running on WebSphere Application Server 3.5:
http://*servername*/_IBM_HP_WebAdmin_/hPubPortal

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

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

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

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

Two additional buttons that are unique to the xmlLegacyGateway servlet are Refresh and Disconnect.

Refresh

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

Disconnect

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

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

Configuring time delays for XML Legacy Gateway

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

There are two delays:

Start Delay

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

Interval Delay

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

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

Edit this file with an ASCII editor. Each record in the file describes one session as defined with the XML Legacy Gateway Portal Administrative servlet. Specify values for DELAY_START and DELAY_INTERVAL to override the defaults. Specify the values in milliseconds; for example, a value of 1000 is 1 second.

Enhancing the xmlLegacyGateway servlet

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

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

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

Tracing the xmlLegacyGateway servlet

Host Publisher Server Administration regards the xmlLegacyGateway servlet as an Integration Object. Tracing and error handling for the servlet is therefore the same as for other Integration Objects, as follows:

1. Open Host Publisher Server Administration.
2. Open Server Tracing Options.
3. Check the box next to Integration Object tracing.
4. Click Submit Changes.
5. Highlight the xmlLegacyGateway object.
6. Click Get Objects to view the object.
7. Click Select All.
8. Click Submit Changes.

To see the trace file and error log, in Host Publisher Server Administration, select Problem Determination > View Trace.

Chapter 7. Performance and tuning

System requirements

Host Publisher Studio

Host Publisher Studio requires:

- Intel platforms that support Microsoft Window 95, Windows 98, Windows NT, and Windows 2000
- Pentium processor with a speed of 366 MHz or greater
- Minimum of 256 MB memory
- Minimum of 102 MB free disk space

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

Host Publisher Server

Host Publisher Server is supported on Intel, RS/6000, Sun Sparc, IBM eSeries zServer, and iSeries hardware platforms, and performs best on modern machines that are designed to operate as servers. These machines are typically built for improved scalability and performance. Server machines usually have the capacity for large amounts of memory and offer large Level 1 (L1) and Level 2 (L2) caches.

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

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

Central processing unit (CPU)

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

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

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

For IBM eServer zSeries, Turbo processors provide superior performance. We recommend the G5 and G6 models.

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

We recommend the iSeries models in the following table; however, all recently-announced servers might not be included here.

iSeries Model	iSeries Feature	Processor CPW Rating
820	2398	3200
	2397	2000
	2396	950
740	2070	4550
	2069	3660
730	2068	2890
	2067	2000
	2066	1050
	2065	560
720	2064	1600
	2063	810
270	2253	2000
	820	950

iSeries Model	iSeries Feature	Processor CPW Rating
170	2388	1090
	2386	460
	2385	460
650	2243	2340
	2240	1794
640	2239	998.6
	2238	583.3
530	2162	509.9
	2153	459.3
S40	2261	2340
	2256	1794
S30	2260	1794
	2259	998.6
	2258	583.3
S20	2166	759
53S	2157	509.9
	2156	459.3

Memory

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

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

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

Network interface card

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

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

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

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

Hard drive

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

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

Hardware recommendations

The following table depicts system hardware recommendations for each platform operating as a standalone Host Publisher server.

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

Host Publisher Server Hardware Recommendations			
Platform	**Processor Type**	**Hard Disk Space**	**Mem**
Intel	Pentium 350 MHz or greater	2 GB	2 G
RS/600	PowerPC 375 MHz or greater for uni-processor machines, PowerPC 332 MHz or greater for multiprocessor machines	2 GB	2 G
Sun Sparc	UltraSPARC 333 MHz or greater	2 GB	2 G
IBM eServer zSeries	G4, G5, or G6 series	3 GB	2 G
iSeries	Java-compatible with CPU rating of 260 or greater	2 GB	2 G

Server capacity

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

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

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

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

Chapter 8. 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 and limitations" on page 98. If the symptoms of your problem are listed, follow the recommended actions to resolve the problem.
4. Refer to the Host Publisher support page at:

   ```
   http://www.ibm.com/software/webservers/hostpublisher/support.html
   ```

   for hints and tips and fixes.
5. If you are executing the Studio:
   - Error output for Host Access and Host Publisher Studio are automatically routed to the .con files, HostAccess.con and Studio.con, respectively. Check these files for error information.
   - To route error output for Database Access to a console window:
     a. Execute from a command line.
     b. Modify the invocation in the .bat file to use java instead of javaw.
   - Refer to "Enabling tracing" on page 45 for information about enabling tracing in the Host Publisher Studio.
   - Additional error output may be located in the messages_HostPubStudio_0822_x.txt file located in the *install_dir*\studio directory.
6. If you are executing your Host Publisher application on the Server, check the Host Publisher logs for error messages that indicate the cause of the problem. If the problem appears when starting WebSphere Application Server or the Host Publisher Server, or when accessing the first page of your Host Publisher application, check the WebSphere Application Server error logs for more information about the problem. Use the WebSphere

Application Server problem determination procedures. Refer to “Gathering WebSphere Application Server and Web server problem data” on page 109.

If the error log indicates an internal error and that service should be contacted, first try to recreate the problem. Turn on all Host Publisher Server trace options, as well as tracing for the Integration Object being run. Clear the current Integration Object trace files, then recreate the problem.

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

7. Contact IBM service to resolve the problem. When you report a problem, use the product.xml file to provide information such as product version, current APAR levels, and so forth. This file is located in *install_dir*\Server and *install_dir*\Studio. Refer to “Contacting IBM for service” on page 111.

Host Publisher Server Administration Troubleshooting

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

Server prerequisites and general information

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

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

In addition:

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

WebSphere Application Server

- Make sure that WebSphere is started.
- Make sure that Host Publisher Server is started
  1. Start the WebSphere Administrator’s Console.
  2. Click WebSphere Administrative Domain.
  3. Click your host name.
  4. Make sure that Host Publisher Server is listed as Running.

Configuration

To ensure that your configuration is correct, see "Configuring WebSphere Application Server and your Web server for Host Publisher Server" on page 108.

Secure Socket Layer (HTTP server)

Make sure SSL is configured correctly on the server.

- Are certificates provided?
- Is the SSL port enabled in the Web server configuration file? Refer to your Web server documentation for information on enabling the SSL port.
- Is the SSL port configured in the alias list in WebSphere Application Server? To configure the alias list:
  1. From the WebSphere Administrative Console, select default_host.
  2. Click the Advanced tab.
  3. In the Aliases list, duplicate each entry, adding a suffix of **:ssl_*port#***, where *port#* is the SSL port number required by your HTTP server.

WebSphere pagecompile

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

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

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

```
d:\hostpub\WebSphere\AppServer\temp\default_host\hostpublisher\
pagecompile\_HPAdmin\main_xjsp.*
```

server.properties file

If clicking an action button (for example, Submit, Stop Server, and so forth) results in an error message, check the server.properties file in the *install_dir*\server directory to ensure that the value for the servletURL parameter for WebSphere 3.5 is: /_IBM_HP_WebAdmin_/HPAdminServlet. See "Appendix C. Server properties files" on page 161 for details about the server.properties file.

Fully-qualified domain names

If you cannot bring up Host Publisher Server Administration using the fully-qualified domain name, try using just the host name without the domain name. If this works, then you can solve the problem with the following steps:

1. Bring up WebSphere Administrator's Console.
2. Click WebSphere Administrative Domain.
3. Click default_host.
4. Click Advanced.
5. Add the host name with its fully-qualified domain name in the Aliases table.
6. Click Apply.
7. Restart WebSphere.

Common problems and limitations

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

Improving performance for TN3270E sessions

If you experience poor performance with Host Publisher applications running in a TN3270E environment, you might be able to alleviate the problem by setting the TCPNoDelay property to true in the Host On-Demand session properties for your application. The Host On-Demand session properties are defined in the .connspec file for your application. Add the following line to your session properties in the .connspec file, as illustrated in the following example:

```
......
<sessionprops>
   TCPNoDelay=true
   host=ralvm17
   autoReconnect=false
   SSL=false
   TNEnhanced=false
   port=23
   ......
</sessionprops>
......
```

When executing macros on TN3270E connections, Host Publisher can send multiple small requests to the TN server as part of the TN3270E protocol, only the last of which results in a response (screen update) from the host application. Normally, the Java networking code waits for a small interval between the sending of subsequent requests to conditionally receive a response from the connection partner, and thus avoids sending many small packets in the network. For host applications, such responses never arrive, so the wait causes an unnecessary delay that adversely affects performance.

Setting the TCPNoDelay property to true should improve performance because the Java networking code does not introduce this unnecessary delay.

With Host Access and execution of Host Access Integration Objects

Host screen truncated on left at default size

On Korean and Traditional Chinese Windows platforms, some characters may not display in the host session screen, or the wrong characters display and the screen appears corrupted.

This happens because the font used to display characters is not selected correctly, probably because the font properties file is incorrect.

You may solve this problem by installing the latest version of Netscape 4.06; if not, workarounds are available as follows:

Korean: In the \HostPub\Common\JDK\lib directory:

1. Back up the font.properties.ko file.
2. Change the definitions for the Monospaced from:

   ```
   monospaced.0=Gulim,HANGEUL_CHARSET
   monospaced.1=Courier New,ANSI_CHARSET
   monospaced.2=WingDings,SYMBOL_CHARSET,NEED_CONVERTED
   monospaced.3=Symbol,SYMBOL_CHARSET,NEED_CONVERTED
   ```

   to:

   ```
   monospaced.0=\uad74\ub9bc\uccb4,HANGEUL_CHARSET
   monospaced.1=Courier New,ANSI_CHARSET
   monospaced.2=WingDings,SYMBOL_CHARSET,NEED_CONVERTED
   monospaced.3=Symbol,SYMBOL_CHARSET,NEED_CONVERTED
   ```

Traditional Chinese: In the \HostPub\Common\JDK\lib directory:

1. Back up the font.properties.zh_TW file.
2. Modify the definitions for the Monospaced as follows:
   a. Change:

      ```
      monospaced.0=\u7d30\u660e\u9ad4,CHINESEBIG5_CHARSET,NEED_CONVERTED
      monospaced.1=Courier New,ANSI_CHARSET
      ```

      to:

      ```
      monospaced.0=Courier New,ANSI_CHARSET
      monospaced.1=\u7d30\u660e\u9ad4,CHINESEBIG5_CHARSET,NEED_CONVERTED
      ```
   b. Change:

      ```
      fontcharset.monospaced.0=sun.io.CharToByteBig5
      ```

to:

```
fontcharset.monospaced.1=sun.io.CharToByteBig5
```

c. Change:

```
exclusion.monospaced.1=0100-f8ff
```

to:

```
exclusion.monospaced.0=0100-f8ff
```

Macros fail in Host Access Integration Objects

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

See “Defining a screen” on page 18 for some tips for defining a screen in a macro.

