IBM WebSphere Development Tools for AS/400

An Introduction

- Learn how to use the IBM WebSphere Development tools for AS/400
- Create new e-business applications using the wizards
- Work with the WebFacing tool for 5250 applications

Redpaper

Masahiko Hamada
Drew Glasgow
Garry Kipfer

ibm.com/redbooks
Take Note!

Before using this information and the product it supports, be sure to read the general information in Appendix C, “Special notices” on page 97.

First Edition (March 2001)

This edition applies to WebSphere Development tools for AS/400 V4R5.

This document created or updated on March 14, 2001.

Comments may be addressed to:
IBM Corporation, International Technical Support Organization
Dept. JLU Building 107-2
3605 Highway 52N
Rochester, Minnesota 55901-7829

When you send information to IBM, you grant IBM a non-exclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

© Copyright International Business Machines Corporation 2001. All rights reserved.
Note to U.S Government Users - Documentation related to restricted rights - Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract with IBM Corp.
Preface

IBM WebSphere Development Tools for AS/400 (5769-CL3) is a complete package of premium AS/400 and iSeries program development tools. These tools help you quickly develop iSeries and AS/400-based applications for e-business. You can do this by using your choice of traditional 5250 green screen interfaces, HTML browser interfaces, or GUI-based interfaces on workstations running the Windows operating system.

The features of WebSphere Development Tools allow you to:

- Generate Java to Web-enable your RPG applications.
- Write host or server code with the CODE/400 workstation-based client/server development environment.
- Develop and maintain applications in many AS/400 languages, including RPG and Java.

This IBM Redpaper targets IBM marketing personnel, Business Partners, and AS/400e and iSeries customers who are looking to extend and expand their information server into e-business. It shows you how to install, tailor, and configure WebSphere Development Tools for AS/400.

The team that wrote this redpaper

This redpaper was produced by a team of specialists from around the world working at the International Technical Support Organization Rochester Center.

Masahiko Hamada is an Advisory International Technical Support Specialist for the iSeries and AS/400 servers at the International Technical Support Organization, Rochester Center. He writes extensively and teaches IBM classes worldwide on all areas of iSeries e-business. Before joining the ITSO in 2000, he worked in the iSeries field support center in Japan as an AS/400 System Specialist.

Drew Glasgow is a Software Support Specialist in the Database and Languages Team in the UK AS/400 Support Center. He has over 20 years experience in the applications development and maintenance field and he has worked on a variety of hardware and software platforms. He has 10 years experience working with iSeries systems and has worked for IBM for the past 5 years. His main areas of expertise include RPG, Cobol, and DB2 UDB for AS/400.

Garry Kipfer is a Consulting IT Specialist working in the Technical Sales and Support group in Toronto, Canada. He has over 25 years experience in midrange systems with IBM. During that period, he has worked in traditional Application Development, Client Server applications, Data Warehousing, WebSphere application implementation, AS/400 performance, and DB2/400 UDB.

Thanks to the following people for their invaluable contributions to this project:

Alan Chao
Ho-Kee Chiu
George Farr
Willson Hui
Comments welcome

Your comments are important to us!

We want our redpapers to be as helpful as possible. Please send us your comments about this redpaper or other Redbooks in one of the following ways:

- Use the online evaluation form found at ibm.com/redbooks
- Send your comments in an Internet note to redbook@us.ibm.com
Chapter 1. WebSphere Development Tools for AS/400

The IBM WebSphere Development Tools for AS/400 (WDT/400) product is a suite of tools for developing e-business and traditional and client/server applications for the iSeries server. Figure 1 shows the various WebSphere-related products.

For the iSeries server, WebSphere Development Tools for AS/400 (5769-CL3) is a complete package of premium OS/400 program development tools. These tools help you quickly develop OS/400-based applications for e-business using your choice of traditional 5250 green-screen interfaces, HTML browser interfaces, or GUI-based interfaces on workstations running the Windows operating system. You can use these tools to:

- Generate Java to Web-enable your RPG applications
- Write host or server code with the CODE/400 workstation-based client/server development environment
- Develop and maintain applications in many OS/400 languages, including RPG and Java

WebSphere Development Tools for AS/400 includes the following components:

- WebSphere Studio for AS/400
- VisualAge for Java for AS/400
- VisualAge RPG
- CODE/400

Additional information on WebSphere Development Tools for AS/400 is available at: http://www-4.ibm.com/software/ad/wdt400/
1.1 WebSphere Studio for AS/400

WebSphere Studio for AS/400 is optimized specifically for WebSphere Application Server applications. However, you can also use many WebSphere Studio for AS/400 tools for developing applications for other Web servers or Web application servers. WebSphere Studio for AS/400 development tools run on a Windows workstation. You can deploy the resulting application to an iSeries server running either WebSphere Application Server Standard Edition or Advanced Edition as shown in Figure 2.

Figure 2. WebSphere Studio for AS/400 overview

WebSphere Studio for AS/400 provides tools that you can use to manage your Web application project and for creating HTML, Java, and JSPs, including graphics and database access. WebSphere Studio for AS/400 maintains project files in the file system and provides support for team development and version control tools. WebSphere Studio for AS/400 deployment features enable you to configure the projects to deploy to a number of locations, such as the WebSphere Application Server or the WebSphere Test Environment of VisualAge for Java.

WebSphere Studio for AS/400 now incorporates AS/400 Affinity. AS/400 Affinity is designed for OS/400 application developers who want to develop e-business applications but do not have time to acquire the additional skills required to develop Web-enabled applications. WebSphere Studio for AS/400 now gives application developers, with traditional iSeries programming skills, the ability to quickly develop e-business applications without having to learn Java and other Web application skills. As a developer, you can concentrate on the underlying business application logic residing on the OS/400 host, using current ILE RPG (or any other high-level language) skills. You can use the intuitive aspects of WebSphere Studio for AS/400 to design the Web-based front end. And, you can generate the JavaServer Pages (JSP) and Java servlets required to enable the new e-business application.

WebSphere Studio for AS/400 wizards take you through the steps required for creating Web input and output pages. They allow you to define the parameters associated with the design-time controls on the Web pages, and link the fields on the Web pages to the parameters in the business logic. You can do all these tasks without having to deal with JavaServer Pages, JavaScript code, and servlet code.
WebSphere Studio for AS/400 provides the ability to create new e-business applications using a Web-based front end that communicates with the business logic of an ILE language program residing on the OS/400 host. You can create your Web input and output pages with WebSphere Studio for AS/400 and its Page Designer, or generate input and output pages from templates listed in the new Web interaction wizard.

The AS/400 Web interaction wizard also helps you perform these tasks:

- Define the parameters associated with the design-time controls on your Web pages, without dealing directly with JavaServer Pages (JSP), JavaScript code, and servlet code.
- Link the fields on your Web pages to the parameters in the ILE business logic.

Use the Publishing Setup wizard to identify your iSeries servers and to define the publishing information used by the WebSphere Studio for AS/400 publishing function. Once this is done, you can deploy an iteration of your application for testing, or you can deploy the final version for production purposes.

WebSphere Studio for AS/400 includes the following OS/400-specific functions:

- A Web Interaction wizard that creates the underlying code used to communicate between the Web pages and the business logic
- OS/400-specific design-time controls for your Web input and output pages
- A Publishing Setup wizard designed to help you set up your publishing information before you deploy iSeries e-business applications to either a test or production environment

WebSphere Studio for AS/400 contains several wizards that guide you through such tasks as SQL statement generation and creating Web pages to interact with databases and JavaBeans. You can also use the WebSphere Studio for AS/400 Page Designer to edit these generated pages. The following tools are included in WebSphere Studio for AS/400 Version 3.5:

- Studio Workbench
- Studio Wizards
- Page Designer

1.2 VisualAge for Java

VisualAge for Java is an integrated visual environment that supports the complete cycle of Java program development. VisualAge for Java gives you everything you need to perform development tasks as shown in Figure 3 on page 4.
VisualAge for Java provides extensive functionality across the entire
development life-cycle and includes tools for Java code editing and debugging,
Java ServerPage debugging, and the WebSphere Test Environment. VisualAge
for Java also includes a repository that stores project source and compiled code
and an import/export facility that enables interaction with the file system.

One of the most important features of VisualAge for Java is the WebSphere Test
Environment. This feature provides application and Web server environments on
a development machine. This enables you to test and debug the resources of a
Web site locally.

You can use VisualAge for Java’s visual programming features to quickly develop
Java applets and applications. In the Visual Composition Editor, you can simply
point and click to:

- Design the user interface for your program.
- Specify the behavior of the user interface elements.
- Define the relationship between the user interface and the rest of your
  program.
- Generate the Java code to implement what you design in the Visual
  Composition Editor. In many cases, you can design and run complete
  programs without writing any Java code.
- SmartGuides (wizards) to lead you quickly through many tasks, including
  creating new applets, packages, or classes.

The VisualAge for Java Integrated Development Environment (IDE) automatically
compiles Java source code into Java bytecode. When source code is imported
into the workspace (from .java files) or added from the repository, it is compiled
and analyzed with respect to the existing workplace contents.

- When you change, delete, move, copy, or rename program elements, the
  affected code is automatically recompiled to flag any problems. If you
  introduce an error, the IDE warns you and gives you the option to fix the
  problem immediately or add the problem to the All Problems page and fix it
  later.
- VisualAge for Java has a sophisticated code management system that makes
  it easy for you to maintain multiple program editions. When you want to freeze
the state of your code at any point, you can version an edition. This marks the particular edition as read-only and enables you to give it a name. It provides a method to preserve snapshots of significant checkpoints in a development cycle.

With VisualAge for Java, you can develop very robust code. Specifically, you can:

- Build, modify, and use JavaBeans.
- Browse your code at different levels, such as project, package, class, or method.
- Use the integrated visual debugger to examine and update code while it is running.
- Use the distributed debugger to debug Java applications that are developed outside the IDE.

### 1.3 VisualAge RPG

VisualAge RPG features point-and-click simplicity for RPG programmers at the workstation. Use VisualAge RPG to capitalize on your existing RPG skills and to start developing powerful e-business applications. The visual development environment allows you to build RPG applications with a Windows GUI and transparent access to OS/400 resources. You can edit, compile, debug, and execute RPG on the client. The VisualAge RPG compiler is the same code base as the host ILE RPG compiler, so you can easily move logic between the client and server.

After building your applications, they run on a workstation and can access OS/400 host data and other OS/400 objects. VisualAge RPG integrated components allow application developers to preserve their current skills and to easily develop OS/400 applications with graphical user interfaces.

With VisualAge RPG, you can build an application from the top down. You start by focusing on the look and feel of the interface. Then, you tie all the parts together with workstation RPG logic that you write in the VisualAge RPG language. You can reuse RPG logic and Display Files (DSPF) from an existing application.

VisualAge RPG offers:

- A Visual Development Environment
- An RPG compiler on the workstation
- A programmable language-sensitive editor
- A display file import facility
- Transparent PC-to-OS/400 connectivity
- Source-level debugger
- The ability to generate Java applications and Java applets from the VisualAge RPG source

### 1.4 CODE/400

CODE/400 features workstation-based editing, compiling, and debugging of your OS/400 applications. CODE/400 is the preferred development environment for writing host applications for the iSeries server. It is significantly more productive than IBM Application Development Toolset (ADTS). CODE/400 capabilities support RPG, COBOL, C, C++, CL, DDS, and Java.
CODE Designer is a WYSIWYG DDS tool for creating workstation, printer, and database files for the iSeries server. Enhanced debugging capabilities allow you to debug multi-tier e-business applications.

You can use Client Access, Personal Communications, or TCP/IP to connect to the iSeries server. However, CODE/400 doesn't require a continuous connection to the iSeries server. Many of the CODE/400 features can function in a disconnected mode. This gives you the ability to develop your applications at home, on the road, or elsewhere.

Simplify your work by setting up the CODE Project Organizer to access and manage your OS/400 files, members, objects and Application Development Management (ADM) parts. Create a CODE Project Organizer project and set up filters to gain quick GUI access to frequently used OS/400 objects, members, ADM projects, groups, and parts. Use the pop-up menus on these items to perform actions, such as edit, compile, and debug. Use the CODE Actions notebook to create and manage user-defined actions.

Share your projects and actions with other team members to decrease set-up time by importing and exporting projects and importing and exporting actions.

Use CODE Designer to create, design, and update your display and printer files. CODE Designer allows you to graphically navigate through records, keywords, and fields. This allows you to see the DDS source for individual records, fields, and so on, as well as the entire source file.