Failures creating Integration Objects

When creating an Integration Object, Host Publisher builds Java source code and compiles it into an executable file. During this process, 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.

To solve this problem:

- You might 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, 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.

Macro Play Error in Host Access Integration Objects

If you experience a Macro Play Error, one possible cause is as follows.

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

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

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

With Database Access and execution of database Integration Objects

Database Access does not start

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

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

Failures creating Integration Objects

When creating an Integration Object, Host Publisher builds Java source code and compiles it into an executable file. During this process, 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.

To solve this problem:

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

Microsoft Access date fields

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

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

Database interface does not work with Lotus Domino JDBC driver

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

Java errors with Oracle database driver

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

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

Receive error: No suitable driver found

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

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

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

Connect timeout in a database Integration Object has no effect

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

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

Connecting to an iSeries database using Windows 98

If you are running Host Publisher Database Access on Windows 98, you might be unable to connect to an iSeries database.

To solve this problem:

- Edit DBAccess.bat in *Install_dir*\Studio and remove the double quotes at the beginning and end of the classpath value for the SET CLASSPATH statement.
- Save the file.

If you are using Host Publisher Studio to create a database Integration Object by launching Host Publisher Database Access from within Host Publisher Studio, you might be unable to connect to an iSeries database.

To solve this problem:

- Edit webbridge.bat in *Install_dir*\Studio and remove the double quotes at the beginning and end of the classpath value for the SET CLASSPATH statement.
- Save the file.

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

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

- contain the character(s)

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

your end user **must** enter the wildcard character (%) to indicate which operator is used when the Integration Object is run on the Host Publisher Server. After the Integration Object is created, the operators can be used interchangeably by entering the wildcard character in varying positions of the value, as shown in the following table:

Table 1.

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

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

With the Studio and transferring applications

Installation files and problems publishing applications

Ownership of installation files on UNIX operating systems: WebSphere Application Server on Solaris or AIX runs with the user ID and group of *nobody*, and Host Publisher runs with the same permissions. Host Publisher Server 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.

To change the ownership of all the files shipped with Host Publisher from *nobody* to your chosen ID, type the following at a command line after installation:

1. cd /usr/lpp
2. find HostPublisher -exec chown *new_user_ID* {} ″;″
3. find HostPublisher -exec chgrp *new_user_group* {} ″;″

When you specify a Sun Solaris or AIX FTP server (Preferences > Options) in Host Publisher Studio, or when 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.

With the Server and execution of Integration Objects

Connection for Chained Integration Objects in a cloned JVM

When there are cloned JVMs running applications containing chained Integration Objects, the HttpSession invalidation timer thread runs in the JVM where the HttpSession object was created. A chained application might not be running in that JVM; therefore, on HttpSession timeout, the connection associated with a chained application might not get cleaned up if the HttpSession invalidation occurs on a different JVM.

The easiest solution is to modify the maxBusyTime in the .poolspec file for the connection, or use Host Access to set Maximum busy time before disconnection on the Connection Pool Configuration page.

Error page does not display for error in second Integration Object on page

If a JSP contains multiple Integration Objects and an error occurs in the instantiation or execution of an Integration Object after output is written to the HTTP output stream, the error page does not display correctly. If substantial output is written to the HTTP output stream, then redirection of the browser to the error page might not work.

To avoid this problem, ensure that an error is detected before any output to the page can take place. In the example below, make sure all Integration Objects are created and execution is completed before any HTML is written to the output stream. Also, explicitly check for errors and stop processing if any doHPTransaction() fails. Otherwise, HPS5035 errors can occur when subsequent Integration Objects try to use the back-end connection.

```
<jsp:useBean id="Oi7" class="IntegrationObject.Oi7" scope="request"></jsp:useBean>
<jsp:setProperty name="Oi7 property="*" />

<jsp:useBean id="Oi8" class="IntegrationObject.Oi8" scope="request"></jsp:useBean>
<jsp:setProperty name=Oi8" property="*" />

<>%
   Oi7.setHPubErrorPage("Error.jsp");
   Oi8.setHPubErrorPage("Error.jsp");

//insert your code

   Oi7.doHPTransaction(request, response);
   //While an error should have caused a sendRedirect() to the
   //error page to notify the browser, we should also ensure that
   //this servlet must not proceed.
   if(Oi7.getHPubErrorOccurred() != 0)
   {
      return;
   }
   Oi8.doHPTransaction(request,response);
   if (O8i.getHPubErrorOccurred() != 0L
   {
      return;
   }

//insert your code

   //Now that we have made the doHPTransaction(...) method calls (that contain
   //sendRedirect(...)s in cases of error, we can now begin to put in HTML
   //or Java Script code or <INSERT> tags.
%>
<HTML>
<BODY>

<!-- [[ insert your code ]] -->

</BODY>
</HTML>
<% //This should appear AFTER we have written all our tags and output. %>
```

Multiple accesses from the same machine to chained Integration Objects cause problems

Netscape versions prior to 4.7 use a single HttpSession for Web access, even when accessing from different browser windows. The multiple accesses try to use the same data source connection resource, resulting in confusion for the chained IOs.

To avoid this problem, upgrade to Netscape version 4.7 or higher. Each browser window will then get its own HttpSession.

When you double-click the Internet Explorer icon to open the Internet Explorer browser, this problem does not occur; however, if you use any method other than double-clicking the icon to open the browser, the problem will occur.

Servlet generated by page compilation reports exception: Wrong name

This error occurs when aliases in httpd.conf are created using the same characters but different case; for example, HostPublisher and hostpublisher. Most Web servers do not consider the URL to be case sensitive, so it is possible to specify /HostPublisher/ or /hostpublisher/ in a URL and have it go to the same resource, even if the Web server is configured only with the alias /HostPublisher/. The problem is that the PageCompile depends on a case-sensitive path to the JavaServer Page (JSP). Once a URL is used to access a JSP, you must use the same capitalization within the URL to access that page in the future.

The same problem occurs if you use a different case the second time you reference a file; for example, Tax_Init_Page.jsp and then Tax_init_page.jsp. You must request the page by the *exact* name that was first used, you must delete the information in the pagecompile.

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

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

```
c:\HostPublisher\WebSphere\AppServer\servlets\pagecompile\
_HostPublisher\_HPAdmin\main_xjsp.*
```

Unable to access Host Publisher directories on iSeries

If you are attempting to access Host Publisher directories from a browser and you receive a message saying that you are not authorized, add the **DirAccess On** statement to the http configuration.

Generic Web browser timeout message is received instead of an expected Host Publisher error message

To avoid this problem, configure the Web server and/or Web browser timeouts (typically 2 minutes) to be greater than the Host On-demand Macro Timeout (default: 60 seconds).

To configure the Host On-demand Macro Timeout, edit the macro. In the first line of the HASCRIPT tag, set the timeout field to the desired value (1000 equals 1 second).

When you specify Connection Pooling values in Host Access and Database Access, if you set the Maximum Busy Time Before Disconnection to something other than –1 (never), the sum of this timeout and the Host On-demand Macro timeout should be less than the Web server and Web browser timeouts.

Note: In some cases, the Maximum Busy Time Before Disconnection expires and causes the macro to stop; however, a macro timeout error message is generated instead of the busy timeout error message.

Recommended Java heap sizes for Solaris Java Virtual Machine (JVM)

WebSphere Application Server recommends that you set the initial Java heap size to between 10% and 25% of the maximum heap size. These values are set in WebSphere Application Server's Administrative Console under properties.mx (maximum heap size) and java.ms (initial heap size).

WebSphere Application Server does ***not*** recommend making these values equal. The ability of the garbage collection algorithm to track allocated memory could result in performance problems if these values are too similar.

Making WebSphere Application Server handle 100 or more requests

Host Publisher is configured to handle a maximum of 50 concurrent connections. If you want more than 100 concurrent connections, you must apply WebSphere service and modify several Host Publisher Server parameters.

1. Increase Max Heap Size for Host Publisher Server
   a. From the WebSphere Administrative Console, select YourHostName > HostPubServer in the topology tree.
   b. To create more than 200 concurrent sessions, on the General tab, in the Command line arguments field, increase the maximum heap size to 256m (the default is 128m) as follows:
   ```
   Command line arguments:  -mx256m -ms32m
   ```
   c. Click Apply to save the change.
2. Increase Max Connections for Host Publisher Servlet Engine
   a. From the WebSphere Administrative Console, select YourHostName > HostPubServer > HPServletEngine in the topology tree.
   b. On the Advanced tab, change Max Connections to reflect the new maximum number of concurrent requests for Host Publisher (the default is 50).
   c. Click Apply to save the change.
3. Update the Web Server Configuration

   Most Web servers have a configurable limit for the number of concurrent clients they will support. You might have to modify the Web server configuration to make this limit greater than the Max Connections parameter for the servlet engine.

   After making changes to WebSphere and the Web server, stop and restart both servers to pick up the new changes.

PluginTester servlet for debugging WebSphere Application Server problems

Host Publisher version 2.2 and above includes the servlet showCfg, which is the equivalent of the PluginTester. You can access this servlet at this URL: http://*hostname*/*HostPublisher*/showCfg, where *hostname* is your local server and *HostPublisher* is the Host Publisher alias specified during installation; for example, if you chose HP as your alias, you would use http://*hostname*/HP/showCfg to access the servlet. The default alias is HostPublisher.

Pages are not returned

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

To solve this problem, increase the Connecttimeout parameter in the .connspec file, and the timeout attribute of the <HAScript> tag in all the macros.

Note: The .connspec files are located in the \Server\production\poolspecs directory on the Server.

Shutting down Host Publisher Server

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.

To avoid this problem, stop Host Publisher Server before you stop WebSphere Application Server.

Out of memory error starting 20 sessions

Host Publisher creates one thread per pool, and up to 50 threads during session recovery and shutdown. Host Publisher uses Host On-Demand, which creates a maximum of 100 threads regardless of how many sessions are created.

To avoid this problem, 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 Application Server.

Configuring WebSphere Application Server and your Web server for Host Publisher Server

The following instructions are the WebSphere Application Server and Web server configuration steps required by Host Publisher. If Host Publisher Server installation fails or creates configuration problems, follow these instructions as a solution. If Host Publisher Server installation is successful, these steps are performed automatically during installation, requiring no additional configuration by you.

This information refers to Host Publisher's installation directory, which differs by installation and platform as follows:

- **For Windows NT**, <install_dir> and <install_dir2> is machine-specific. The default for both is C:\HostPub.

- **For AIX:**
  - <Install_dir> = /usr/lpp/HostPublisher
  - <Install_dir2> = /var/HostPublisher
- **For Solaris**
  - <Install_dir> = /opt/HostPublisher
  - <Install_dir2> = /var/HostPublisher
- **For OS/400**
  - <Install_dir> = /QIBM/ProdData/HostPublisher
  - <Install_dir2> = /QIBM/UserData/HostPublisher

Web Server Configuration

Modify the Web server's configuration file to contain the following Host Publisher aliases:

Running on WebSphere Application Server 3.5:

```
Alias  /HostPublisher  /install_dir/Server/production/documents/
Alias /_IBM_HP_WebAdmin_  /install_dir/Server/production/documents
Alias /_IBM_HP_doc_  /install_dir/Common/doc/
```

Note: /HostPublisher/ is the default value. If you chose to replace /HostPublisher/ with something different during installation, use that value here.

The Web server's configuration file and location differs based on Web server software and installation choices. For example, the default location for the configuration file for IBM HTTP Server on Windows NT is c:\Program Files\IBM HTTP Server\conf\httpd.conf.

Notes:

1. Depending on the configuration you are modifying, you may have to use a Pass statement instead of an Alias statement. Refer to other alias examples within the configuration to understand which statement to use.
2. Do not change the aliases _IBM_HP_WebAdmin_ and _IBM_HP_doc_.

Gathering WebSphere Application Server and Web server problem data

WebSphere Application Server 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 Application Server while trying to build or run a servlet

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

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

Standard output and standard error output logs

WebSphere Application Server can create one or more JVMs to run Java servlets, Host Publisher Server, and Integration Objects. WebSphere Application Server creates two files that contain the standard output and standard error output from each Java virtual machine.

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

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

OS/400
/QIBM/UserData/HostPublisher/log

Sun Solaris
/usr/WebSphere/AppServer/logs

Windows NT
By default, these files are named stdout.txt and stderr.txt and are located at *Install_dir*\Log. The names and locations are determined by the settings for a given JVM. Use the WebSphere Administrative Console to review these files.

Error logs redirected from Web servers

WebSphere Application Server is configured to use Web servers. WebSphere Application Server redirects error logs from Web servers into the WebSphere Application Server 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 Application Server 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:

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

OS/400
/QIBM/UserData/WebASAdv/default/logs

Sun Solaris
/usr/WebSphere/AppServer/logs

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

Error logs for problems and events discovered by WebSphere Application Server

WebSphere Application Server 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 you are executing the Studio:
   - Error output for Host Access and Host Publisher Studio are automatically routed to the .con files, HostAccess.con and Studio.con, respectively. Check these files for error information.
   - To route error output for Database Access to a console window:
     a. Execute from a command line.
     b. Modify the invocation in the .bat file to use java instead of javaw.
   - Refer to "Enabling tracing" on page 45 for information about enabling tracing in the Host Publisher Studio.
   - Additional error output may be located in the messages_HostPubStudio_0822_x.txt file located in the *install_dir*\studio directory.

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. When you report a problem, use the product.xml file to provide information such as product version, current APAR levels, and so forth. This file is located in *install_dir*\Server and *install_dir*\Studio.

If you determine that you need to contact IBM, you can do any of the following:

- Consult the **Customer Service and Support Guide**, which is a card contained in the product package.
- Access the Host Publisher Web page at:

  `http://www.ibm.com/software/webservers/hostpublisher`
- 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. Technical overview

Execution models for Integration Objects

Host Publisher licensing requirements restrict the execution of an Integration Object to a WebSphere-created JVM. Given this restriction, there are various execution models possible for invoking Integration Objects.

1. An Integration Object executed in a Web context (for example, a Web container) can be used to build dynamic HTML (or other markup language) content using Java Server Page (JSP) technology. Alternatively, a servlet that drives one or more Integration Objects can also be developed for dynamic HTML generation.
2. An Integration Object can be executed in an EJB container to exploit the benefits of EJB technology such as resource location, replication and load balancing, and to facilitate its use in composite EJB applications. Host Publisher Studio provides special support for building EJB support for Integration Object execution.
3. To facilitate the execution of an Integration Object from a non-WebSphere JVM, Remote Integration Object (RIO) technology is provided. The non-WebSphere JVM, such as a browser JVM running an applet, invokes RIO proxy code that communicates with the real Integration Object in a WebSphere JVM running on a separate machine and/or process.

The following section elaborates on alternative 1 using Java Server Pages.

An Integration Object running in a WebSphere Container

The figure below illustrates the Host Publisher development and execution environments for Web-based Host Publisher applications.

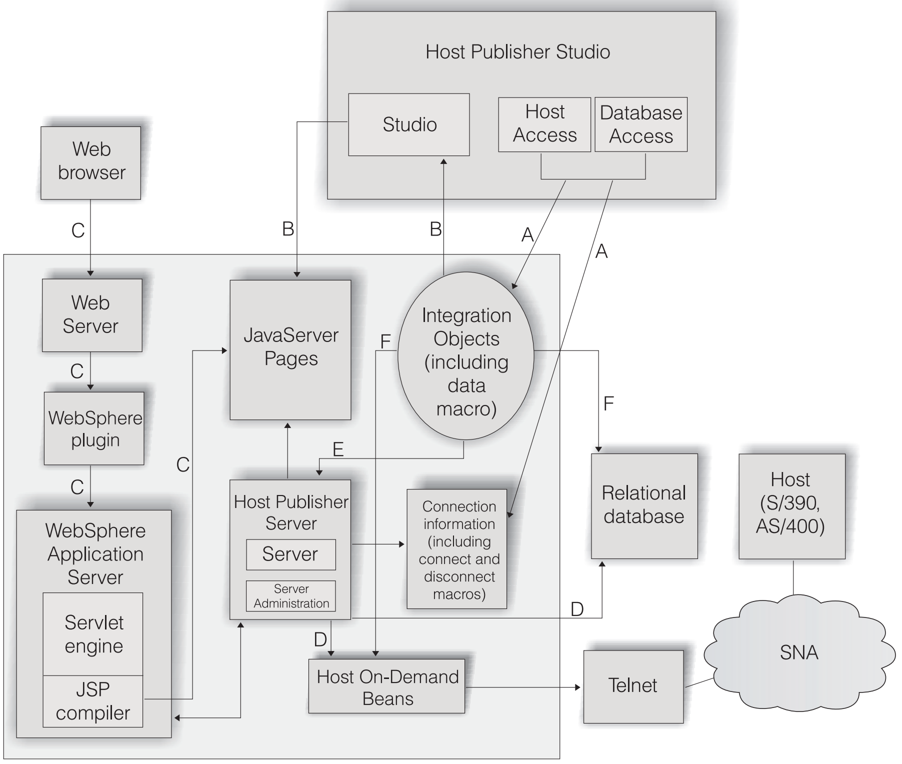


Figure 6. Host Publisher development and execution environments

The Host Publisher Studio applications, Host Access and Database Access, create Integration Objects and connection information representing access to either the host terminal-oriented application or to the relational database (see A in Figure 6.) In addition, Host Access creates Host On-Demand macros for executing the connect, data, and disconnect macros for a terminal-oriented application. Host Publisher Studio uses the Integration Object to build JSPs that create the Integration Object, set its parameters, execute it, and retrieve its output (B). These objects and files produced by the Studio are published to and used by the Server at execution time.

When these objects and files are published to the server, they are ready to be executed. The execution environment for an Integration Object is provided by:

- The Java support provided by a JVM created by the WebSphere Application Server.
- The Host Publisher Server, which provides support for connection pooling, connection setup and priming, user ID management, administration, tracing/logging, and so forth.
- Host On-Demand or JDBC code for access to various data sources.

Note: Host Publisher uses the components of the Host On-Demand product, a Java applet-based terminal emulator, to communicate with terminal-oriented applications. Host On-Demand consists of a set of Java-based telnet (TN) clients that can communicate with:

1. TN3270(E) servers for the 3270 data stream to communicate with z/OS-based applications
2. TN5250(E) servers for the 5250 data stream to communicate with iSeries-based applications
3. Telnet servers for the VT datastream to communicate with applications that run on various Unix, OS/2, and other platforms.

Host On-Demand also provides programmatic access to these TN client functions to create connections to terminal-oriented applications and navigate through them. Host Publisher uses the programmatic access to Host On-Demand functions.

Execution of a Studio-generated application starts with a browser accessing a JSP. WebSphere Application Server contains support for JSP compilation and a servlet engine that supports the servlet API. The JSP is compiled into a servlet, and the servlet executed (C).

The resulting servlet first creates an Integration Object, then sets its parameters, then executes it. Execution of an Integration Object begins with it asking the server for a host connection (E). The server creates the connection either to a TN server for a terminal-oriented application or to a relational database (D). For terminal-oriented applications, the server either establishes a new connection and runs the connect macro using the Host On-Demand API, or it returns an existing connection from a pool. For database applications, the server either establishes a connection with the database or it returns an existing connection from a pool.

Once the Integration Object acquires the connection, it either runs a data macro (in the case of terminal-oriented application) or executes a database statement (in the case of a database application), and retrieves the data from the data source (F). The Integration Object’s output properties are now set and ready for retrieval. The rest of the JSP execution uses those output properties for dynamic HTML generation. Integration Object execution ends with it returning the connection to the server, where the server decides if it is returned to a pool or disconnected for its source, and its output properties set.

Appendix B. Server error messages and recovery actions

If you need to contact IBM, see the information in “Contacting IBM for service” on page 111.

(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
Try again. 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 Application Server 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 Application Server 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 Application Server 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 Application Server 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 Application Server. Install Host Publisher again, or correct the servlet property using the WebSphere Application Server 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 Application Server 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 Application Server 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

Try again. 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 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 connection 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 connection pool.