You can group records together to design screens and reports. CODE Designer drag-and-drop capabilities simplify screen and report design, and provide you with built-in DDS verification and compile features. You can also browse the source listing for your display or printer file.

When your program is finished, and there are no syntax errors, you can save time by using the Program Verifier before you compile your program on the iSeries server. Program Verifier checks for compile errors on your PC before you even send your compile request to the iSeries server. Program Verifier is a handy tool when you are writing code but are not connected to an iSeries server. Program Verifier brings up the Error List, which helps you view and manage any errors found.

Once you are satisfied with the state of your code, invoke the Program Generator to select the redesigned and user-friendly compile options you desire for your program. The Program Generator provides the ability to update and create programs if you are using ILE modules. Program Generator notifies you when your compile is complete and, if there were any errors, the Error List appears. The Program Generator also supports Java and gives you the ability to compile on the OS/400 host and run Java programs remotely on the iSeries server.

The CODE Debugger allows you to browse source, set, delete, enable, and disable watch and line breakpoints as well as step through your code. The Debugger is a powerful tool that enables you to get your applications up and running quickly.
1.5 Summary

WebSphere Development Tools for AS/400 is a suite of tools that allows you to do all of your development and maintenance work on the workstation, as well as produce and deploy iSeries-based e-business solutions. Common components allow you to learn how to work with something once and leverage that knowledge across all of your development activities.

These tools help you to:

- Quickly develop OS/400-based applications for e-business using your choice of traditional 5250 green-screen interfaces, HTML browser interfaces, or GUI-based interfaces on workstations running the Windows operating system.
- Generate Java to Web-enable your RPG applications.
- Write host or server code with the CODE/400 workstation-based client/server development environment.
- Develop and maintain applications in many iSeries languages, including RPG and Java.
Chapter 2. Before you begin

This chapter discusses installing and configuring the products that are required to generate the environment necessary to enable the iSeries server as a Web server and to develop the e-business aspect of the application.

The components discussed here are:
- WebSphere Development Tools for AS/400
- IBM HTTP Server for AS/400
- WebSphere Application Server for AS/400
- WebSphere Administrative Console

2.1 Installing WebSphere Development Tools for AS/400

For the purposes of this redpaper, we installed the WebSphere Development Tools for AS/400 on a Windows NT workstation.

2.1.1 Prerequisite information

Before you install WebSphere Development Tools for AS/400 on your workstation, make sure that all hardware and software prerequisites have been satisfied. See Table 1 and Table 2 for workstation hardware and software requirements.

<table>
<thead>
<tr>
<th>Component</th>
<th>Memory minimum</th>
<th>Memory recommended</th>
<th>Disk minimum</th>
<th>Disk recommended</th>
</tr>
</thead>
<tbody>
<tr>
<td>WebSphere Studio for AS/400</td>
<td>96 MB</td>
<td>128 MB</td>
<td>385 MB</td>
<td>500 MB</td>
</tr>
<tr>
<td>VisualAge RPG</td>
<td>64 MB</td>
<td>96 MB</td>
<td>235 MB</td>
<td>235 MB</td>
</tr>
<tr>
<td>CODE/400</td>
<td>64 MB</td>
<td>96 MB</td>
<td>235 MB</td>
<td>235 MB</td>
</tr>
<tr>
<td>VisualAge for Java for AS/400</td>
<td>128 MB</td>
<td>192 MB</td>
<td>495 MB</td>
<td>550 MB</td>
</tr>
<tr>
<td>Total product</td>
<td>128 MB</td>
<td>256 MB</td>
<td>1.3 GB</td>
<td>1.5 GB</td>
</tr>
</tbody>
</table>

Table 1. Hardware requirements

<table>
<thead>
<tr>
<th>Component</th>
<th>Development time</th>
<th>Run time</th>
</tr>
</thead>
<tbody>
<tr>
<td>WebSphere Studio for AS/400</td>
<td>Windows 95, 98, NT, or 2000; Microsoft Internet Explorer 4.0 or later; IBM OS/400 V4R4 or later; WebSphere Application Server V3.02 or later</td>
<td>OS/400 V4R4 or later; Netscape 4.7 (provided) or later or Internet Explorer 4.0 or later; IE5.0 for Page Detailer; WebSphere Application Server V3.5 or later</td>
</tr>
<tr>
<td>VisualAge RPG</td>
<td>Windows 95, 98, NT, or 2000; ADTS (5769-PW1); OS/400 V4R3 or later</td>
<td>Windows 95, 98, NT, or 2000 or any JRE V1.2 or later (including a browser); OS/400 V4R3 or later</td>
</tr>
<tr>
<td>CODE/400</td>
<td>Windows 95, 98, NT, or 2000; ADTS (5769-PW1); OS/400 V4R3 or later</td>
<td>N/A</td>
</tr>
</tbody>
</table>

Table 2. Software requirements
2.1.2 Installing WebSphere Development Tools for AS/400

To install WebSphere Development Tools on your workstation, perform the following steps:

1. Insert the WebSphere Development Tools for AS/400 CD-ROM Disk 1 in the CD-ROM drive of your Windows NT workstation.

2. If Autorun is enabled, the Windows Install Shield program initiates automatically. If Autorun is not enabled, run the Windows Install Shield program by using Windows Explorer to navigate to your CD-ROM drive. Double-click the setup.exe file.

3. Follow the instruction panels in the WebSphere Development Tools for AS/400 InstallShield program. You should always reboot your workstation when prompted.

Before you proceed further, you should load the latest version of the WebSphere Development Tools for AS/400 PTF service pack. For the latest PTF service pack information, visit the Web site at: http://www.ibm.com/software/ad/wdt400/

2.2 Setting up the default instance of the WebSphere Application Server

This section covers installing and configuring the environment required to run the servlets and JavaServer Pages discussed in this redpaper. This includes the IBM HTTP Server for AS/400 and the IBM WebSphere Application Server (WAS).

The following sections specifically deal with installing and configuring the default instance of the WebSphere Application Server, HTTP server, and the Administrative Console. Users who are experienced in these areas may want to skip this section and continue with 2.3, “Setting up your instance of the WebSphere administrative server” on page 28.

In this redpaper, all references to WebSphere without other designation should be interpreted as references to IBM WebSphere Application Server for AS/400, Standard Edition, Version 3.5.

Installing and configuring the WebSphere Application Server on the iSeries server should take a minimal amount of time and effort. Before you begin, make sure your system meets the iSeries software prerequisites. For more information about minimum software requirements, refer to WebSphere for AS/400 Documentation Center. The online version can be accessed at the AS/400 WebSphere Web site: http://www.iseries.ibm.com/products/websphere/index.htm

<table>
<thead>
<tr>
<th>Component</th>
<th>Development time</th>
<th>Run time</th>
</tr>
</thead>
<tbody>
<tr>
<td>VisualAge for Java for AS/400</td>
<td>Windows 95, 98, NT, or 2000; OS/400 V4R4 or later</td>
<td>OS/400 V4R4 or later; JRE V1.2.2 or later; Netscape 4.7 (provided) or later or Internet Explorer 5.0 or later to access HTML-based help and Web documentation; TCP/IP communication protocol configured and running</td>
</tr>
</tbody>
</table>
2.2.1 Installing and configuring the default instance of the WAS

The following steps explain how to set up the WebSphere Application Server environment. This part of the scenario covers installing WebSphere Application Server and the WebSphere Administrative Console. You also configure the Web server instance for WebSphere Application Server.

2.2.1.1 Installing WebSphere Application Server on the iSeries server

IBM WebSphere Application Server support is provided by installing the 5733-WA3 product on your system. You can install WebSphere Application Server in two different ways:

- Directly to your iSeries server from the CD-ROM drive of your iSeries server
- Remotely to your iSeries server from the CD-ROM drive of a Windows 32-bit operating system workstation

For the purposes of this redpaper, we installed WebSphere Application Server directly from the CD-ROM drive of our iSeries server.

You can install WebSphere Application Server directly on an iSeries server by using the SETUP script in Qshell Interpreter or the Run Java (RUNJVA) command. If you are more familiar with UNIX programming, use Qshell Interpreter. If you prefer iSeries commands, use the RUNJVA command on the OS/400 command line.

Note

To run either of these commands, you need a user profile with *ALLOBJ authority.

To install WebSphere Application Server from the CD-ROM drive of your iSeries server, follow these steps:

2. We used all of the default settings for WebSphere Application Server. If you want to use different settings, refer to the WebSphere for AS/400 Documentation Center. The online version can be accessed at the AS/400 WebSphere Web site:

Follow either the Qshell Interpreter steps or the OS/400 command line steps if you want to use all of the default settings:

- **Using the Qshell Interpreter:**
  1. To start the Qshell Interpreter, on the OS/400 command line type:
     
     STRQSH
  2. Change directories to the root directory on the CD-ROM drive, and type:
     
     cd /QOPT/WebSphere
  3. Type SETUP
12 IBM WebSphere Development tools for AS/400: An Introduction

---

**Note**

Do not issue any other commands until the installation is complete. Qshell Interpreter runs interactively. Entering commands may cause the installation to stop prematurely.

---

**OS/400 command line:**
Enter the following command, exactly as shown, as one continuous line:

```plaintext
RUNJVA CLASS(SETUP) CLASSPATH('/QOPT/WebSphere/OS400: /QOPT/WebSphere/OS400/INSTALL.JAR:/QOPT/WebSphere')
PROP(os400.runtime.exec QSHELL)
```

---

**Note**

Although this command is displayed on more than one line, you must *enter it as one continuous line* on the OS/400 command line. Be sure to use the same capitalization as shown.

---

After you complete this step, messages appear that indicate what the installation process is currently doing. Installing WebSphere Application Server may take between thirty minutes and one hour.

Before you continue with the configuration of the WebSphere Application Server, you *must* install the latest PTFs for WebSphere Application Server Standard Edition for AS/400.

For the latest WebSphere PTF information, go to the Web site at:


Click the PTFs link.

---

### 2.2.1.2 Starting the WebSphere Application Server environment

You start the QEJBBSBS subsystem (and therefore, the WebSphere environment) by issuing the following command:

```plaintext
STRSBS SBSD(QEJB/QEJBBSBS)
```

Two jobs constitute the administrative server environment. The monitor job has the default name QEJBMNTR. An administrative server job has the default name QEJBADMIN.

Before you start your administrative console, you should ensure that the environment has started successfully.

When the WebSphere Application Server environment is ready for use, a message is written to the job log of the administrative server job, QEJBADMIN. The message indicates that the WebSphere administrative server is ready.

To determine if the WebSphere environment is ready, perform the following steps from an OS/400 command line:

1. Run the Work with Active Jobs (`WRKACTJOB`) command. Specify the QEJBBSBS subsystem on the subsystem (SBS) parameter:

```plaintext
WRKACTJOB SBS(QEJBBSBS)
```
2. Find your administrative server job. The default administrative server job is named QEJBADMIN.

3. Specify option 5 (Work with Job) on the option line next to the job, and press Enter.

4. On the command line of the Work with Job display, specify option 10 (Display joblog, if active), and press Enter.

5. Look for the following message:
   
   WebSphere administration server QEJBADMIN ready.
   
   See Figure 4.

   ![Display Job Log](image)

   Example: DisplayJobLog
   
   System: TORAS48F
   
   Job . . . : QEJBADM In User . . . : QEJB Number . . . : 437440
   
   WebSphere administration server QEJBADMIN ready.
   
   Job 438766/QUSER/QSQSRVR used for SQL server mode processing.
   
   Job 438766/QUSER/QSQSRVR used for SQL server mode processing.
   
   Job 438075/QUSER/QSQSRVR used for SQL server mode processing.
   
   Job 438374/QUSER/QSQSRVR used for SQL server mode processing.

   Press Enter to continue.

   F3=Exit    F5=Refresh    F10=Display detailed messages    F12=Cancel
   
   F16=Job menu    F24=More keys

   Figure 4. QEJBADMIN ready

6. Press F5 to refresh the joblog messages until the message appears.

   When the message appears, the WebSphere Application Server environment has successfully started. It may take several minutes for the message to appear, depending on your iSeries server.

7. To display the port number on which the administrative server is listening for the Administrative Console, position the cursor on the message and press F1. The following message appears:

   WebSphere administration server QEJBADMIN is ready to handle requests from the WebSphere administration console on port 900.

   See Figure 5 on page 14.

   **Note:** Port 900 is the default port.
2.2.1.3 Installing the WebSphere Administrative Console on your workstation

The WebSphere Administrative Console allows you to administer WebSphere Application Server for AS/400. The Console is a standalone Java application that runs on a workstation (Windows NT, AIX, or Solaris). It connects to the iSeries server on which administrative data is stored. For the purposes of this redpaper, we installed the WebSphere Administrative Console on a Windows NT workstation.

To install the Administrative Console for WebSphere Application Server on your workstation, perform the following steps:

1. Insert the WebSphere Application Server V.3.5, Standard Edition, for Windows NT CD-ROM in the CD-ROM drive of the Windows NT workstation on which you plan to run the console.

2. If Autorun is enabled, the Windows Install Shield program appears automatically. If Autorun is not enabled, run the Windows InstallShield program by using Windows Explorer to navigate to your CD-ROM drive. Double-click the setup.exe file.

3. Perform the steps of the WebSphere Application Server InstallShield program:
   a. You should only install the WebSphere Administrative Console portion of WebSphere Application Server on your workstation.
      
      **Note**: Ignore any messages that instruct you to shut down your Web server.
   b. Choose the custom installation option instead of the default installation.
   c. If you do not have the most current edition of the IBM Java Development Kit product supported by WebSphere Application Server installed, select this option, in addition to the Administrator's Console option. For the correct IBM Java Development Kit version for your workstation, refer to the
“Workstation Requirements” section of the WebSphere for AS/400 Documentation Center. You can access the online version at the AS/400 WebSphere Web site:

d. Deselect all other options.

e. When the program prompts you for database information, select the InstantDB option or enter false information with the other options. No databases are actually created on your workstation because you are only installing the Administrative Console and no other server components.

Before you proceed further, load the latest version of the WebSphere Application Server Standard Edition for AS/400 Group PTF on the administration console. For the latest WebSphere PTF information, go to the Web site at:
http://www.as400.ibm.com/websphere

Click the PTFs link.

2.2.1.4 Starting the WebSphere Administrative Console

To start the WebSphere Administrative Console to connect to your iSeries server, you must specify the OS/400 host name and optionally specify the bootstrap port that the administrative server is using on OS/400. To determine the host name, follow these steps:

1. Enter the Configure TCP/IP (CFGTCP) command on the OS/400 command line.
2. Select option 12 (Change TCP/IP Domain).

Note the Host name value. This value should be used as the Host name parameter. The Host name parameter is case sensitive. See Figure 6 on page 16.

---

Note
It is required that you have a host name entered on OS/400. The WebSphere Administrative Console will not connect if the entry is not present. If you do not have a host name entry, add it. Additionally, if the OS/400 host name is in lowercase, you must also enter the name in lowercase when connecting to the WebSphere Administrative Console.
The WebSphere Administrative Console uses port 900 by default. If you changed the default port with the admin.bootstrapPort parameter when you started the administrative server, you need to specify that port for the WebSphere Administrative Console. The admin.bootstrapPort parameter is specified in the admin.properties file.

2.2.1.5 Starting the Administrative Console on Windows NT

To start the WebSphere Administrative Console on a Windows NT workstation, you can use the Start menu shortcut that was created during installation on Windows NT, or you can run a batch file from a command prompt.

- From the Start menu shortcut:
  1. Change the shortcut to specify the host and optionally specify the port.
  2. Find the shortcut for the WebSphere Administrative Console. Use Windows NT Explorer to navigate to the folder where the shortcut profile is stored. By default, it should be in the following directory structure:
     
     C:\WINNT\Profiles\All Users\Start Menu\Programs\IBM WebSphere\Application ServerV3.5\Administrator's Console.link
  3. Right-click the Administration Client shortcut, and select the Properties view.
  4. Go to the Shortcut notebook page. Change the Target field to add a host and optionally add a port. For example, if TORAS48F is your host name, and you are using the default port number 900, you would change the target field from C:\WebSphere\AppServer\bin\adminclient.bat to:
     
     C:\WebSphere\AppServer\bin\adminclient.bat TORAS48F 900
  5. Start the WebSphere Administrative Console on Windows NT. To do so from the Start menu, select Programs->IBM WebSphere->Application Server V3.5->Administrator's Console.
• From the Windows DOS command prompt:

Open a command prompt, and enter the following two commands:

```
cd WebSphere\AppServer\bin
adminclient TORAS48F 900
```

Here, `WebSphere\AppServer\bin` is the directory where the WebSphere Administrative Console is installed. `TORAS48F` is the host name of your iSeries server, and `900` is the iSeries default port number that you intend to use. The port number should match what was specified for the `admin.bootstrapPort` parameter in the `admin.properties` file.

The amount of time that it takes for the WebSphere Administrative Console to start depends on the number of objects that you configured. It is not uncommon for this to take several minutes if you configured a large number of objects.

The Console Ready message is displayed in the message area in the bottom of the WebSphere Administrative Console window. See Figure 7.

![Figure 7. WebSphere Administrative Console: Console Ready](image)

### 2.2.2 Creating an HTTP server configuration and server instance

WebSphere Application Server plugs into IBM HTTP Server for AS/400. IBM HTTP Server routes certain client requests (such as for a servlet or JSP file) to WebSphere for processing. You must create a new HTTP server configuration (or modify an existing configuration) that contains the information that IBM HTTP Server needs to route requests to the appropriate WebSphere Application Server methods.

#### 2.2.2.1 Starting an *ADMIN instance of IBM HTTP Server for AS/400

Start an *ADMIN instance of IBM HTTP Server for AS/400 to create, change, or display an IBM HTTP server instance configuration by using the IBM HTTP Server for AS/400 Configuration and Administration forms.
You can start the *ADMIN server instance from Operations Navigator or from the OS/400 command line by following the process explained in this section.

- From Operations Navigator:
  
  Operations Navigator is the graphical point-and-click interface to the iSeries server. Operations Navigator is part of the AS/400 Client Access product. For more information on Client Access and Operations Navigator, see the Client Access product Web site at:
  

  2. Double-click your iSeries server in the main tree view of Operations Navigator.
  3. Double-click Network.
  5. Double-click TCP/IP.
  6. Right-click HTTP Administration in the right pane.
  7. Select Start from the pop-up menu.

- From the OS/400 command line:
  
  Start the *ADMIN instance from the OS/400 command line by typing:

  ```
  STARTCPSVR SERVER(*HTTP) HTTPSVR(*ADMIN)
  ```

  Press Enter.

2.2.2.2 Creating a new HTTP server configuration

To create a new IBM HTTP Server configuration, open the Configuration and Administration forms and follow these steps:

1. Start your JavaScript-enabled browser.

2. In the URL location or address window, type:


   Here, your.server.name is the host name of your iSeries server. In our example, we used: [http://TORAS48F:2001](http://TORAS48F:2001)

   Press Enter.

3. You are prompted for an OS/400 user ID and password. Your OS/400 user ID must have *ALLOBJ authority.

   The AS/400 Tasks page appears. See Figure 8.
4. Click **IBM HTTP Server for AS/400**.

The IBM HTTP Server for AS/400 page appears. See Figure 9.
5. In the left pane, click **Configuration and Administration**.

   The IBM HTTP Server Configuration and Administration page appears. See Figure 10.

   ![IBM HTTP Server Configuration and Administration](image)

   **Figure 10. IBM HTTP Server Configuration and Administration**

6. Click the **Configurations** link in the left frame to expand it.

7. Click the **Create configuration** link in the left frame. The Create Configuration page is shown in the right frame.

8. Provide a unique name for your configuration, and select whether you want to create an empty configuration or copy an existing configuration if appropriate.

9. Click the **Apply** button at the bottom of the page. The message **The configuration file was successfully created** should be displayed. See Figure 11.
2.2.2.3 Creating an HTTP server instance

You need to create an HTTP server instance to implement your HTTP server configuration settings as an OS/400 job. This topic discusses creating an HTTP server instance. Customizing your HTTP server configuration to work with WebSphere Application Server for AS/400 and starting your HTTP server instance are discussed in 2.2.2.4, "Editing the IBM HTTP Server for AS/400 configuration file" on page 22, and 2.2.2.5, "Starting the IBM HTTP Server for AS/400 instance you created" on page 24.

To create an HTTP server instance, follow these steps:

1. Click the Server Instances link in the left frame to expand the section.
2. Click the Create server instance link in the left frame.
   
   The Create server instance page is displayed in the right frame.
3. Specify a unique name for the Server instance field.
   
   Note: We recommend that you give the server instance the same name as the configuration that it implements to simplify administration.
4. Select the configuration that you created in 2.2.2.2, "Creating a new HTTP server configuration" on page 18, from the drop-down list in the Configuration field.

Figure 11. Create configuration page
5. Click the **Create** button at the bottom of the page. A message stating “The server instance was successfully created” is displayed. See Figure 12.

![Create server instance page](image)

**Figure 12. Create server instance page**

### 2.2.2.4 Editing the IBM HTTP Server for AS/400 configuration file

You need to edit the configuration file to add the directives NameTrans, Authorization, Service, ServerInit, ServerTerm, and Pass. These directives allow a Web browser to access the Servlet Engine and the Web applications that come with WebSphere Application Server.

To edit the configuration file to add directives, follow these steps:

1. Start your JavaScript-enabled browser.
2. In the URL location or address window, type:
   ```plaintext
   http://your.server.name:2001
   ```
   Here, **your.server.name** is the host name of your iSeries server. In our example, we used: http://TORAS48F:2001
   Press Enter.
3. You are prompted for an AS/400 user ID and password. Your AS/400 user ID must have *ALLOBJ authority. The AS/400 Tasks page appears.
5. In the left pane, click **Configuration and Administration**. The IBM HTTP Server Configuration and Administration page appears.

6. Click the **Configurations** link in the left-hand frame to expand it.

7. Select your HTTP server configuration from the pull-down menu at the top of the left-hand frame.

8. For the purposes of this exercise, we used the default port (80). If you want to use a port other than the default port (80), or if you want to change your existing port number, click **Basic** in the navigation menu. Specify the desired port number in the Default port field, and click **Apply**.

9. Update the configuration with the WebSphere Application Server directives. Perform the following steps:
   a. In the left navigation frame, click **Java servlets**.
   b. In the main content frame, select **WebSphere version 3**, and click **Servlets and JavaServer Pages (JSP)**.
   c. In the main content frame, select your WebSphere instance name from the WebSphere Domain list box.
   d. Click **Apply**. See Figure 13.

![Figure 13. Editing the HTTP server configuration](image-url)
All necessary NameTrans, Authorization, Service, ServerInit, ServerTerm, and Pass directives are automatically added to your server instance configuration. Directives used to run previous releases of WebSphere Application Server are also removed.

**Note**

The directives are added as the first entries in the HTTP configuration file. If you have existing directives you want processed before the WebSphere directives, you must manually rearrange or add them.

### 2.2.2.5 Starting the IBM HTTP Server for AS/400 instance you created

IBM HTTP Server for AS/400 runs in the QHTTPSVR subsystem, and each HTTP server instance starts multiple jobs. The WebSphere Application Server code plugs into IBM HTTP Server for AS/400 and runs in the HTTP server job that communicates with the administrative server and one or more application servers.

Start the HTTP server instance that you created by typing the following command on the OS/400 command line:

```
STARTCPYSVR SERVER(*HTTP) HTTPSVR(my_instance)
```

Here, `my_instance` is the name of your HTTP server instance.

If you change your HTTP server instance configuration, stop and then start your HTTP server instance. To stop your HTTP server instance, enter the following command from the OS/400 command line:

```
ENDTCPYSVR SERVER(*HTTP) HTTPSVR(my_instance)
```

Again, `my_instance` is the name of your HTTP server instance.

**Note**: You can also start and stop your HTTP server instance from the IBM HTTP Server Configuration and Administration forms. See Figure 14.

The Configuration and Administration forms also provide the option to restart your HTTP server instance. When you restart the server instance, the HTTP server recognizes all configuration changes except for the changes to the Basic and Security configuration forms.
You can verify the installation of WebSphere Application Server by running the HelloWorldServlet servlet. The successful execution of the HelloWorldServlet verifies that your default application server is working correctly. The HelloWorldServlet servlet is automatically created and configured during the initial setup of the administrative server. You can find the HelloWorldServlet in the default_app Web application under the default Servlet Engine of the Default Server once the WebSphere Administrative Console has started.

To verify your installation, start the WebSphere Administrative Console on your workstation. We used the default port (80) in our example for this redpaper.