(HPS5031) Object {0} not defined.

Explanation
The named object is not defined. A Host Publisher Integration Object refers to a connection pool that is not defined on the server.

User Action
Try deploying the Host Publisher application again. This will also deploy the required connection pool definition.

(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 Integration Object tried to acquire an existing connection from the Web session, but there is no web connection. Either the Integration Object that requested the connection or the Integration Object that saved the object are miscoded, or the web connection has timed out, and the web connection (HttpSession) has been deleted.

User Action

Examine the Integration Objects for errors, and determine whether or not the timeout value for the web connection (HttpSession) is too short.

(HPS5035) There is no data source object {0} in HttpSession. A possible cause is the use of multiple browsers from a single machine to a chained application. See documentation for more information.

Explanation

A chained Integration Object attempted to retrieve a data source resource from the HttpSession, but the resource is not in the HttpSession object. This may occur for a number of reasons:

1. Netscape browsers use a single HttpSession for Web access, even when accessing from different browser windows. The multiple windows request the same data source resource, when the multiple windows should be using different resources.
2. When you double-click the Internet Explorer icon to open the Internet Explorer browser, this problem does not occur; however, if you use any method other than double-clicking the icon to open the Internet Explorer browser, the problem will occur because the Internet Explorer browser windows will then share a single HttpSession.

3. A single JSP or servlet runs multiple Integration Objects without checking for an error in a previous Integration Object and stopping the next Integration Object from executing. The previous Integration Object will not have placed the data source resource back in the HttpSession if it encountered an error.
4. A user clicks Submit or clicks a link twice before the first button press or link click can be completely processed at the server. If processing is taking longer than expected, a user might click on a link to the next page of a chained Integration Object application twice rather than wait for the next page to process its Integration Object and send the output page back to the client. The second click causes the next page to be processed again, while the Integration Object still has ownership of the data source resource. During this processing, a second instance of the Integration Object cannot find the resource in the HttpSession, and causes a HPS5035 error to be logged and the output of the first Integration Object processing is lost.
5. A very long-running Integration Object (usually more than a minute) might cause the Web server to time out while Host Publisher and the Integration Object are still processing. If this occurs, the Web server may destroy the HTTP connection to the browser client. Internet Explorer sometimes reacts to this by immediately requesting the same page again without user intervention. This second request is just like a second Submit or link click, causing a second instance of the Integration Object to run. This second instance fails to find the data source resource in the HttpSession, which causes a HPS5035 error. Additionally, the output of the first Integration Object processing is lost.
6. If the maximum busy time before disconnection specified for the connection expires because a chained Integration Object application was left idle between Integration Objects executing (for example, the Integration Objects are on different pages), then the next Integration Object in the chain will log an HPS5035 if a user continues with the application after the expiration time.
7. If the HttpSession Activity Timeout, set in WebSphere Application Server, occurs because a chained Integration Object application is left idle between Integration Objects executing (for example, the Integration Objects are on different pages), then the next Integration Object in the chain logs a HPS5035 if a user continues with the application after the expiration time.

User Action

The correct action depends on the cause. Possible actions for the listed causes are, respectively:

1. To avoid this problem, do not attempt to run multiple applications containing chained Integration Objects concurrently from the same client machine running Netscape.
2. If you are running multiple applications containing chained Integration Objects, be sure to double-click the Internet Explorer icon to open the Internet Explorer browser.
3. JSPs or servlets should not attempt to continue running subsequent chained Integration Objects after an error has occurred in a previous Integration Object in the chain. Instead, subsequent Integration Objects should not be executed if an error occurred in a previous Integration Object, and the Integration Object in error should redirect the client to an error page.

   You might avoid this problem by using servlet or JSP code like this:

   ```
   myBean.doHPTransaction(request, response);
   if (myBean.getHPubErrorOccurred() != 0)
   {  // Do not call another Integration Object's doHPTransaction()
    or produce output.
      return;
    // Instead, we allow the sendRedirect() to answer the client.
   }
   ```
4. You might avoid this problem by using a JavaScript snippet like this one:

   ```
   <SCRIPT Language="Javascript">
   /*********************************************************************
   *  Prevent multiple Submit or href click invocations from this page.
   *  The 'submitFlag' variable is the switch.  When the function is
   *  called and the flag is zero, 'true' is returned to the onSubmit
   *  parameter of the form. Otherwise a 'false' is returned, preventing
   *  additional submit buttons/links from being reselected.
   **********************************************************************
   var submitFlag = 0;
   function chkSubForm()
   {
      if (submitFlag == 0)
      { submitFlag = 1;   // first time a button has been clicked
        return true;
      }
      else
      { return false;    // submit has already been clicked
      }
   }
   ```

```
</SCRIPT>
<a href="<%= response.encodeUrl("TheNextPage.jsp") %>"
  onClick="return chkSubForm()">Next</a>
```

5. The simplest way to prevent this is to have no Integration Object run for a very long time. Alternatively, you may be able to adjust the Web server time-out value.
6. If this is not what you want, you can lengthen or disable the maximum busy time before disconnection on your connection.
7. If this is still not what you want, you can lengthen or disable the HttpSession Activity Timeout.

(HPS5036) Attempt to create user list 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 list with invalid name: {0}.

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

User Action
Check the .userpool file to ensure that the localuserpool name is set to a value. If it is not, you can edit the file. A valid user list name must be unique and cannot contain:

- Front or back slash (/ \)
- Colon (:)
- Asterisk (*)
- Question mark (?)
- Quotation mark (″)
- Less than or greater than brackets (< >)
- System device names: con, com, lpt, prn, or nul

Caution: If you have limited experience editing macro files, please call IBM service.

(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 list 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 list {0}.

Explanation
The named user list has no free users available.

User Action
Retry the request later, or enlarge the size of the user list.

(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 Application Server or reloadable servlet classpath. Check both. Use the WebSphere Application Server PluginTester servlet, as described in "Contacting IBM for service" on page 111, 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 list, 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 list 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 list. 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 list. 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 list 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 list {0} is encrypted.

Explanation
The named user list 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 list {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 list.

User Action

Examine the user list being used by this application. If you do see entries that define values for the property _password, contact IBM service. Otherwise, the user list 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 list data failed. The supplied encryption key is incorrect or the user list data has been corrupted.

User Action

Examine the user list's XML file, and ensure you are supplying the correct encryption key. You may have to republish the user list 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 list.

(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 list {1}.

Explanation

The named user was not found in the named user list.

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.

(HPS5102) Insufficient rights to the file {0}. Please check file owner and permissions.

Explanation

The permissions of the files are not set to values which allow the requested action. This message could result from a failed attempt to deploy an application. If the files being deployed cannot be cleaned up from the staging area, then the application is not deployed and this message can result.

User Action

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

(HPS5103) The data source connection has been removed, probably because the maxBusyTime has expired.

Explanation

The connection to the host application or database has already been removed. It is possible that maxBusyTime has been configured in the .poolspec file for this application and time has expired, causing the Host Publisher Server to terminate the connection before the Integration Object returned the connection to the Server. This can occur when a chained Host Publisher application is running, and the end user pauses between two middle-in-chain Integration Objects for a time longer than the maxBusyTime configured for the connection pool.

User Action

Ensure that maxBusyTime is set to an appropriate value. In Host Access, click the Connection Pools tab in the left pane. In the right pane, click the Connection Pool Configuration tab, then in the Connection Timeouts box, type a new value for Maximum busy time before disconnection.

(HPS5104) Deploy application ″{0}″.

Explanation

This is an informational message. The deployment of the specified application has been attempted. If there is a problem with deployment, other error messages will follow.

User Action

None.

(HPS5105) Remove application ″{0}″.

Explanation

This is an informational message. The removal of the specified application has been attempted.

User Action

None.

(HPS5106) No host name provided to acquireHostConnection call: {0}

Explanation

The name of a host was not present in the Properties object which was supplied to the acquireHostConnection call.

User Action

Specify the host name in the Properties object. The host name is required.

(HPS5107) User list {0} not found.

Explanation

An unexpected error occurred.

User Action

If the problem persists, clear all Host Publisher trace and log files, turn on all Host Publisher Server tracing and tracing for the Integration Object(s) associated with this application, and try to recreate the condition. Contact IBM service and provide the trace and log file information.

(HPS5108) Connection Pool {0} not found.

Explanation

An unexpected error occurred.

User Action

If the problem persists, clear all Host Publisher trace and log files, turn on all Host Publisher Server tracing and tracing for the Integration Object(s) associated with this application, and try to recreate the condition. Contact IBM service and provide the trace and log file information.

(HPS5109) Hostname {0} cannot be resolved.

Explanation

The hostname provided cannot be resolved in the network.

User Action

Verify that the hostname provided exists in the network and is reachable.

(HPS5110) RemoteException received when making RMI call:\n {0}

Explanation

When attempting the specified call, a RemoteException was received. This usually indicates a network outage.

User Action

Verify that the server being administered is still available.

(HPS5111) Integration object {0} has no methods. Check your Java CLASSPATH.

Explanation

While attempting to access the specified Integration object, no methods could be found. This most commonly happens if the Integration Object class is not accessible to WebSphere Application Server.

User Action

Verify that WebSphere Application Server's CLASSPATH correctly reflects accessibility to the Integration Object.

(HPS5112) Error adding name to RMI registry. Exception:\n {0}

Explanation

An error occurred while attempting to add a name to the RMI registry. This is usually caused by a network connectivity problem.

User Action

Verify network connectivity. Examine the Exception specified for more details about this problem.

(HPS5113) Error creating RMI registry. Exception:\n {0}

Explanation

The RMI registry could not be created.

User Action

Examine the Exception specified and other messages in the message log for more details about this problem.

(HPS5114) Error unbinding name from RMI registry during JVM shutdown. Exception:\n {0}

Explanation

An error occurred while attempting to remove an entry from the RMI registry. This usually occurs because the JVM is going down and the status of items like Sockets is suspect.

User Action
Examine the Exception specified for more details about this problem.

(HPS5115) Protocol error - first object from nondistinguished Server not JVM name:\n {0}

Explanation
A protocol error occurred during inter-JVM communication. This is an internal error.