If your HTTP server configuration uses a port other than the default (port 80), you must update the Host Aliases table under the virtual host, which is default_host, to reflect the correct HTTP port number. This is achieved by following these steps:

1. From the WebSphere Administrative Console topology view, locate the virtual host, which is named default_host.
2. Click default_host. The settings for default_host appear in the right side of the console.
3. Go to the Advanced settings by clicking the Advanced tab.
4. In the Host Aliases menu, update the specified host aliases with the correct port number. For example, the host name TORAS48F would become TORAS48F:12345. Here, 12345 is the port number you used for your HTTP server instance.
5. Click Apply. See Figure 15 on page 26.
6. In the Topology view of the WebSphere Administrative Console, find the node that has the same name as the host name of the iSeries server, for example TORAS48F. Expand that node and right-click the Default Server application server instance.

7. Start the application server.

You can use the Administrative Console to start and stop application servers. If you attempt to stop an application server and it does not stop, you can stop it by using OS/400 commands. When you start or stop an application server, you also start or stop everything that runs within the application server, such as servlet engines.

**Starting the application server**

To start an application server instance, follow these steps using the WebSphere Administrative Console:

1. In the topology tree, locate the application server that you want to start.
2. Right-click your selected application server.
3. In the resulting menu, click **Start**. It may take a few seconds for this operation to complete. A dialog box appears to announce the successful start of the application server.

**Stopping the application server**

To stop an application server instance, follow these steps using the WebSphere Administrative Console:

1. In the topology tree, locate the application server that you want to stop.
2. Right-click the application server.
3. On the resulting menu, click **Stop**. It may take a few seconds for this operation to complete. A dialog box appears announcing the successful stop of the application server.

   If the stop operation fails, you can use the force stop option instead. Click **Force stop** instead of Stop.

   **Note**: Using the force stop option ends objects abnormally and does not clean up WebSphere resources as a normal stop operation does.

### 2.2.2.6 Verifying the IBM HTTP Server for AS/400 instance

The first time you start WebSphere Application Server, you must start the application server before you start your HTTP server instance. The first time the application server starts a initialization is performed that affects your HTTP server instance. If the instance is already running, restart it after you start the application server to refresh the configuration.

Open a browser and go to the URL:

http://your.server.name:port/servlet/hello

**Note**: The URL is case sensitive. Therefore, the capitalization must be consistent with the above example.

In the URL, `your.server.name` is the name of your iSeries server, and `port` is the port number of your HTTP server instance.

If you see “Hello World” as the result, your WebSphere Application Server has been set up successfully. See Figure 16 on page 28.
2.3 Setting up your instance of the WebSphere administrative server

This section is targeted for users with previous experience setting up the default administrative server and who now want to create additional instances of the WebSphere administrative server. For users with little or no previous experience in this area, we recommend that you refer to 2.2, “Setting up the default instance of the WebSphere Application Server” on page 10.

A single administrative server allows you to run many application servers. Each application server runs in its own process. In most cases, a single administrative server handles your scalability and isolation needs. Additionally, a single administrative server allows you to use a single Administrative Console to manage all the server resources.

However, since all application servers in an administrative domain share the same Persistent Name Server name space, you may want to create multiple administrative servers on a single iSeries server for any of the following reasons:

- To create separate development environments for different developers. This allows them to have different versions of the same objects in their own name space.
- To create separate development and test environments.

All references to WebSphere without other designation should be interpreted as references to IBM WebSphere Application Server for AS/400, Standard Edition, Version 3.5.
2.3.1 Creating additional administrative servers

To create additional instances of the WebSphere Application Server product, you must create multiple administrative servers.

To create a new administrative server, run the script that creates all new server directories and sets up the correct authorities. Your OS/400 user profile must have *ALLOBJ authority to run the script. Follow these steps:

1. On an OS/400 command line, run the STRQSH (Start Qshell Interpreter) command.
2. Set up Qshell Interpreter to run the WebSphere Application Server scripts.

All WebSphere Application Server scripts are located in the /QIBM/ProdData/WebASAdv/bin directory and must be run from Qshell Interpreter.

Use one of the following methods to run the scripts in Qshell Interpreter. Use the cd command to change to the /QIBM/ProdData/WebASAdv/bin directory, and then run the script. In our case, we typed the following option on the command line:

cd /QIBM/ProdData/WebASAdv/bin
crtnewinst -instance itso2 -bootstrap 790 -lsd 791

Here, crtnewinst is the name of the script, and -instance, -bootstrap, and -lsd are the parameters that are passed to the script.

The other methods are included here for completeness:

- Invoke the fully qualified path name of the script.
  An example is: /QIBM/ProdData/WebASAdv/bin/script_name parameters
  Here, script_name is the name of the script, and parameters indicate the parameters that are passed to the script.

- Update your PATH environment variable to automatically locate the script when you run it. Follow these steps:
  i. Edit the .profile file in the /home/user_profile_name directory. Here, user_profile_name is the name of your iSeries user profile.

Note

If this file does not exist, create it in this directory. Save the file in an ASCII code page format. Also note that .profile is the full name of the file. When Qshell Interpreter is started, it searches for the .profile file, and executes the commands listed in it. You can use the .profile file to set persistent environment variables for your Qshell Interpreter session.

ii. Add the following line to the .profile file:

   export PATH=/QIBM/ProdData/WebASAdv/bin:$PATH

iii. Save the file.

To run the scripts, follow these steps:

a. From an OS/400 command line, run the STRQSH (Start Qshell Interpreter) command.
b. Invoke the script by name, for example:

    script_name parameters

Here, `script_name` is the name of the script, and parameters are the parameters that are passed to the script.

c. Run the script.

`Crtnewinst` is the script for creating new WebSphere administrative server instances. The syntax for the crtnewinst script is shown here:

```
```

The parameters are shown in Table 3.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-instance</td>
<td>The required value <code>&lt;instance_name&gt;</code> specifies the name of the instance. The script creates the new administrative server instance in the <code>/QIBM/UserData/WebASAdv/instance_name</code> directory.</td>
</tr>
<tr>
<td>-bootstrap</td>
<td>The required value <code>&lt;bootstrap_port&gt;</code> specifies the number of the TCP/IP port from which the client (such as the Administrative Console) connects to the administrative server instance. Specify an unused port number on your iSeries server. Port 900 is used by the default administrative server instance and should not be used for other instances. Use the Work with TCP/IP Network Status (<code>NETSTAT *CNN</code>) command to display a list of port numbers that are currently being used.</td>
</tr>
<tr>
<td>-lsd</td>
<td>The required value <code>&lt;lsd_port&gt;</code> specifies the number of the TCP/IP port on which the Location Service Daemon (LSD) service listens. Specify an unused port number on your iSeries server. Port 9000 is used by the default administrative server instance and should not be used for other instances. Use the Work with TCP/IP Network Status (<code>NETSTAT *CNN</code>) command to display a list of port numbers that are currently being used.</td>
</tr>
</tbody>
</table>

The remaining parameters are optional and were not used for the purposes of this redpaper. For a description of these parameters, refer to the Web site at: http://www.iseries.ibm.com/products/websphere/index.htm

2.3.2 Starting additional administrative servers

To start an additional administrative server, perform the following steps:

1. On the OS/400 command line, run the Start Qshell Interpreter (`STRQSH`) command.

2. Set up Qshell Interpreter to run WebSphere Application Server scripts as explained in Step 2 in 2.3.1, “Creating additional administrative servers” on page 29.

3. To start the WebSphere administrative server instance, run the `strwasinst` script. For this redpaper, we ran the following script:

```
strwasinst -instance itso2
```

The syntax for the strwasinst script is shown here:

```
strwasinst -instance <instance_name> [-http <web_server_instance>] [-job <job_name>] [-help]
```
The parameters are shown in Table 4.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-instance</td>
<td>The required value &lt;instance_name&gt; specifies the name of the instance.</td>
</tr>
<tr>
<td>-http</td>
<td>The optional value &lt;web_server_instance&gt; specifies the name of the Web server instance that you configured in 2.2.2.3, “Creating an HTTP server instance” on page 21. The script starts the instance for you.</td>
</tr>
<tr>
<td>-job</td>
<td>The optional value &lt;job_name&gt; specifies the name of the monitoring job. If you do not specify this parameter, the default value is instance_nameMNTR, where instance_name is the name of your instance.</td>
</tr>
<tr>
<td>-help</td>
<td>This optional value displays the help message.</td>
</tr>
</tbody>
</table>

Alternately, you could run the following command (as a single command) on an OS/400 command line:

```
SBMJOB CMD(CALL PGM(QEJB/QEJBMNTR) PARM('-p' 'myDirectory/properties/admin.properties')) JOB(my_mntr) JOBD(QEJB/QEJBJOBD) JOBQ(QEJB/QEJBJOBQ) USER(QEJB)
```

Here, myDirectory is the fully qualified path name of the root directory on which the additional administrative server core directories reside (for example, /QIBM/UserData/WebASAdv/myAdmin). my_mntr is the job name you want your monitor to appear as within the QEJBBSBS subsystem.

4. Verify that the ITSO2ADMIN job is ready for use. From an OS/400 command line, follow these steps:

a. Enter the command:

```
WRKACTJOB SBS(QEJBBSBS)
```

b. Find the ITSO2ADMN administrative server job.

c. Specify option 5 (Work with job) and then specify option 10 (Display joblog).

d. Press F5 to refresh the joblog messages until the following message appears (Figure 17 on page 32):

```
WebSphere administration server itso2ADMN ready.
```

e. Position the cursor on the message and press F1. See Figure 18 on page 32.
2.3.3 New HTTP server configuration: Starting an HTTP server instance

To create a new HTTP server configuration and start an HTTP server instance, follow these steps. Each step has been covered earlier in this redpaper:

1. Start the HTTP Server Configuration and Administration forms by typing in the URL location or address window of your browser:

   http://your.server.name:2001

---

Display Job Log

System: TORA48F

Job . . . : ITSO2ADMN User . . . : QEJB Number . . . : 442565

Job 442565/QEJB/ITSO2ADMN started on 00/12/08 at 09:50:50 in subsystem QEJBSBS in QEJB. Job entered system on 00/12/08 at 09:50:50.
Output file RDBENTRIES created in library QTEMP.
Member RDBENTRIES added to output file RDBENTRIES in library QTEMP.
WebSphere server started with JDK 1.2.
Job 442526/QUSER/QSQSRVR used for SQL server mode processing.
Job 442596/QUSER/QSQSRVR used for SQL server mode processing.
Job 442595/QUSER/QSQSRVR used for SQL server mode processing.
Job 442601/QUSER/QSQSRVR used for SQL server mode processing.
Job 442648/QUSER/QSQSRVR used for SQL server mode processing.
WebSphere administration server itso2ADMN ready.

Press Enter to continue.

F3=Exit  F5=Refresh  F10=Display detailed messages  F12=Cancel
F16=Job menu  F24=More keys

Figure 17. ITSO2ADMN ready

Additional Message Information

Message ID . . . . . . . : EJB0106  Severity . . . . . . . : 00  
Message type . . . . . . : Information  Date sent . . . . . . . : 00/12/08  Time sent . . . . . . . : 09:53:29

Message . . . . : WebSphere administration server itso2ADMN ready.
Cause . . . . . : WebSphere administration server itso2ADMN is ready to handle requests from the WebSphere administration console on port 790. Diagnostic information is controlled on port 5154.

Press Enter to continue.

F3=Exit  F6=Print  F9=Display message details  F12=Cancel  
F21=Select assistance level

Figure 18. ITSO2ADMN ready to handle requests

---

IBM WebSphere Development tools for AS/400: An Introduction
Here, your.server.name is the host name of your server. For the purposes of this exercise, we used: http://TORAS48F:2001

2. Create a new HTTP server configuration. We followed these steps for our configuration:
   a. Create a new HTTP server configuration called ITSO2 as shown in Figure 19.

   ![Figure 19. Creating an HTTP configuration (Part 1 of 4)](image)

   b. Add the Basic details. Set the Host name to TORAS48F and set the Default port to 792 as shown in Figure 20 on page 34.
c. From the Servlets page, select **WebSphere version 3**, and click the **Servlets and JavaServer Pages** box as shown in Figure 21.
d. Select **Request Processing->Methods**, and click **POST**. See Figure 22 on page 36.

**Note:** Be sure to click **Apply** after each action.
3. Create an HTTP server instance. We created an HTTP server instance called ITSO2 with configuration ITSO2. See Figure 23.
4. Follow this process to edit the configuration file:
   a. Start the Administrative Console. From a DOS prompt, we entered:

   ```
   cd websphere\appserver\bin
   adminclient TORAS48F 790
   ```

   You should enter your own host name and selected port number here. See Figure 24.

   Figure 24. Starting the Adminclient

   b. Highlight default_host.
c. Click the **Advanced** tab.

d. Update the **HostAliases** table with the correct port number.

Change every entry in the table by appending :port_number to the entry. For example, we changed `localhost` to `localhost:792`. See Figure 25.

![Editing the configuration file](image)

**Figure 25. Editing the configuration file**

5. Follow this process to start the HTTP server instance:

   a. Go to the HTTP Server Configuration and Administration form.

   b. Select **Work with server instances**.

   c. Select your server instance from the drop-down list. We selected ITSO2.

   d. Click **START**. See Figure 26.
IBM HTTP Server for AS/400 runs in the QHTTPSVR subsystem, and each HTTP server instance starts multiple jobs. The WebSphere Application Server code that plugs into IBM HTTP Server for AS/400 runs in the HTTP server job that communicates with the administrative server and one or more application servers.

The HTTP server instance can also be started from an OS/400 command line by typing:

```
STRTCPSVR SERVER(*HTTP) HTTPSVR(my_instance)
```

Here, `my_instance` is the name of your HTTP server instance.

If you change your HTTP server instance configuration, stop and then start your HTTP server instance. To stop your HTTP server instance, enter the following command from the OS/400 command line:

```
ENDTCPSVR SERVER(*HTTP) HTTPSVR(my_instance)
```

Here, `my_instance` is the name of your HTTP server instance.

The Configuration and Administration forms also provide the option of restarting your HTTP server instance. When you restart the server instance, the HTTP server recognizes all configuration changes, except for changes to the Basic and Security configuration forms.

6. Verifying the installation.

You can verify the installation by running the HelloWorldServlet. Successful execution of the HelloWorldServlet verifies that your application server is working correctly.

Open a browser, and go to the following URL:

```
http://your.server.name:port/servlet/hello
```
**Note:** The URL is case sensitive. Be sure to use the same capitalization as shown in the example.

In the URL, `your.server.name` is the name of your iSeries server, and `port` is the port number of your HTTP server instance.

In the example we created for this redpaper, we typed:

```
http://TORAS48F:792/servlet/hello
```

If you see the “Hello World” as the result, then your WebSphere Application Server has been set up successfully. See Figure 27.

---

**Figure 27. Hello World**
Chapter 3. Building new e-business applications

IBM WebSphere Studio for AS/400 is an extension of IBM WebSphere Studio and is part of the IBM WebSphere Development Tools for AS/400. WebSphere Studio is the tool of choice for creating and managing Web sites. WebSphere Studio is a competitive Web development tool as demonstrated by its “Best of Show” award at Internet World in October 1999. It combines graphical development wizards with tools for Web site design and Java development.

IBM WebSphere Studio for AS/400 is designed to allow OS/400 users to quickly enable new applications on the Web. It does this by extending IBM WebSphere Studio with wizards that provide the AS/400 interface and allows the user to take advantage of their current iSeries or AS/400 skill set.

The wizards allow the user to define the data and program interfaces between the Web pages that they design and AS/400 ILE programs. WebSphere Studio for AS/400 generates the JSPs, servlets, and beans for handling the interface between the Web pages and the iSeries or AS/400 programs written by the user.

3.1 WebSphere Studio for AS/400 extensions

The ability to implement iSeries or AS/400 applications on the Web in WebSphere Studio for AS/400 is provided in three distinct areas:

- **Design Time Controls**
  Design Time Controls (DTC) has been added to the page designer in WebSphere Studio for AS/400. It gives the user the ability to define AS/400 data fields, push buttons, radio buttons, combo boxes, lists, and subfiles. These special features are used to inform WebSphere Studio for AS/400 that these items are to be used to communicate from the Web page to an iSeries or AS/400 ILE program.

- **Publishing SetUp Wizard**
  The Publishing setup wizard is used to define the OS/400 environment and WebSphere Application Server environment for the application. This includes the system name, user, directory paths, server instance to be used, and the libraries that are required.

- **Web Interaction Wizard**
  This wizard is used to create the interaction between the Web pages and the ILE programs on the iSeries and AS/400 servers. It is used to define input, output, error Web pages, and to identify the ILE program to call for a particular interaction. It also allows the user to identify the parameters that are passed to the ILE program and to map the input and output fields from the pages to the ILE program parameters.

3.2 Basic application architecture

WebSphere Studio for AS/400 uses the Web application model, which is outlined in Figure 28 on page 42.
The application model consists of one or more “interactions”. Each interaction is defined as one input Web page that provides input to the ILE program, and one output browser page to receive output from the ILE program and an associated error browser page to handle exceptions. The input page communicates with the WebSphere Application Server where the servlets, JSPs, and beans, created by WebSphere Studio for AS/400, call the user supplied ILE program. The output from the program is supplied, via parameters, back to the servlets and JSPs which, in turn, present the output page to the browser user.

There are cases where the application logic requires that one of several different pages be used for output. This changes the normal flow of input and output pages as defined by WebSphere Studio. These exceptions can be handled by a feature called flow control, which can be accessed by the Web Interaction Wizard. While this gives the developer some control over the application, you should remember that this is a browser-based application rather than a 5250 application. It provides different functions than a 5250 application, which has the ability to output multiple formats to one screen.

Once a browser user makes a request with a Web input page, that request causes a Java application code to be executed in the WAS, which calls an ILE program. This causes a job to be started on the iSeries server, and the requested ILE program loads and executes. The input fields from the Web input page are passed to the ILE program via parameters. The ILE program then executes and passes the results back to the Java application code running in the WAS. The WAS application code, servlets, and JSP map the ILE program parameters to the output fields for the results or output page and send the output page to the browser user.

Code your ILE program the way you normally would, except that you do not code DDS screens or create display files. This aspect of your application is now handled by the Web interface you created.

Note that the parameters you code in your program need to be in the same order as the parameter definitions you created in the Web Interaction wizard. Of primary importance is the business logic you need to code to correctly run the program calls submitted through the Web interface. For a more detailed example of a Web application, see “Creating Web Applications” in the Getting Started with
Chapter 3. Building new e-business applications

WebSphere Development Tools for AS/400 online information. This information also refers to samples that are packaged in the product that you can review and use.

Also take into account the requirements for running a cleanup program if your Web application times out, or if the user inadvertently ends the browser session (the default session time out is 30 minutes, but it can be changed in the Session Manager of WAS). If this occurs, a call is automatically made to the cleanup program @AFFCLNUP. You need to create this program to invoke any program cleanup routines that close open files or restore database records. This program must not contain any input or output parameters. You also need to ensure that it is in a library that you specified in the library list in the Publishing Setup wizard. If the program does not exist, or is not located in the library list, your program cleanup will not occur even though the browser session ends.

Some control over the flow of the screen within the application is provided by WebSphere Studio for AS/400. The developer can ask to use flow control with the Web Interaction Wizard. One of the parameters is designated for the flow control. This ILE program dictates which page is presented next and moves the name of the JSP that controls a page's output into the flow control parameter. WebSphere Studio for AS/400 also provides:

- The ability to use AS/400 message files
- Code to do input field editing, such as restricting numeric input fields to numeric only
- The ability to access UDB DB2 for AS/400 table definitions and synchronize the definition in the application code with the database so that changes made to the database are reflected in the application.

### 3.3 Sample application

To illustrate the abilities of WebSphere Studio for AS/400, we outlined the steps required for implementing one part of an application. In this example, we design the first stage of the Parts Order Entry application for the ABC Company. This sample consists of the first Web input page that requires the input of a customer number, and the output page displays the customer number, the next available order number, the customer name, and address information. This a simple application. However, it provides a good overview of the features and functions of WebSphere Studio for AS/400 and the steps required to build an application. The application was developed with IBM WebSphere Studio for AS/400 using IBM WebSphere Application Server Advanced Edition, Version 3.5, and OS/400 Version 4 release 5.

The sample application covers the AS/400 features in WebSphere Studio for AS/400. This sample assumes that you are familiar with the other features of IBM WebSphere Studio. If you require more information, refer to Web Enabling AS/400 Applications with IBM WebSphere Studio, SG24-5634.

Remember that there are several steps involved in developing an application using WebSphere Studio for AS/400. To build the application, perform these tasks:
- Design a page and define AS/400 input using Design Time Controls.
- Set up and publish the pages to the iSeries server.
- Define how the Web pages interact with the ILE programs via the Web Interaction Wizard.
- Publish the results of the interaction to the iSeries system.

We have an ILE RPG program called ORDENTR (the sample code can be viewed in Appendix B, “Sample code” on page 93), and we are using the ITSO2 instance of the WAS.

3.3.1 Design and create your Web interface

This stage involves designing Web pages and defining their input and output fields. The design portion uses the standard IBM WebSphere Studio features, and the AS/400 interface uses the Design Time Controls feature.

To begin the process, follow these steps:

1. Start WebSphere Studio for AS/400.
2. Select File->New Project.
3. Assign a project name. In our case, we chose ABCDemo. Remember that this project name becomes the name of the Web application server that runs on our instance of WebSphere Application Server. See Figure 29.

![Figure 29. Naming the project2](image)

4. Select OK. The window displaying the project appears as shown in Figure 30.
5. Right-click **ABCDemo**. Select **Insert->File**.

6. The Insert File Dialog appears as shown in Figure 31. You want to create a JSP for your input page. In our case, we called it custno.jsp.

7. The custno.jsp now appears in the Theme folder. Expand the Theme folder and double-click the **custno.jsp** to open the design widow for custno as shown in Figure 32 on page 46.
8. Some images and a simple logo are added to this page. Since you are interested in the iSeries aspect, skip directly to that stage. As shown in Figure 33, you are ready to begin defining the fields.

9. Position the cursor below the Parts Order Entry.

10. Select Insert->**Form and Input Fields**->**Form**. Then, a form appears on the page as shown in Figure 34.
11. Inside the form, type:

   Enter Customer Number

12. With the cursor positioned after the data that was just entered, select **Insert->Design-time Control** as shown in Figure 35.

   ![Design-time Control](image)

   Figure 35. Design-time Control

   The Design Time Controls dialog box appears as shown in Figure 36 on page 48.
13. The Design Time Control box lists the types of OS/400 objects that you can use to communicate to the ILE program. Note that these items must appear inside a form, such as the one you just generated. Otherwise, the information associated with items is not sent to the iSeries server. The capabilities that are listed include combination boxes and check boxes. Two entries that are of immediate interest to you are the AS/400 Subfile DTC, which is used to define subfiles, and the AS/400 Entryfield DTC.

Select the **AS/400 Entryfield DTC** to define the input entry fields for the Web page. The outline of the entry field appears within the form (Figure 37).

14. Right-click the data entry box. Select **Control Properties** from the pop-up menu. The AS/400 Entryfield DTC Properties box appears (Figure 38).
The Properties dialog allows you to define the Web page field characteristics. In our case, we named the field “custno” and defined it as a 4-byte character field. Note that you are defining the display characteristics of the field, not whether the field is to be used for input or output. That is done when you define the Web interaction.

There is also an opportunity to define how the field should be edited and if it should be a hidden field. There is an additional option for importing the definitions from the DB2 UDB for AS/400 database. If you click the DB Reference box, you are taken through a series of displays and prompts that allows you to connect to an iSeries server and retrieve the characteristics of a data field.

Using the Events tab, you can also define that certain actions on this field cause an associated Java script to execute. The developer must supply the desired Java Script.

15. Select OK. You have now defined how your Web page will look and the data fields that you want on the Web page. You must define the action that can be taken.

16. Place the cursor after the input box that was just defined and press Enter. The cursor is now positioned under the “Enter Customer Number” text. You now define the Submit button that causes the Web page to be submitted.

17. Select Insert->Design-time Control, and the Design-time Control dialog box appears. Select AS/400 Pushbutton DTC as illustrated in Figure 39 on page 50.
18. Right-click the **PushButton400** box in Figure 40, and select **Control properties** from the pop-up menu that appears.

19. This displays the AS/400 Pushbutton DTC Properties dialog, which is shown in Figure 41.
In this dialog, we assigned a field name of getButton. The contents of the Label box are the text that appears on the push button. The Value box is used to pass data to the ILE program if more than one button is defined. The ILE program uses this to determine which button caused the action to be taken. You can also define the button type. In our example, we use a submit button.

Click OK.

20. You are now finished designing your page and the data fields that you require. In the Design window, click the HTML tab to review the HTML that has been generated, or click the Preview tab to review how the page will look.

21. Select File->Save to save the page.

Your page is now designed and saved, and you are ready to proceed to the next step: publishing your results.