User Action
If the problem persists, clear all Host Publisher trace and log files, turn on all Host Publisher Server tracing and tracing for the Integration Object(s) associated with this application, and try to recreate the condition. Contact IBM service and provide the trace and log file information.

(HPS5116) Protocol error - JVM name null or zero length.

Explanation
A protocol error occurred during inter-JVM communication. This is an internal error.

User Action
If the problem persists, clear all Host Publisher trace and log files, turn on all Host Publisher Server tracing and tracing for the Integration Object(s) associated with this application, and try to recreate the condition. Contact IBM service and provide the trace and log file information.

(HPS5117) Key {0} not present in message file.

Explanation
The information for the specified message could not be found.

User Action
If the problem persists, clear all Host Publisher trace and log files, turn on all Host Publisher Server tracing and tracing for the Integration Object(s) associated with this application, and try to recreate the condition. Contact IBM service and provide the trace and log file information.

(HPS5119) User id and passticket could not be acquired from the DCAS server because no host application id was found.

Explanation
A request could not be sent to the DCAS server to retrieve the user ID and passticket because the application ID was not present. The application ID should have been reported

through a custom event, via a <custom> tag in the macro that was recorded by the Host Access application.

User Action
Verify that the <custom> tag is specified correctly in the macro.

(HPS5120) User id and passticket could not be acquired from the DCAS server because the processing was not occurring in the context of a Web request.

Explanation
An express logon-enabled Integration Object is being run outside a Web container in WebSphere Application Server.

User Action
Express logon-enabled Integration Objects can execute only in a Web container. If you believe that you are running such Integration Objects from servlets or JSPs, contact IBM service.

(HPS5121) No password specified to unlock the key ring class file, containing the client certificate to be used for express logon.

Explanation
An unexpected error occurred.

User Action
If the problem persists, clear all Host Publisher trace and log files, turn on all Host Publisher Server tracing and tracing for the Integration Object(s) associated with this application, and try to recreate the condition. Contact IBM service and provide the trace and log file information.

(HPS5122) A timeout occurred before DCAS client-server protocols could be completed to acquire a user id and passticket.

Explanation
The timeout expired before the DCAS client-server protocol could complete.

User Action
Retry the operation. If the problem still exists, increase the connection attempt timeout. In Host Access, click the Connection Pools tab in the left pane. In the right pane click the Connection Configuration tab, then in the Connection Information box, type a new value for Connection attempt timeout. (You can also make this change by increasing the Connecttimeout value in the .connspec file.) Check your DCAS Server status. If the problem persists, clear all Host Publisher trace and log files, turn on all Host Publisher Server tracing and tracing for the Integration Object(s) associated

with this application, and try to recreate the condition. Contact IBM service and provide the trace and log file information.

(HPS5123) Internal error - unexpected return code from DCAS client.

Explanation
An unexpected error occurred.

User Action
If the problem persists, clear all Host Publisher trace and log files, turn on all Host Publisher Server tracing and tracing for the Integration Object(s) associated with this application, and try to recreate the condition. Contact IBM service and provide the trace and log file information.

(HPS5124) DCAS server {1} cannot be located.

Explanation
The specified DCAS server could not be located in the network.

User Action
Verify that the DCAS server is reachable.

(HPS5125) User id and passticket could not be acquired from the DCAS server because there was no Web browser certificate associated with this request.

Explanation
While attempting to send a request to the DCAS server, the Web browser certificate could not be acquired.

User Action
The Web browser and/or the Web server being used to access the Host Publisher application has not been set up for HTTPS using client authentication.

(HPS5126) The key ring database class file could not be found. Check that it is in the "common" directory.

Explanation
While attempting to access the keyring database class file during express logon processing, the class file could not be found.

User Action
Verify that the key ring database class file is in the \Common directory of the Host Publisher Server installation, or in a directory that is in WebSphere's CLASSPATH for the JVM running Host Publisher.

(HPS5127) The key ring database class file is invalid and could not be loaded. You might have to recreate it using iKeyman.

Explanation
: A problem was encountered while attempting to access the keyring database class file during express logon processing.

User Action
: Verify that the key ring database class file is valid. Try to recreate it using iKeyman.

(HPS5128) The password supplied does not match the password that was used to protect the key ring class file.

Explanation
: The supplied password was not valid for the key ring class file.

User Action
: Check the password and try again. Passwords are case sensitive.

(HPS5131) Some other (non-Host Publisher) TCP application running on this host is already using port {0}. Administration of the Host Publisher Server instance(s) on this host will not be possible.

Explanation
: While attempting to administer the Host Publisher Server instance(s) on the specified host, it was found that the Host Publisher Server Administration server component is not running on the specified host and port. Some other TCP program appears to be running on that host and port.

User Action
: Examine the message log for other messages that might give more details. Investigate the port conflict on the host between the Host Publisher Server and the other TCP program.

(HPS5132) No free TCP port could be found on the machine to create an RMI registry.

Explanation
: While attempting to create an RMI registry, no free TCP port could be found. Other non-Host Publisher TCP applications were found to be using all ports attempted.

User Action
: Examine the message log for other messages that might give more details. See if you can shut down some of the programs that listen on TCP ports.

(HPS5133) CreateRegistry method called more than once with TCP port number {0}. Registry information: {1}

Explanation

An unexpected error occurred.

User Action

If the problem persists, clear all Host Publisher trace and log files, turn on all Host Publisher Server tracing and tracing for the Integration Object(s) associated with this application, and try to recreate the condition. Contact IBM service and provide the trace and log file information.

(HPS5134) Received unexpected request code {0} during registry validation protocol.

Explanation

A protocol error occurred during inter-JVM communication. The specified request code was found to be invalid.

User Action

If the problem persists, clear all Host Publisher trace and log files, turn on all Host Publisher Server tracing and tracing for the Integration Object(s) associated with this application, and try to recreate the condition. Contact IBM service and provide the trace and log file information.

(HPS5135) Received unexpected response code {0} during registry validation protocol.

Explanation

A protocol error occurred during inter-JVM communication. The specified response code was found to be invalid.

User Action

If the problem persists, clear all Host Publisher trace and log files, turn on all Host Publisher Server tracing and tracing for the Integration Object(s) associated with this application, and try to recreate the condition. Contact IBM service and provide the trace and log file information.

(HPS5136) There is no Host Publisher server on host ″{0}″ named {1}.

Explanation

The Host Publisher Server with the specified name, which is being searched for on the specified host, does not exist.

User Action

Examine the message log for other messages that may give more details. Verify that the names of the host and the Host Publisher Server are correct. Verify that Host Publisher Server is running on the specified host, and that the port number you specified is the one defined in the file server.properties, as the value of the property called HPAdminPortNumber.

(HPS5137) Checkin screen did not match current PS. Connection = {0} ECL error = {1} Screen description = {2} Screen dump = {3}

Explanation
While verifying the state of the connection, it was found that the current screen of the 3270, 5250, or VT application does not match the screen it should be on, for the host connection to be put back into the connection pool.

User Action
Examine your macros to see if the screen that is in the log for this error message could have appeared after a normal execution of a data macro of an Integration Object (the last one in a chain, if chained). If so, your checkin screen description defined by the Host Publisher Host Access application is incorrect. This can occur if you ignore warning messages in Host Access while creating your Integration Object. The checkin screen defined in the Studio is the last screen of the connect macro if one exists, or the first screen of the data macro (of the first Integration Object in a chain, if chained).

(HPS5900) Server started by com.ibm.HostPublisher.EJB.HPubEJB

Explanation
This is an informational message.

User Action
None.

(HPS5901) Exception occurred attempting to start the Host Publisher Runtime Environment

Explanation
An exception occurred when the Host Publisher EJB attempted to start the Host Publisher Server.

User Action
Check the following related messages or refer to the message log for other messages that provide more details about this problem.

(HPS5902) Exception occurred while processing the EJB Properties input argument

Explanation

An exception occurred while the Host Publisher EJB was processing the input argument received from an EBJ Access bean.

User Action

Check the following related messages or refer to the message log for other messages that provide more details about this problem. Check both the EJB Access bean .jar file and Integration Object .jar file to verify that the Integration Object properties classes have the same size and date.

(HPS5903) Exception occurred while processing the EJB Helper object

Explanation

An exception occurred while the Host Publisher EJB was processing the Helper object name received from an EJB Access bean.

User Action

Check the following related messages or refer to the message log for other messages that provide more details about this problem. Check the Integration Object .jar file to verify that the Integration Object Helper class exists.

(HPS5904) Exception occurred invoking the EJB Helper object

Explanation

An exception occurred when the Host Publisher EJB invoked the Helper object to process the EJB Access bean request.

User Action

Check the following related messages or refer to the message log for other messages that provide more details about this problem.

(HPS5905) Exception occurred attempting to invoke ConnMgr to cleanup a connection

Explanation

An exception occurred when the Host Publisher EJB attempted to clean up the connection assigned to the processing of an Integration Object chain.

User Action

Check the following related messages or refer to the message log for other messages that provide more details about this problem.

(HPS5950) {0}: {1}: {2}: JNDI_NAME={3}, PROVIDER_URL={4}, INITIAL_CONTEXT_FACTORY={5}

Explanation
The EJB Access bean was unable to locate or connect to the Host Publisher EJB with the specified JNDI_NAME.

User Action
Check the following related messages for additional detail on the problem. Verify that the JNDI_NAME and PROVIDER_URL of the Host Publisher EJB are correct, and modify the Access bean .jar .properties with the correct values. Verify that the Host Publisher EJB is deployed and started on the target application server with the same JNDI name; correct if necessary.

(HPS5951) {0}: accessHandle==null || hPubLinkKey==null

Explanation
An Integration Object chain is being processed and the EJB Access bean is unable to retrieve the handle of the Host Publisher EJB or the key used to access the connection for the chain.

User Action
Check the following related messages for additional detail on the problem. If this EJB Access Bean is being processed by a servlet or JSP, see the Explanation and User Action for message HPS5035 on page 124 for possible causes. If this EJB Access bean is being processed from a Java program, verify that the program is setting the accessHandle and hPubLinkKey properties prior to invoking the middle or last Integration Objects in the chain. Refer to the *IBM WebSphere Host Publisher Programmer's Guide and Reference* for information on hPubLinkKey.

(HPS5952) {0}: getEJBObject(): {2}: {3}

Explanation
An exception occurred during an attempt to retrieve the EJB object for an Integration Object chain.