3.3.2 Using the Publishing Setup Wizard

Your page is now ready to be published. You do this using the Publishing Setup Wizard. You define the WebSphere Application Server environment that you want to use, such as the instance, where you want to store your HTML, servlets, and JSPs.

1. Highlight the project name in the left panel, and click View->Publishing.

The publishing window should now appear. Your page should appear similar to the example in Figure 42. Select the project ABCDemo.

![Figure 42. WebSphere Studio for AS/400 (Publishing)](image)

2. Select Tools->Wizards->Publishing Setup Wizard as shown in Figure 43 on page 52.
3. The Published Setup Wizard dialog shown in Figure 44 appears. This wizard allows you to define the WebSphere Application environment.

![Publishing Setup Wizard](image)

Figure 44. Publishing Setup Wizard

The information specified in the Publishing Setup wizard is applied to the WebSphere Application Server configuration as shown in Figure 45.
This dialog is rather important, so it is reviewed item by item in Table 5.

**Table 5. Publishing parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Host Name</td>
<td>This is the name of the WebSphere Application node where your instance of the WAS is running. The name must be described in full; the complete IP address is shown.</td>
</tr>
<tr>
<td>XML node name (A in Figure 45)</td>
<td>This is the name of the XML node and is used by WebSphere Studio for AS/400 to configure the WebSphere Application server for you. You see the results of this in the next step when it creates the application on the WebSphere Application Server.</td>
</tr>
<tr>
<td>WAS bootstrap port</td>
<td>This is the port number that is used to create your instance of the WAS. In your case, you created an instance called ITSO2 using bootstrap port 790. The default port is 900. You created your own instance of the WAS to provide a unique testing environment.</td>
</tr>
<tr>
<td>Web application name (B in Figure 45)</td>
<td>This is the name of our application. Notice that it is filled in automatically and has the same name as the project that you have been working on in WebSphere Studio for AS/400. In your case, it is called ABCDemo.</td>
</tr>
</tbody>
</table>
4. It is not necessary to change any of the directory entries, unless you desire to separate your object for your instance of WAS. In this case, you would enter your instance name here in place of our itso2. You may also use your own directory structure.

5. Click the **Validate** button. This checks to see if the directories exist on the IFS. If they do not, it creates them. You may be asked for an AS/400 user ID and password.
6. Click Next. The AS/400 environment display shown in Figure 46 appears.

![AS/400 environment settings](image)

**Figure 46. AS/400 environment**

Enter the name of the host system used and the user ID and password for the user who executes the ILE programs on the iSeries server. It is important that this user has the appropriate authority to run the application.

Enter the library name that the application uses. In our case, it was APILIB. Remember, if the library is specified here, it is not necessary to specify the library in the user library list. It does not cause problems, but it logs extra error messages as the application is running.

7. Click Next to display the Review panel shown in Figure 47. This panel provides the opportunity to review your input and to go back to make any desired changes.

![Review panel](image)

**Figure 47. Review the publishing settings for the Web application**
8. Click **Finish**.

### 3.3.2.1 Publishing

Now that the Web page is designed, the Web page data fields are defined, and the publishing options are specified, you are ready to publish. In this step, WebSphere Studio for AS/400 creates application code, configures the WAS, and moves the objects to the appropriate directories on the iSeries server.

Before you publish, start the Administration Client for your WAS instance and take note of the contents. To start the Administration Client, on a command prompt, enter:

```bash
cd /websphere/appserver/bin
adminclient TORAS48F 790
```

**TORAS48F** is your node or system name, and **790** is the bootstrap port number for your WAS instance.

The default configuration is displayed and should look like the dialog in Figure 48.

![Figure 48. Initial WAS configuration](image)

Our node for the WAS shown in Figure 48 is **TORAS48F**, and that node has defined a Default Server and a Default Servlet Engine. This configuration is updated by WebSphere Studio for AS/400 during publishing. This configuration is revisited after publishing to see the effects of WebSphere Studio for AS/400.

You now publish your page by performing the following steps:

1. Highlight **ABCDemo** in the left pane. Right-click and select **Publish whole project**. This displays the Publishing Options dialog, which is shown in Figure 49.

---

56  IBM WebSphere Development tools for AS/400: An Introduction
This allows the user to specify the types of warning, prompts, and reports that they want to have during the publishing phase. For our example, we chose not to verify published files via HTTP and to publish all the files, as opposed to only publishing the modified files.

2. Click OK. The Files to Publish dialogue appears and asks which files you want to publish. This display only appears if some of the files are checked out. If the dialog appears, check the files you want to publish and click OK.

3. WebSphere Studio for AS/400 displays the Set Publishing Option dialog shown in Figure 50 on page 58. Choose to view the Web page after publishing and supply the Web page name and the HTTP port you want to use. In our example, we used port 702 from our earlier configurations.

Check the configurations by supplying the version of the WAS you are using and the name of the WAS instance (which, in our example, was itso2). If the Check Configuration box is selected, WebSphere Studio for AS/400 checks to see if there is an application server configured on this instance of the WAS. It creates one if none exist. There is no requirement to have WebSphere Studio for AS/400 create these items. You may create them yourself using the Administrative Server and simply publish them to the correct path.
4. If you selected the Check Configuration box, a WebSphere Studio for AS/400 message box appears explaining that it is retrieving configuration information. This may take some time, so be patient. It returns with the Configuration Check Result dialog, which is shown in Figure 51.

5. Click **Proceed with Configuration** to create the Web application.

6. A “Performing Configuration on TORA$48F” status bar appears. **TORA$48F** is the name of the node that you are using. Studio for AS/400 now creates the application server, the servlet engine, and the application in your WAS instance.

7. Once WebSphere Studio for AS/400 is finished with the configuration, it displays the Configuration Check Result window shown in Figure 52. Click **OK**.
8. WebSphere Studio for AS/400 opens a session on your default browser containing a publishing report. It tells you if there were any problems encountered during publishing.

9. A WebSphere Studio for AS/400 dialog box asks whether you want to restart WAS. Select Yes.

10. Now that the application is created, review the changes that occurred in the WAS and in WebSphere Studio for AS/400.

11. In WebSphere Studio for AS/400, a number of files were added. As illustrated in Figure 28 on page 42, WebSphere Studio for AS/400 created some servlets, etc., and placed them on the iSeries server. Use Operations Navigator or the OS/400 WRKLNK command to review the directories that you specified in the Publishing Set Up Wizard. Then, you see the new entries.

WebSphere Studio for AS/400 also configured the WAS instance for you. If you start the Admin Client, you see that new entries have been added.
Figure 53. WebSphere after Publish

Figure 53 illustrates how WebSphere Studio for AS/400 added Wdt400AppServer as an application server (WDT400ServletEngine) and the ABCDemo application for you.

On the display shown in Figure 53, click the Advanced tab and note the entries for the Document Root and for the Class path. These entries were entered and used during the publishing phase. They must agree with those that are used during the publishing phase.

3.3.3 Web Interaction Wizard

Up to this point, you have designed the Web page and the data fields on the Web page that sends or receives data from the ILE program on the iSeries server. The final stage is to define how the pages and data fields interact with the ILE program. This is accomplished with the Web Interaction Wizard.

Remember that the basic architecture provided by WebSphere Studio for AS/400 is a Web page in (with some input fields) and a Web page out (with some output fields) as well as an associated error page. As mentioned earlier, this can be modified with tools such as Flow control. The basic elements that you are dealing with and trying to tie together are:

- Web pages and their input and output fields
- The parameters that are passed to and from the ILE program

In your example, use the page you designed earlier as an input page. Let WebSphere Studio for AS/400 define the output page and request an associated error page. Also, let WebSphere Studio for AS/400 design and create the error page. Follow these steps:
1. Highlight the Project name, which was ABCDemo in our case. Select **Tools->Wizards->Web Interaction Wizard**. The Web Interaction Wizard panel appears. It provides a good overview of the wizard functions. Take some time to read the outline in Figure 54. It provides a good overview of the functions of the Web Interaction Wizard.

![Web Interaction Wizard](image)

**Figure 54. Web Interaction Wizard: Introduction**

2. Click **Next**. Then, the “Specify a name for your web interaction” dialog appears. Figure 55 on page 62 shows an example of how you can provide a name for the particular interaction that you are about to define. If you previously defined Web interactions for this or other pages, you can also select the **Edit existing** button and choose the page that you want to work on from the associated drop-down list.
The Invalidate session after the interaction occurs option allows you to inform the session manager that the session should be terminated after this interaction. This is specified on the last interaction in a series of interactions.

The “No host program call” option allows you to define a Web interaction without specifying ILE program parameters. This is used when defining interactions for JSPs that use subfiles.

In our case, we selected Create and called our name interaction “getcustno”.

3. **Click Next.** Then, the “Specify the input and output pages for your Web application” dialog appears. This dialog allows you to specify which input and output pages you want to use, or if you want WebSphere Studio for AS/400 to create the pages for you. You can also request that WebSphere Studio for AS/400 create an error page (Figure 56).
4. In this example, you want to use the page that you just created for your input. To do this, select **Use existing input pages**. Click **Add**.

The Select JSP box shown in Figure 57 appears.

5. Select **custno.jsp** and click **OK**.

The name of your input JSP, custno.jsp, appears as illustrated in Figure 58 on page 64.
In your case, let WebSphere Studio for AS/400 design your output or results page and error page. Select **Create a new results page**. Select **Create an error page**.

6. Click **Next**.

The “Specify the input and output parameters for your ILE program” dialog box appears (Figure 59). It contains an icon for a program and one default data field.
This program name is the name of the Java program that is generated and that calls your ILE program. By default, it is the name of the interaction that you defined earlier appended with PGM.

You can now define the parameters associated with the ILE program that you want to call.

**Note:** This refers to the parameters that you pass to the programs. The steps that are outlined later allow you to link these parameters with the input and output fields on the Web pages where you want to use these parameters.

You can define data fields and structures that can be passed as parameters to the ILE programs. A structure is a collection of data fields that you define and can store for reuse in another interaction later. Only seven fields can be passed as parameters to the ILE service program and these fields may be structures.

7. Select `getcustnoPGM`.

Select **Program object**, and enter the name of the ILE program to be called. In our example, it was `ORDENTR`.

Select **Program library**, and enter the library name for the ILE program. In our example, it was `APILIB`.

The dialog should appear similar to the example shown in Figure 60.

![Figure 60. Specifying the AS/400 program](image)

8. You have now defined the ILE program you want to call and are now ready to define the parameters that you want to pass to the program. In the next step, you assign names and characteristics to these parameters. The number of parameters and their characteristics must match those defined in your ILE program.
You want to pass:

- An input parameter to the ILE program: Customer number
- Return or output parameters from the ILE program
  - Customer number
  - Next available order number
  - Customer address line 1
  - Customer address line 2
  - City
  - State
  - ZIP code
  - RETURN code to tell you whether the call was successful, which communicates whether the customer number was found

9. Define the fields for the output parameters. Select `getcustnoPGM`. Right-click, and select `Add->Structure`. The “Create a new structure” dialog box appears as shown in Figure 61.

![Create a new structure dialog box](image)

Figure 61. Naming the parameter structure

In the “Create a new structure” dialog box, name the structure `CSSTRUC`, and click `Set`.

This adds another field below the `getcustnoPGM` icon and a structure. The structure `CSSTRUC` has a default field added to it called `Field2` as shown in Figure 62. You can now begin to define your parameters with this field name.
10. Select Field2. In the program, select the *Field name under Program element attributes, and change the name to CZIP. Change Length to 10 so that the characteristics of this parameter (a 10-byte character parameter) matches the definition in the ILE program.

Right-click CZIP, and select Add->Field to add the next field. Continue this until all of the remaining parameters are defined as shown in Table 6. The program call definition box appears as shown in Figure 63 on page 68.

**Table 6. Field of structure "CSSTRUC"**

<table>
<thead>
<tr>
<th>Field name</th>
<th>Type</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>CID</td>
<td>Character</td>
<td>4</td>
</tr>
<tr>
<td>ORDNRBR</td>
<td>Character</td>
<td>4</td>
</tr>
<tr>
<td>CUSTOMER</td>
<td>Character</td>
<td>30</td>
</tr>
<tr>
<td>CADDR1</td>
<td>Character</td>
<td>20</td>
</tr>
<tr>
<td>CADDR2</td>
<td>Character</td>
<td>20</td>
</tr>
<tr>
<td>CSTATE</td>
<td>Character</td>
<td>2</td>
</tr>
<tr>
<td>CZIP</td>
<td>Character</td>
<td>4</td>
</tr>
</tbody>
</table>

11. Select CSSTRUC in the Program element attributes panel. This usage is defined as inputoutput by default. However, you should use this structure as output to the results page so the usage is changed to output.
Notice that the getcustnoPGM now also has a structure called Field3. You should rename this structure by following these steps (Figure 64):

a. Click Field3.
b. Select the Field name entry field on the right side.
c. Change the name to Custparm.