User Action
Retry the request. If this problem persists, turn on all WebSphere Application Server traces if the EJB Access bean is running in a WebSphere Application Server environment, and try to recreate the condition. Contact IBM service and provide the trace information.

(HPS5953) {0}: processIO({1}): {2}: {3}

Explanation
An exception occurred while processing the execution method of the Host PublisherEJB.

User Action
Check the following related messages for additional detail on the problem.

(HPS5954) {0}: HPubEJB.remove(), ejb={1}: {2}: {3}

Explanation
An exception occurred while remove an instance of the Host Publisher EJB.

User Action
Check the following related messages for additional detail on the problem. If this problem persists and the messages indicate a java.rmi.RemoteException occurred, turn on the WebSphere Application Server traces and Host Publisher traces, and try to recreate the condition. Contact IBM service and provide the trace information.

(HPS5955) {0}: ejb.getHandle(), ejb={1}: {2}: {3}

Explanation
An exception occurred during an attempt to get a handle for the HPublisher EJB object.

User Action
Retry the request. If this problem persists, turn on all WebSphere Application Server traces if the EJB Access bean is running in a WebSphere Application Server environment, and try to recreate the condition. Contact IBM service and provide the trace information.

(HPS5956) {0}: HttpServletRequest.getSession(true), hPubLinkKey={2}

Explanation
An Integration Object chain is being processed and an error occurred with the linkKey preventing further processing of the chain.

User Action
Retry the request. If this problem persists, turn on all WebSphere Application Server traces if the EJB Access bean is running in a WebSphere Application Server environment, and try to recreate the condition. Contact IBM service and provide the trace information.

(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 data 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

If you are running an Integration Object that requires numeric input fields, make sure you enter a value for each numeric input field. Numeric input fields cannot be blank.

(HPS6006) Introspection exception

Explanation

An Exception occurred while the Integration Object was performing internal introspection.

User Action

Contact application vendor if problem persists.

(HPS6007) Connection Link Key is null when trying to acquire a chained non-web connection

Explanation
A chain of Integration Objects is attempting to run, and the identification of this sequence has been lost.

User Action
Retry your request. If the request continues to fail, contact the application vendor.

(HPS6008) Connection Link Key is null when trying to release a chained non-web connection

Explanation
A sequence of Integration Objects is attempting to run, and the identification of this sequence has been lost.

User Action
Retry your request. If the request continues to fail, contact the application vendor.

(HPS6009) Unexpected exception: {exception text}

Explanation
An unexpected exception has occurred during execution of the Integration Object. The exception text is included in the message.

User Action
Correct the error identified in the exception text, then retry your request. If you do not know how to correct the error, contact the application vendor.

(HPS6010) Internal macro processing error, _userid improperly returned

Explanation
An error occurred while obtaining a user ID from Host Publisher express logon.

User Action
Retry your request. If the request continues to fail, contact the application vendor.

(HPS6011) Internal macro processing error, _password improperly returned

Explanation
An error occurred while obtaining a password from the Host Publisher express logon feature.

User Action
Retry your request. If the request continues to fail, contact the application vendor.

(HPS6100) Unable to acquire session from the Session Manager

Explanation

A connection to the data source, 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 screen, but the terminal screen coordinates of this data are not defined within this terminal screen.

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 data sources.

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, or 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.

(HPS6210) Numeric input values cannot be blank. Enter a value for all numeric input fields.

Explanation

At least one of the entry fields on the Web page is blank. All numeric fields must have a value.

User Action

Enter a value for all numeric entry fields on the Web page.

Appendix C. Server properties files

Server properties files are Java properties files that contain Host Publisher Server settings. There are two types of server properties files: one that contains global settings for a node, and one that contains settings for a particular Java virtual machine (JVM). The server properties files are written to the `Server` directory of the Host Publisher installation directory on the server. When you use Host Publisher Server Administration to make changes, the values you specify are written to the server properties files.

The server properties file for the global settings is server.properties. The file is created when Host Publisher Server is installed. The server.properties file is primed with default values when you start the Host Publisher Server for the first time.

The server properties file for a particular JVM is ras_*xxx*.properties, where *xxx* represents the JVM name (for example, HostPubServer_8120). There is one server properties file for each Host Publisher runtime instance running in a JVM. The ras_*xxx*.properties file contains the trace and log settings for that particular application server. If an application server starts the Host Publisher runtime instance within the JVM, and the same name is generated for that JVM, the trace and log settings are restored to the values that existed prior to a shutdown.

If no value is specified for any of the properties in the server properties files, Host Publisher uses the default value defined for that property.

In addition to using Host Publisher Server Administration to update this file, you can edit the server properties files manually to make changes to the server settings. If you manually edit the files, you should restart the server to ensure that the changes you make take effect.

Note: JVM refers to a Java virtual machine started by WebSphere on a node, and is also referred to as an application server.

The server.properties file

The server.properties file contains the following properties:

num_licenses
Specifies the number of licenses you purchased.

The value is an integer that is set during installation. There is no default.

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

licenseTracking
Specifies whether Host Publisher tracks license usage or not.

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

0 Host Publisher does not track license usage.

1 Host Publisher tracks license usage for all JVMs in a node. The Host Publisher Server tracks the number of Host Publisher connections to host or database resources and logs a message when the value exceeds the number of licenses purchased. The license usage information is written to a file named license*x*.txt in the `log` directory of the Host Publisher installation directory on the server, where the *x* is either 1 or 2.

The maximum size of the license usage files is 512 KB. When the file size of the license1.txt file reaches 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 each row representing one hour of operation. The values are separated by a space ( ).

1. Date
2. Time
3. The highest license count since the server was started
4. The highest license count in the last hour (the maximum of the last 60 entries)
5. The license count for each minute (1–60)

The value is binary. The default is 0.

%logFile
The name used as a template to generate names for each JVM file to which log messages are written. Refer to the maxLogFiles property comment in any ras*.properties file for a description of the algorithm for generating each JVM log file name. A ras*.properties file is generated for each WebSphere Application Server JVM configured to run Host Publisher. When editing the file, specify any backslash in the path with a double slash (\\).

%traceFile
The name used as a template to generate file names for each JVM file to which Host Publisher trace messages are written. Refer to the maxTraceFiles property comment in any ras*.properties file for a description of the algorithm for generating each JVM trace file name. A ras*.properties file is generated for each WebSphere Application Server JVM configured to run Host Publisher. When editing the file, specify any backslash in the path with a double slash (\\).

%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.

%HPAdminPortNumber
The port number used by Host Publisher Server on a machine to enable each installation running in a JVM on that machine to be administered remotely. Remote administration is currently based on remote method invocation (RMI).

You can specify any valid TCP port number. The default is 30099.

autoDeploy
Specifies whether Host Publisher deploys applications that have been transferred to the server when the server is started.

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.

docURLBase
Specifies an internal alias used by Host Publisher to point to documentation. This value should never be changed.

ServletURL
Specifies an internal alias used by Host Publisher to point to administration files. This value should never be changed.

The ras_xxx.properties file

The ras_*xxx*.properties file contains the following properties:

maxLogFiles
The maximum number of log information files.

The base log file name in server.properties is used as a template to generate unique sets of log files for each application server (JVM). The default base name for a log file is messages.txt, which can be changed in server.properties. The JVM running Host Publisher is the concatenation of the following values:

- The value of the hostpublisher.server.name property defined to the JVM (the default is HostPubServer if the property is not defined)
- The underscore (_) character
- The unique port number configured for that JVM's servlet engine (a random port number is generated if there is no servlet engine configured)
- Another underscore (_) character

The JVM ID (for example: *SSSS_PPPP_*) is then appended to the base file name to generate the template for the log files for a JVM, which becomes messages_*SSSS_PPPP_*.txt. Finally an index is added to this name (1, 2, 3, and so forth) to distinguish multiple log files. Therefore, if the port number for the application server running Host Publisher is 8120 (the configured default), and the application server name is HostPubServer (the configured default), the log file for that application server is named messages_HostPubServer_8120_.txt. With multiple log files configured, the log file names for this JVM are messages_HostPubServer_8120_1.txt, messages_HostPubServer_8120_2.txt, and so forth.

When messages_*SSSS_PPPP*_1.txt reaches maxTraceFileSize, it is closed and renamed to messages_*SSSS_PPPP*_2.txt. A new messages_*SSSS_PPPP*_1.txt is opened.

When messages_*SSSS_PPPP*_1.txt reaches maxTraceFileSize again, it is renamed to messages_*SSSS_PPPP*_2.txt, messages_*SSSS_PPPP*_2.txt is renamed to messages_*SSSS_PPPP*_3.txt, and a new messages_*SSSS_PPPP*_1.txt is opened.

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

maxLogFileSize
Specifies the maximum size, in kilobytes, that a message log file reaches before an additional log file is opened.

Note: You can only modify this property by editing the ras_*xxx*.properties file.

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

maxTraceFiles
The maximum number of trace information files.

The base trace file name in server.properties is used as a template to generate unique sets of trace files for each application server (JVM). The default base name for a trace file is trace.txt, which can be changed in server.properties. The JVM running Host Publisher is the concatenation of the following values:

- The value of the hostpublisher.server.name property defined to the JVM (the default is HostPubServer if the property is not defined)
- The underscore (_) character
- The unique port number configured for that JVM's servlet engine (a random port number is generated if there is no servlet engine configured)
- Another underscore (_) character

The JVM ID (for example: *SSSS_PPPP_*) is then appended to the base file name to generate the template for the trace files for a JVM, which becomes trace_*SSSS_PPPP_*.txt. Finally an index is added to this name (1, 2, 3, and so forth) to distinguish multiple trace files. Therefore, if the port number for the application server running Host Publisher is 8120 (the configured default), and the application server name is HostPubServer (the configured default), the trace file for that application server is named trace_HostPubServer_8120_.txt. With multiple trace files configured, the trace file names for this JVM are trace_HostPubServer_8120_1.txt, trace_HostPubServer_8120_2.txt, and so forth.

When trace_*SSSS_PPPP*_1.txt reaches maxTraceFileSize, it is closed and renamed to trace_*SSSS_PPPP*_2.txt. A new trace_*SSSS_PPPP*_1.txt is opened.

When trace_*SSSS_PPPP*_1.txt reaches maxTraceFileSize again, it is renamed to trace_*SSSS_PPPP*_2.txt, trace_*SSSS_PPPP*_2.txt is renamed to trace_*SSSS_PPPP*_3.txt, and a new trace_*SSSS_PPPP*_1.txt is opened.