So far, under getcustPGM, you defined your parameters, including one for input, and the structure containing the parameters for output. You need to
define a field that can contain a return value from the ILE program to tell you whether the action was successful.

Call this parameter RETURN.

12. Right-click **getcustPGM** in the left-hand list. Select **Add->Field**. A field is added to the end of the parameter list. Make this the RETURN parameter (Figure 65).

a. Click the added field on the right side. Change the *Field name to RETURN.
b. Change the length to 8.
c. Change the usage to output.

The usage was changed to output since it will be output only from the ILE program.

You are now ready to continue to the next step. However, before exiting this dialog, there are several items you should note.

If you right-click **CSSTRUC**, you notice that you can add fields from a database. You can also add this structure to a repository for later use.

If you right-click the **getcustnoPGM** icon, you also notice that, if you enter the Program source in the right list of attributes, that you can now click the Show Source push button at the bottom. This starts a session of CODE400 and allows you to have the ILE program open for reference as you define the parameters.

13. Click **Next**. The Map and link input parameters to input fields panel appears (Figure 66 on page 70). The list of Input Parameters reflects the parameters and their field names that we defined in the previous steps and are shown on the left-hand side. The input fields for the Web page are defined on the right-hand side. Perform the following steps:
a. Select Field1 from the Input parameters.
b. Select custno from the Input fields.
c. Click the Link button.

This links the input parameter for the ILE program to the input field custno from the Web page.

We also have the ability to Save an input field to the session by selecting Session from the bottom of the Input fields area. You can now restore this data from the session in a later interaction as an input parameter that is not linked to an input field. This is similar to having a global variable for the session.

Figure 66. Linking input parameters

14. Click Next.

The Select the output parameters to be included dialog that appears in Figure 67 defines the parameters that you want included on your output or results page. You can also change the order of the parameters.
The value of your error return parameter should not appear on the output or results page. Highlight the \textit{RETURN} parameter in the parameter list, and click \textit{Include} on the right-hand side to exclude this parameter.

15. Click \textit{Next}.

You are now presented with the “Specify the layout and error messages for the error pages dialog” box. You can use an existing error page or create a new error page as shown in Figure 68. You can also define program and user error handling. If you do not choose to define the program and user error, and request the Create new error page, the error text listed in the box appears on the error page.

In your case:

a. Click the \textit{Create new error page} button.

b. Click the \textit{Define program and error handling} check box.

To display more meaningful messages, create a message file called ORDMSGF in the APILIB and add an error message (ORD0001) that is displayed if the customer number is not found. Also, determine that, if the customer number is found, a return value of 0 appears in the \textit{RETURN} parameter. Figure 69 shows the results of your selection. If the customer number is found, your ILE program sets \textit{RETURN}=0. If the customer number is not found, your ILE program sets \textit{RETURN}=ORD0001. This is required because there may be many error conditions and the application must know which one to use. This means that, if \textit{RETURN} does not equal 0, and there is no valid return code, a blank page is displayed.
16. Click Finish.

WebSphere Studio for AS/400 is now ready to begin generating the application code and generate the JSP files and Java files listed in Figure 69.

WebSphere Studio for AS/400 creates the Results and Error page JSPs, servlets, and DataBeans that are required for this interaction. This is expected, so click the OK button.

17. WebSphere Studio for AS/400 displays a dialog box indicating that it is creating the objects (Figure 70).
Some other dialog boxes appear asking if you want to restart the WAS. You should answer Yes.

18. Publish the results and place the generated code on the iSeries server. Return to the publishing view and publish the project.

Figure 71 shows an example of the type and number of items that WebSphere Studio for AS/400 generated.

Now that the objects are created, test them to see how they are working.

19. Request the initial JSP. Start your JavaScript-enabled browser. In the URL location or address window, type:

   http://your.server.name:port/getorder/custno.jsp

   Here, your.server.name is the host name of your iSeries server, and port is the port number using the HTTP server (Figure 72 on page 74). In our example, we used: http://TORAS48F:702
20. Type a valid customer number and check the results (Figure 73). In our example, we entered 0001.

21. If you typed invalid results, you receive the error message that you defined earlier (Figure 74).
3.3.4 Tips

Whenever you are developing applications with WebSphere Studio for AS/400 and encounter problems, be sure to check the following items:

- Is the HTTP server running?
- Is your WAS instance active?
- Is AS/400 NetServer host job active (used during publishing)?
- Is the Web application’s Web path, in the publishing set up, correct and unique?
- Did you specify the correct port numbers where required?
- Was the host name fully qualified in the first step of the publishing wizard?
- Did you fully qualify the XML host name? If so, this is incorrect and may cause problems.
- Are the fields that you expected to see as input or output for the iSeries server defined within a form?
- Are the paths correct (this is the most common problem)?
- If there are runtime errors when navigating the Web pages, one can get more detailed error message on the error page by selecting the “Display Detailed Errors” check box in “Set Publishing Options” during publishing.

3.3.5 Considerations

WebSphere Studio for AS/400 is designed to allow iSeries users to quickly Web enable new applications by using their current iSeries skills. WebSphere Studio for AS/400 does this by generating the Java code that is required to execute and by providing an easy-to-use interface to the ILE programs.
WebSphere Studio for AS/400 can also be used to enable current applications. However, the design of the applications has a major impact on how practical it is to use this approach. This approach basically replaces the 5250 screens with Web pages. As mentioned earlier, there are significant differences that may make this difficult to do.

One other consideration is the use of subfiles. In WebSphere Studio for AS/400, the Web pages that use subfiles must interact with *SRVPGM objects, and there is no input allowed into the subfile. This consideration can cause a considerable modification to current programs. Subfiles are implemented by APIs in the *SRVPGM. The particulars for these APIs are contained in Appendix A, “Subfile Design Time Control” on page 87.
Chapter 4. Leverage and extend 5250 applications to the Web

iSeries servers have a wealth of 5250 host applications. Web-enabling these host applications is the easiest and fastest method for rapidly increasing the number of iSeries e-business applications. The WebFacing Tool is part of WDT/400. This tool is designed to take existing iSeries programs and convert them to run on the Web. This is in contrast to WebSphere Studio for AS/400 that is designed to enable new applications on the Web.

IBM introduced a WebFacing Tool Early Adopter Program for its AS/400 Solution Providers. This program has two goals. First, by helping solution providers convert 5250 applications to graphical applications running on a Web browser, IBM hopes to increase the number of available iSeries Web-enabled applications. Second, through its work with the solution providers, IBM wants to help define the DDS keyword support required to make the transformation process simpler and usable when the product becomes generally available.

The Early Adopter Program is a closed beta program with a difference. IBM plans to provide sales and marketing support for solution provider applications that have been successfully converted using the WebFacing Tool. For such applications, IBM will publish a joint IBM and Solution Provider press release regarding the successful Web-enabled version of the solution provider product.

Solution providers can apply to participate in the Early Adopter Program by filling out a form on the WebSphere Development Tools Web site at:
http://www-4.ibm.com/software/ad/wdt400/

4.1 Introduction to the WebFacing Tool

The WebFacing Tool is a new product that provides a quick method for deploying entire existing iSeries host applications to Internet users with minimal modifications to the original application.

The WebFacing Tool provides a simple mechanism for facing existing 5250 applications with HTML user-interfaces. This allows users to interact with the same application from a Web browser.

The WebFacing Tool consists of two parts:

- A Display File (DSPF) to JavaServer Pages (JSP) conversion tool
- A runtime intercept to enable the iSeries application to be run as a Web-enabled application or as a 5250 display device application

The WebFacing Tool allows users to convert their existing 5250 display file source (DDS) to corresponding JSP and associated JavaBeans. The user interface is converted to JSPs only once (at development time). The WebFacing Tool is not a 5250 emulation or screen-scraping product. This approach provides significant performance improvements over the “screen-scraping” approach that attempts to convert a 5250 data stream to HTML on the fly.

The JSPs maintain the look and feel of the original application. JavaBeans are used to communicate input and output data between the JSPs and the original iSeries application. The JSPs and JavaBeans are then deployed to the WebSphere Application Server running on the server.
At runtime, users can invoke the iSeries application from a browser or from a traditional 5250 Display device. The invocation method is detected by the iSeries server and a switch is set that enables workstation manager to identify if the application was invoked from a Web browser or from 5250 Display device. Accordingly, the iSeries server then knows to send and receive the program data to and from the Web browser, or to create a 5250 data stream for transmission to the 5250 Display device.

Figure 75 shows the WebFacing runtime scenario.

When the WebFacing runtime has output data from the program, it converts it into a Java data bean and sends it off to the appropriate JSP for displaying in a Web browser. When the WebFacing runtime receives a Java data bean from the browser, it converts it into a program buffer that can be understood by the original iSeries program.

Depending on how the application was started, the WebFacing runtime allows the same iSeries application to run in either a 5250 display device or in Web mode. This is made possible by the WebFace runtime switching application I/O data from the 5250 data-stream generation path to a Web-publishing path when the application is invoked from a browser.

To support the WebFacing Tool, you must install WebSphere Application Server Version 3.5 and configure it on your iSeries server.

4.2 WebFacing sample

In the following sample, use the WebFacing Tools to convert an existing iSeries application to illustrate how the WebFacing Tools work.

In this example, the application presents a 5250 screen and requests that the user enter a customer number as shown in Figure 76.
Chapter 4. Leverage and extend 5250 applications to the Web

Figure 76. Entering the customer number

When the user enters the customer number, the application returns the next order number and the customer information. The screen also allows the input of the part number and the quantity ordered as shown in Figure 77.

Figure 77. Returned customer information

If the user does not know the correct part number, they can use CMD 4 to prompt for a listing of parts as shown in Figure 78 on page 80.
This sample uses WebFacing Tools to convert the DDSs and allows the application to run on the Web. Since this is a preview, all of the steps involved are not covered. Rather, the major areas are covered and the results achieved with the WebFacing Tool are shown.

Start the WebFacing Tool and choose to create a new project called webface. Once you connect to the iSeries server, you are presented with a library list. Choose the DDSs that you want to convert. You can then select them as shown in Figure 79.
Chapter 4. Leverage and extend 5250 applications to the Web

There are several more choices that must be made, such as Web page style, before you request that the project be created.

Once the project is created, you see a dialog box similar to the one displayed in Figure 79. You have already supplied WebFacing information via the Publish Information icon telling WebFacing the name of the iSeries server that runs your application.

You are now ready to convert any or all of the DDSs that you selected for this project. As shown in Figure 79, you chose to convert the three DDSs since you know from experience that these are the DDSs that you require to run your application. Once you select the DDSs, WebFacing converts them and supplies the JSPs, servlets, and beans that you require to run your application.

Figure 80 on page 82 shows the results of the conversion and all of the classes that were generated for the ORDENTD DDS.
You must now move these objects to the iSeries server to execute them. Indicate to the WebFacing tool which iSeries server you want to use, and select **File->Export**. Then, the Export dialog in Figure 81 appears.
You must now supply the directory name that you are using and the resource that you are exporting. In this example, you are exporting the webface project. The Directory that you supply in the Export dialog is the directory that is used by your WAS instance. In this example, you use the ITSO1 instance of the WAS.

The DDSs for your application has been converted and exported to the iSeries server and you are ready to test the results. Figure 82 through Figure 84 on page 85 show the results.

Figure 82. Request customer number page
Figure 83. Return customer information page
Now you have an application that you can access from the Web as well as from the 5250 screens. As you have seen in this chapter, this example does not require any changes to the RPG source program.
Appendix A. Subfile Design Time Control

WebSphere Studio for AS/400 Subfile Design Time Control (DTC) provides similar functionality to the iSeries subfile. For example, you can use the Subfile DTC to display a list of records that consist of more than one field. You can use the properties dialog of the Subfile DTC to control properties, such as the number of records displayed at a time.

A.1 Subfile DTC and ILE programs

Subfile DTC provides a way for your ILE program to present the user with a list of items. Users select a record in the Subfile DTC by selecting the radio button or check box that is associated with each record. If the subfile is defined as single select, check boxes are shown. If the subfile was defined as multiple select, radio buttons are shown.