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

maxTraceFileSize

Specifies the maximum size, in kilobytes, that a trace file reaches before an additional trace file is opened.

Note: You can only modify this property by editing the ras_*xxx*.properties file.

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

%logMask

Specifies the types of messages that are logged. Add the values for each of the following message types to determine the value for this property:

1	Informational messages
2	Warning messages
4	Error messages

The value is a decimal integer. The default is 4.

Note: Error messages are always logged.

%traceMask

Specifies the types of traces for the Server and the Integration Objects. This property does not affect JDBC or Host On-Demand tracing. Add the values for each of the following traces to determine the value for this property:

16	API traces
384	Entry and exit traces
1024	Miscellaneous traces

The value is a decimal integer. The default is 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.

If %HODDisplayTerminal is 1, when Host Publisher creates a host connection (for example, in response to a request from an Integration Object), it automatically creates a host terminal display. If this property is set to 0, this does not occur; however, whether or not the host terminal display is created automatically during host connection creation, a Host Publisher administrator can use Host Publisher Server Administration to turn host terminal displays on or off for individual host connections.

%HODMacroTracingLevel
Specifies the level of tracing for the Host On-Demand macros.

The value is an integer in the range 0 to 3, where 3 is the maximum level of tracing. The default is 0, which means there is no tracing.

%HODPSTracingLevel
Specifies the level of tracing for the Host On-Demand presentation space.

The value is an integer in the range 0 to 3, where 3 is the maximum level of tracing. The default is 0, which means there is no tracing.

%HODSessionTracingLevel
Specifies the level of tracing for Host On-Demand sessions.

The value is an integer in the range 0 to 3, where 3 is the maximum level of tracing. The default is 0, which means there is no tracing.

%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.

Appendix D. Examples for designing custom Web pages

Use the following examples to design custom Web pages. To cut and paste an example, use the online version of this book, located at Start > Programs > Host Publisher [Server or Studio] > Administrator's and User's Guide.

Note: You can find more code examples on the Host Publisher Web page (http://www.ibm.com/software/webservers/hostpublisher/library.html) under Product documentation.

Java access to page parameters

How can I get access to parameters passed to my Web page?

Write a Java scriptlet to call the request.getParameter method and specify the name of the input parameter.

This example gets the values of input parameters tu_in and dupno_in and places them in String variables. They are then displayed as part of HTML h2 headers.

```
<%
   String tu_num = request.getParameter("tu_in");
   String dup_num = request.getParameter("dup_in");
%>

<h2>Val1 is <%=tu_num %></h2>
<h2>Val2 is <%=dup_num %></h2>
```

Redirecting based on Integration Object results

How can I test the output of an Integration Object and call subsequent JavaServer Pages (JSPs) based on the results of my test?

Write a Java scriptlet to get the desired property from the executed Integration Object. Test the property and issue a response.sendRedirect to the appropriate JSP based on the test.

This example calls getTaxmenuid and tests the value to determine the page to redirect to.

```
<%
   String status=Tax_Menu_Init.getTaxmenuid();
   if (status.indexOf("Sign") != -1){
      response.setStatus(response.SC_MOVED_TEMPORARILY);
      response.setHeader("Location","TaxSignon_Error.jsp");
   out.close();
 return;
}
%>
```

Invoking Integration Objects based on previous Integration Object results

How can I test the output of an Integration Object and not invoke a subsequent Integration Object if the first Integration Object failed?

Write a Java scriptlet to test a property from the first executed Integration Object. Use an IF statement for the test and wrap the execution of the subsequent bean within the IF.

In this example, the Tax_Details_F12 Integration Object will execute ***only*** if the word DUPLICATE is ***not*** found in variable status.

```
<%
   String status = Tax_Addr.getTaxstatus();
   if (status.indexOf("DUPLICATE") == -1) { %>

      <jsp:useBean id=Tax_Details_F12" TYPE="IntegrationObject.Tax_Details_F12"
      <class="IntegrationObject.Tax_Details_F12" SCOPE="request">
      </jsp:useBean>
      <jsp:setProperty name="TaxDetails_F12" property="*" />
      <% Tax_Details_F12.doHPTransaction(request, response); %>

<%}%>
```

Building dynamic HTML based on Integration Object properties

How can I build dynamic output based on the properties of an Integration Object?

Write a Java scriptlet that will use out.println to write output into the HTML stream. Properties of the Integration Object can be embedded in the out.println statements.

```
<%
String empno;
try
{
empno = Dbfirst.getDANAPEMPLOYEEEMPNO_(0);
out.println("<P>Employee number: <b>" + empno + "</b><p>");
out.println(
   Dbfirst.getDANAPEMPLOYEELASTNAME_(0) + "," +
   Dbfirst.getDANAPEMPLOYEEFIRSTNME_(0) + "" +
   Dbfirst.getDANAPEMPLOYEEMIDINIT_(0) + "." +
);

out.println("<p>Show <A HREF=\"showfirst.jsp?empno=" + empno +
 "\">Next</A> Employee");
out.println("<p>Show <A HREF=\"Deleted.jsp?empno=" + empno +
 "\">Delete</A> Employee");

}
catch(Exception e)
{
}
%>
```

Validating user input

How can I validate user input from an HTML form?

Use JavaScript to validate user input on an HTML page.

In this example, the function is called by *onSubmit="return padzero(this);"* on the input HTML FORM statement. Function padzero ensures that the input value is numeric and is between 1 and 999999 inclusive.

```
<SCRIPT LANGUAGE="JavaScript1.1">
function padzero(f)
{
   var num = parseInt(f.elements[0].value,10);
   if (num < 1 || num > 999999 || isNaN(num) ||
     num != f.elements[0].value )
    {
     alert("Employee Number not valid ... value must be between" + " " 1 and 999999");
     return false;
    }
   var len = f.elements[0].value.length;
   var zeros = "";
   for (var i = len; i < 6; i++)
    zeros = zeros.concat("0");
   f.elements[0].value = zeros.concat(f.elements[0].value);
   return true;
}
</SCRIPT>
```

Testing for successful database record deletion

How can I test to ensure a database access record delete is successful?

Use a Java Scriptlet to query the Integration Object.

This example calls getHPubNumberOfRowsChanged and tests to see if a row was changed.

```
<%
   DbDelete.setHPubStartPoolName("Db");
   DbDelete.doHPTransaction(request, response);
   int changed;
   changed = DbDelete.getHPubNumberOfRowsChanged();
   if (changed == 0)
{   out.println("Error deleting employee number");   }
   else
   if (changed == -1)
   {   out.println("No entry found for employee number ");   }
   else
   {   out.println("Successfully deleted employee number");   }
%>
```

Testing for successful database record addition

How can I test to ensure a database access record addition is successful?

Use a Java Scriptlet to query the Integration Object.

This example calls getHPubNumberOfRowsChanged and tests to see if a row was changed. Note that a write of the empno and name into the HTML stream is also in the scriptlet.

```
<%
   DbAddMin.setHPubStartPoolName("Db");
   DbAddMin.doHPTransaction(request, response);
   int changed;
   changed = DbAddMin.getHPubNumberOfRowsChanged();

   if (changed == 0)
   {   out.println("Error adding employee number");   }
   else
   {   out.println("Successfully added employee number");   }

   out.println("<b>" + DbAddMin.getEmpno());
out.println("</b> (" + DbAddMin.getLast() + "," + DbAddMin.getFirst()    + " " +
 DbAddMin.getMiddle() + ".).");

   if (changed == 0)
   {   out.println("<p>Employee number may already be assigned.");   }
%>
```

Passing Java variables to JavaScript function

How can I pass Java variables to JavaScript?

In this example, maxfields is passed to JavaScript function CheckData.

```
<% int maxfields = 99; %>

<SCRIPT Language="Javascript">
function CheckData(formitem, max)
{
   if (formitem.fgn_number..value > max)
      alert("Value is too large. Please re-enter")
}
</SCRIPT>
```

```
Sample HTML form to call function:
   <FORM Name="testform" Method="pose"
      onSubmit="return CheckData(this,<%= maxfields %>)" >
     <input type="text" name="fgn_number" size="12" maxlength="12"></p>
   </FORM>
```

Using Java to display variables passed into a page

How do I display the variables that are passed into a page?

Use a Java scriptlet and call request.getParameter("<name>"), then use out.println. Parameters are sent to pages as Name/Value pairs.

```
<%
   out.println("Test Bad");
   String payee_name = "";
   String fgn_input = request.getParameter("fgn_number");
   out.println(fgn_input + " ");
   out.println(fgn_input.length());
%>
```

Using Java to pad an input value and passing it to an Integration Object

How can I pad input data to the correct length without using JavaScript on the client side?

You can use Java to pad an input variable to the proper length for the host application.

In this example, the dcn_number is padded to be 9 characters.

```
<%
   String dcn_input = request.getParameter("dcn_number");
   String final_dcn_value = "";
   dcn_input = dcn_input.trim();
   for( int i = dcn_input.length(); i < 9; i++)
     final_dcn_value = final_dcn_value.concat("0");
   dcn_input = final_dcn_value + dcn_input;
%>


<jsp:useBean id="Okdhs_dbupdate" class="IntegrationObject.Okdhs_dbupdate
 SCOPE="request">
</jsp:useBean>
<jsp:setProperty name="Okdhs_dbupdate" property="*" />
<% Okdhs_dbupdate.setHPubStartPoolName("OKLADHSDB"); %>
<% Okdhs_dbupdate.setDcn_number(dcn_input); %>
<% Okdhs_dbupdate.doHPTransaction(request, response); %>
```

Using a function type passed from a hidden HTML form variable to determine page to execute

How can I use one input form to redirect to different host functions?