Your ILE program interacts with the subfile by coding to the APIs that are documented in the following section. The data for a subfile is actually stored in a user space object (*USRSPC object type) on the iSeries server. This user space is created in library QTEMP for the job that is executing the ILE program.

When you create a subfile DTC on one of your JSPs, one of the properties you set is the name of the service program that should be called each time the subfile is shown. It should also be called when the user presses the Page Up or Page Down push buttons that are generated by the subfile. This service program should provide the procedures (shown in Table 7) that are called by the subfile DTC. Note that these procedure names are case sensitive. Each of these procedures are passed two parameters. The first parameter is a 10-character string that is the name of the subfile DTC as defined in the subfile DTC control properties settings (this name is required by each of the subfile APIs). The second parameter of the Subfile service program procedure is a structure that maps to the parameter-list defined in the Subfile DTC properties page.

Table 7. Procedure of Subfile DTC

<table>
<thead>
<tr>
<th>Procedure</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>INIT</td>
<td>This procedure is called each time the page containing the subfile DTC is about to be displayed. You can use this procedure to initially fill the subfile with records or a page worth of records. INIT is only invoked by the Subfile DTC when the DTC is first displayed (in other words, created).</td>
</tr>
<tr>
<td>PGUP</td>
<td>This procedure is called when the user presses the Page Up button associated with the subfile to view the previous page of records. If previous records exist in the user space, this procedure is not called and the subfile automatically displays the previous page.</td>
</tr>
<tr>
<td>PGDN</td>
<td>This procedure is called when the user presses the Page Down button associated with the subfile to view the next page of records. If the next page of records exist in the subfile, this procedure is not called and the subfile automatically displays the next page.</td>
</tr>
<tr>
<td>CLUP</td>
<td>This procedure is called when the session ends and performs a cleanup operation.</td>
</tr>
</tbody>
</table>
A.2 Subfile DTC APIs examples

This section describes each of the APIs that are available for processing a WebSphere Studio for AS/400 subfile DTC (Table 8).

Table 8. Subfile DTC APIs

<table>
<thead>
<tr>
<th>API name</th>
<th>Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>QdtsAppendSF</td>
<td>Add a record</td>
</tr>
<tr>
<td>QdtsChainSF</td>
<td>Chain to a record</td>
</tr>
<tr>
<td>QdtsClearSF</td>
<td>Clear the subfile</td>
</tr>
<tr>
<td>QdtsReadcSF</td>
<td>Read a selected record</td>
</tr>
<tr>
<td>QdtsInitSF</td>
<td>Initialize the subfile</td>
</tr>
<tr>
<td>QdtsUpdateSF</td>
<td>Update a record</td>
</tr>
</tbody>
</table>

A.2.1 QdtsAppendSF: Adding a record to the subfile

Use the QdtsAppendSF API to add a record to the subfile. Records are added to the end of the subfile (Figure 85).

Figure 85. Syntax of the QdtsAppendSF API

The parameters are (Figure 86):

- uRC: 0=Success, -1=Fail
- pSFName: Address of the space name
- pRcd: Address of the subfile record to be added
- length: The length of the subfile record

Figure 86. Sample code for the QdtsAppendSF API
A.2.2 QdtsChainSF: Chaining to a specific subfile record

Use the QdtsChainSF API to chain to a specific record in the subfile. If the chain is successful, the subfile record will available in the record data structure (Figure 87).

```
Eval uRC=QdtsChainSF(pSFName: pRcd: length, uRRN)
```

Figure 87. Syntax of the QdtsChainSF API

The parameters are (Figure 88):

- `uRC`: 0=Success, -1=Record not found
- `pSFName`: Address of the space name
- `pRcd`: Address of the structure to receive the subfile record
- `length`: The length of the subfile record
- `uRRN`: The relative record number of the subfile record to chain to (read)

```
Eval uRC=chain(%Addr(SFName: %Addr(Record): len: uRN)
* Name = Donna (possibly)
* City = Guelph (possibly)
* uRRC = 3 (possibly)
```

Figure 88. Sample code using the QdtsChainSF API

A.2.3 QdtsInitSF: Initializing the subfile

Use the QdtsInitSF API to initialize the subfile for first time use. This API is typically called by entering it in the INIT procedure that is called each time the JSP containing the subfile is to be displayed. The INIT procedure is only invoked by the Subfile DTC when the DTC is first displayed (in other words, created) (Figure 89).

```
Eval uRC=QdtsInitSF(pSFName: pReserved: length)
```

Figure 89. Syntax of the QdtsInitSF API

The parameters are (Figure 90 on page 90):

- `uRC`: 0=Success, -1=Fail
- `pSFName`: Address of the space name
• **pReserved**: Address of any 10 character string (unused by the API, but must be coded)
• **length**: The length of the subfile record

![Sample code using the QdtsInitSF API](image)

### A.2.4 QdtsReadcSF: Reading the selected record

Use the QdtsReadcSF API to read a record that was selected in the subfile. Select records in a subfile by selecting the radio button or check box that is associated with each subfile record (Figure 91).

![Syntax of the QdtsReadcSF API](image)

The parameters are (Figure 92):

- **uRRN**: The relative record number of the selected subfile record
- **pSFName**: Address of the space name
- **pRcd**: Address of the structure to receive the subfile record
- **length**: The length of the subfile record
- **ustartRRN**: The record from which to start searching

![Sample code using the QdtsReadcSF API](image)
A.2.5 QdtsClearSF: Clearing the subfile

Use the QdtsClearSF API to remove all records from the subfile. You can call this API if you are filling the subfile a page at time (Figure 93).

```
CL0N01Factor1+++++++Opcode&ExtExtended-factor2+++++++++++++++++++++++++++++++ 
Eval uRC=QdtsClearSF(pSFName)
```

*Figure 93. Syntax of the QdtsClearSF API*

The parameters are (Figure 94):
- **uRC**: 0=Success, -1=Fail
- **pSFName**: Address of the space name

```
DName+++++++++++ETDsFrom+++To/L+++IDc.Keywords+++++++++++++++++++++++++++++++ 
D clear PR 10U 0 Extproc(QdtsClearSF) 
D pSFName * Value 
D uRC S 10U 0 
D SFName S 10A Inz('SFL1') * 
CL0N01Factor1+++++++Opcode&ExtExtended-factor2+++++++++++++++++++++++++++++++ 
Eval uRC=clear(%Addr(SFName))
```

*Figure 94. Sample code using the QdtsClearSF API*

A.2.6 QdtsUpdateSF: Updating a subfile record

Use the QdtsUpdateSF API to update an existing record in the subfile. It is not necessary to read the record before you update it (Figure 95).

```
CL0N01Factor1+++++++Opcode&ExtExtended-factor2+++++++++++++++++++++++++++++++ 
Eval uRC=QdtsUpdateSF(pSFName: pRcd: length, uRRN)
```

*Figure 95. Sample code using the QdtsUpdateSF API*

The parameters are (Figure 96 on page 92):
- **uRC**: 0=Success, -1=Record not found
- **pSFName**: Address of the space name
- **pRcd**: Address of the structure to receive the subfile record
- **length**: The length of the subfile record
- **uRRN**: The relative record number of the subfile record to update
Figure 96. Sample code using the QdtsUpdateSF API

```
D Name+++++++++++ETDeFrom+++To/I+++IDc.Keywords+++++++++++++++++++++++++++++++
D update PR 10U 0 Extproc(QdtsUpdateSF)
D pSFName * Value
D pRcd * Value
D len 10U 0 Value
D SFName S Inz('SFL1')
D Record DS
D Name 20A
D City 30A
D len S 10U 0 Inz(%Size(Record))
D uRC S 10U 0
D uRRN S 10U 0
*
CL0N01Factor1+++++++Opcode&ExtExtended-factor2+++++++++++++++++++++++++++++++
Eval Name = 'Donna'
Eval City = 'Guelph'
Eval uRRN = 3
Eval uRC=update(%Addr(SFName): %Addr(Record): len: uRN)
```
Appendix B. Sample code

This appendix contains the sample code that was used in the programing example in Chapter 3, “Building new e-business applications” on page 41.

*---------------------------------------------------------------------*
* ROUTINE: ORDENTR *
* This program uses a parameter list to communicate with web pages. *
* The web pages and the interface to this program were created with *
* WebSphere Studio for AS/400. *
* It expects a customer number as input and will return the customer *
* name and address and next available order number to be displayed on *
* a web page. *
* If the customer is not found it will return the error message number *
* in the Return parameter. If it is found it will set Return = 0. *
* *---------------------------------------------------------------------*

FITEM IF E K DISK
FDSTRCT UF E K DISK
FCSTMR UF A E K DISK

** Input parameter from the web page
DCustnoi S LIKE(CID)

** Data structure to specify output structure to the servlet
** Uses the field names that were used for the DSP
D CSPARM DS
D CID
D ORDNR 4A
D CUSTOMER 30A
D CADDR1
D CADDR2
D CCITY
D CSTATE
D CZIP

** Return value to page
D Return S 8

D Custinfo DS INZ
D CustomerID LIKE(CID)
D DistrictID LIKE(CID)
D WarehouseID LIKE(CID)

* *---------------------------------------------------------------------*
* MAINLINE: *
* *---------------------------------------------------------------------*
C *entry plist
C parm custnoi
c parm csparm
c parm Return
C* set return code for not found message. 
C* if records found RqsCust sets Return = 0... meaning found it
C eval Return='ORD0001'
C EXSR RqsCust
C SETON LR
C RETURN
C
* *---------------------------------------------------------------------*
* SUBROUTINE: RqsCust *
* To prompt for a customer *
* *---------------------------------------------------------------------*
CSR RqsCust BEGSR
C* Clear error indicators
C EXSR ClrError
**SUBROUTINE: VfyCust**

* To verify a valid customer

---

**SUBROUTINE: ClrError**

* To clear any error indicators

---

**SUBROUTINE: GetOrdNbr**

* Determine the next order number

---

* Key list for CSTMR file

* Key list for DSTRCT file
Appendix C. Special notices

This publication is intended to help BM marketing personnel, Business Partners, and AS/400e and iSeries customers who are looking to extend and expand their information server into e-business. The information in this publication is not intended as the specification of any programming interfaces that are provided by WebSphere Development tools for AS/400. See the “Publications” section of the IBM Programming Announcement for WebSphere Development tools for AS/400 for more information about what publications are considered to be product documentation.

References in this publication to IBM products, programs or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM's product, program, or service may be used. Any functionally equivalent program that does not infringe any of IBM's intellectual property rights may be used instead of the IBM product, program or service.

Information in this book was developed in conjunction with use of the equipment specified, and is limited in application to those specific hardware and software products and levels.

IBM may have patents or pending patent applications covering subject matter 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 the IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact IBM Corporation, Dept. 600A, Mail Drop 1329, Somers, NY 10589 USA.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The information contained in this document has not been submitted to any formal IBM test and is distributed AS IS. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer's ability to evaluate and integrate them into the customer's operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

Any pointers in this publication to external Web sites are provided for convenience only and do not in any manner serve as an endorsement of these Web sites.
The following terms are trademarks of the International Business Machines Corporation in the United States and/or other countries:

- e (logo)®
- IBM ®
- AIX
- AS/400e
- CT
- IBM.COM
- SP
- WebSphere
- XT
- Redbooks
- Redbooks Logo
- AS/400
- AT
- DB2
- OS/400
- VisualAge
- Wizard
- 400

The following terms are trademarks of other companies:

- Tivoli, Manage. Anything. Anywhere., The Power To Manage., Anything, Anywhere., TME, NetView, Cross-Site, Tivoli Ready, Tivoli Certified, Planet Tivoli, and Tivoli Enterprise are trademarks or registered trademarks of Tivoli Systems Inc., an IBM company, in the United States, other countries, or both. In Denmark, Tivoli is a trademark licensed from Kjøbenhavns Sommer - Tivoli A/S.

- C-bus is a trademark of Corollary, Inc. in the United States and/or other countries.

- Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and/or other countries.

- Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States and/or other countries.

- PC Direct is a trademark of Ziff Communications Company in the United States and/or other countries and is used by IBM Corporation under license.

- ActionMedia, LANDesk, MMX, Pentium and ProShare are trademarks of Intel Corporation in the United States and/or other countries.

- UNIX is a registered trademark in the United States and other countries licensed exclusively through The Open Group.

- SET, SET Secure Electronic Transaction, and the SET Logo are trademarks owned by SET Secure Electronic Transaction LLC.

Other company, product, and service names may be trademarks or service marks of others.