Use an onClick method to set a function type in a hidden HTML form variable. This variable can then be tested by Java to determine correct JSP to handle the function. A sample form follows the example.

```
<%
   String function_selected = request.getParameter("funtype");
  try {
     db_pin_value = Okdhs_dbselect.getDMCPEAKEOKLADHSDCN_(0);
     if (db_pin_value.indexOf(dcn_input( != -1)
     {
      if(function_selected.indexOf("CHANGE") == -1)
        newurl = "cfrr_page.jsp?fgn_number=" + fgn_input;
      else
        newurl = "change_pin.jsp?fgn_number=" + fgn_input;
     } else {
      newurl = "cfrr_error.jsp?fgn_number="+
        request.getParameter("fgn_number");
      response.sendRedirect(newurl);
     }
    }
  catch(exception e)
  {
   newurl = "cfrr_error.jsp?fgn_number=" +
      request.getParameter("fgn_number");
   response.sendRedirect(newurl);
  }
%>
```

Sample HTML form:

```
<form name="subform" method="POST"
   ACTION="<%=response.encodeUrl("csml_page.jsp") %>">
   <p align="center"><font face="Comic Sans MS">Please type your identification
   number</font>
   <input type="text" name="fgn_number" size="12" maxlength="12></p>
   <p align="center"><font face="Comic Sans MS">Please enter your PIN</font><input
type="password" name="pin_number" size="9" maxlength="9">>/p>

   <input type="hidden" name="funtype" size="6" maxlength="6">>

   <p align="center"><input type="submit" value="Inquire" name="B1"
onClick="subform.funtype.value='INQUIRE'">&nbsp;&nbsp;
   <input type="reset" value="Clear Form" name="B2">&nbsp;&nbsp;
   <input type="submit" value="Change PIN" onClick="subform.funtype.value='CHANGE'">
   </p></p>
</form>
```

Using Java to prevent blank lines in an HTML table

If I extract more data than is available, how do I prevent blank lines from being added to the HTML table?

Use a Java scriptlet to test the property of the Java bean. If it is blank, break out of the repeat loop.

```
<HTML>
<BODY>
<jsp:useBean id="Ha_mcat1" class="IntegrationObject.Ha_mcat1" scope="request">
</jsp:useBean>
<jsp:setProperty name="Ha_mcat1" property="*" />
<% Ha_mcat1.setHPubStartPoolName("mcatpool"); %>
<% Ha_mcat1.doHPTransaction(request, response); %>
<P>Search Results:<TABLE BORDER>
<th>title</th>
<th>author</th>
<% for (int idx1 = 0; idx1 %= 2147483647; idx1++) { try { %><tr>
<%
if (Ha_mcat1.getTitletitle(idx1).indexOf(" ")!=-1) break;
%>
<td><%= Ha_mcat1.getTitletitle(idx1) %></td></tr>
<% }
catch (java.lang.ArrayIndexOutOfBoundsException ae) {
break;
}
catch (java.lang.NullPointerException ae) {
}
}
%>
</TABLE>
</Body>
<% out.close(); %>
</HTML>
```

Using Java to control display of an HTML table based on host results

If there is no data for the table extracted from the host, how can I prevent the display of an empty table?

One way is to also extract a status line from the terminal screen, assuming a status line exists. Then use an IF statement to test the status line to determine if the table should be displayed.

Note: This pertains only to Integration Objects created by the Host Access application.

```
<% String status_value = Spurs_add1.getStatus_line();
   if (status_value.indexOf("VENDOR NOT FOUND") == -1 { // IF "Vendor not found"
   was in status line, do not display the table and instead display a
   "Vendor not found" message.
%>
```

```
<P>Vendor Results:<TABLE BORDER>
<th>Vendors</th>
<% for (int idx1 = 0; idx1 <= 2147483647; idx1++) {
try { %><tr>
<td><%= Spurs_add1.getVendors_data(idx1_ %></td>
<% }catch (java.lang.ArrayIndexOutOfBoundsException ae) {
break;
}
catch (java.lang.NullPointerException ae) {
break;
}
}
%>
</TABLE>
<% } else { %>
<h3 align="center">Vendor not found in vendor file.</h3>
<% } %>
```

Determining number of page downs and tabs for making a selection

How can I code the Host Publisher application such that it selects one item when the host application contains lists of items on multiple terminal screens?

There are two ways to accomplish this.

1. Use Integration Object chaining and return one screen of items at a time. When the user clicks an item in the list box, use the index of the selected item to select the entry on the screen.
2. Return all the items and display them in a list box. Position the host screen on the first item. Use the following example to calculate the number of page downs and tabs required to get to the proper entry. The value passed in from the previous page is the index of the item in the list box.

Note: This pertains only to Integration Objects created by the Host Access application.

This example assumes there are 9 items per host screen. If your application has a different number per screen, change this value.

```
<HTML>
<BODY>
<!-- Retrieve the parameter passed into this page -->
<!-- This parameter indicates which entry was selected from the inspection list -->
<% int selNum = new Integer(request.getParameter("sel")).intValue(); %>

<!-- Calculate the number of pages downs and tabs -->
<% int numPgDowns =0;
   numPgDowns = selNum / 9;
   int numTabs =0;
   numTabs = selNum%9;
   out.println("numTabs string is "+numTabs);
   out.println("selNum string is "+selNum);
%>

<!-- Build the strings to cause the macro to return to the proper screen -->
<%
   String pgDownString = new String();
   String tabsString = new String();
   //
   //out.println("adding "+numPgDowns+" page downs");
   //out.println("adding "+numTabs+" tabs");

   for(int i =0; i < numPgDowns; i++)
      pgDownString=pgDownString+"[pagedn]";

   //out.println("page down string is "+pgDownString);
   for(int j =0; j < numTabs; j++)
      tabsString += "[tab]";

%>

<jsp:useBean id="ha_lab4" class="IntegrationObject.Ha_lab4" scope="request">
</jsp:useBean>
<jsp:setProperty name="Ha_lab4" property="*" />

<% Ha_lab4.setHPubStartPoolName("Halab4"); %>
<% Ha_lab4.setOpt(pgDownString+tabsString); %>
<%Ha_lab4.doHPTransaction(request, response); %>
```

Changing the action value of a form based on the clicked button

How can I use one form to go to a different page based on the user's selection?

The action value for a form can be changed dynamically.

```
<%
   String invoice_url = response.encodeUrl("WCB_clinic.jsp");
   String cheque_url = response.encodeUrl("WCB_chk_logon.jsp");
%>


<FORM>
   <INPUT type="submit" value="Enter Invoice"
      onClick = "document.forms[0].action = '<
%=invoice_url %>'">

   <INPUT type="submit" value="View Check Information"
      onClick = "document.forms[0].action = '<
%=cheque_url %>'">
```

Using HTTP Session object to pass values

How can I pass values from one page to a subsequent page?

Store the values in an HTTP session object.

```
** First page stores the values.**
   HttpSession ssess = request.getSession(false);
   if (sess!=null) sess.putValue("caregiver_type", WCB_caregiver.getCaregiver_type());
   if (sess!=null) sess.putValue("caregiver_idn", WCB_caregiver.getCaregiver_idn());
%>

** Subsequent page retrieves the values.**
<%
HttpSession sess = request.getSession(false);
   if (sess != null) WCB_fee_code.setCaregiver-type((String)
sess.getValue("caregiver_type");
   if (sess != null) WCB_fee_code.setCaregiver_idn((String)
sess.getValue("caregiver_idn"));
%>
```

Disabling the browser back button

How can I disable the browser back button?

Place the following JavaScript on every page, including the logon page.

This example uses the browser's history object to tell the browser to go forward after the page is loaded.

```
<HTML>
<SCRIPT Language="Javescript">
function goHist(a)
{
   history.go(a);   //Go forward one.
}
</SCRIPT>
</head>

<<body onLoad="goHist(1);">
   some HTML ...
</body>
</HTML>
```

Using Java to control which HTML table to display based on host results

If there are multiple tables on the JSP that can display, how can I prevent the display of an empty table HTML on the JSP?

You can extract a property and use Java logic to test that property. Then use an IF statement to test the status line to determine if the table should display. Note the use of the ELSE statement.

```
<HTML>
<BODY>
<jsp:useBean id="Mcat_conditional" class="IntegrationObject.Mcat_conditional"
 scope="request>
</jsp:useBean>
<jsp:setProperty name="Mcat_conditional" property="*" />
<% Mcat_conditional.setHPubErrorPage("error.jsp"); %>
<% Mcat_conditional.setHPubStartPoolName("Mcat_keyw"); %>
<% Mcat_conditional.doHPTransaction(request, response); %>
<% String s = Mcat_conditional.getJim();
   if (s.indexOf("NAUGLE J.C.") != -1) {
%>
<P>Author Index Results:<TABLE BORDER>
<th>indexall_data</th>
<% for (int idx1 = 0; idx1 <= 2147483647; idx1++) {
try { %><tr>
<td><%= Mcat_conditional.getAuthor_indexall_data (idx1) %></td>
<% }
catch (java.lang.ArrayIndexOutOfBoundsException ae) {
break;
}
catch (java.lang.NullPointerException ae) {
break;
}
}
%>
</TABLE>
<% } else { %>
<P>Author Guide Results:<TABLE BORDER>
<th>guidelast</th>
<th>guidefirst</th>
<% for (int idx2 = 0; idx2 <= 2147483647; idx2++) {
try { %><tr>
<td><%= Mcat_conditional.getAuth_guidelast(idx2) %></td>
<td><%= Mcat_conditional.getAuth_guidefirst(idx2) %></td>
<% }
catch (java.lang.ArrayIndexOutOfBoundsException ae) {
break;
}
catch (java.lang.NullPointerException ae) {
break;
}
}
%>
</TABLE>
<% } %>
</BODY>
<% out.close(); %>
</HTML>
```

Appendix E. 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

This information contains sample application programs in source language, which illustrates programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Programming interface information

This Administrator's and User's Guide contains information on intended programming interfaces that allow the customer to write programs to obtain the services of Host Publisher.

Appendix F. Trademarks

The following terms are trademarks of the IBM Corporation in the United States or other countries or both:

- AIX
- Client Access
- DB2 Universal Database
- IBM
- OS/400
- RS6000
- 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.

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.

UNIX is a registered trademark in the United States and other countries licensed exclusively through The Open Group.

Index

Readers' Comments — We'd Like to Hear from You

IBM® WebSphere® Host Publisher
Administrator's and User's Guide
Version 3 Release 5

Publication No. GC31-8728-02

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
GC31-8728-02

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
Software Reengineering
Department G7IA/Building 503
Research Triangle Park, NC
27709-9990

Fold and Tape **Please do not staple** Fold and Tape

GC31-8728-02

Cut or Fold
Along Line

Part Number: CT69ZNA

Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.

(1P) P/N: CT69ZNA

GC31-8728-02