

Host Access Transformation Services on z/OS

HATS version 5.0.4 for z/OS

Enterprise modernization
examples

Performance and
deployment guide



Scott G. Peters



International Technical Support Organization

Host Access Transformation Services on z/OS

March 2005

Archived

Note: Before using this information and the product it supports, read the information in “Notices” on page vii.

First Edition (March 2005)

This edition applies to Host Access Transformation Services Version 5.0.4

© Copyright International Business Machines Corporation 2005. All rights reserved.

Note to U.S. Government Users Restricted Rights -- Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Notices	vii
Trademarks	viii
Preface	ix
The team that wrote this redbook.	ix
Become a published author	x
Comments welcome.	x
Part 1. Introduction	1
Chapter 1. Executive overview	3
1.1 What is HATS	4
Chapter 2. Product overview	7
2.1 HATS overview	8
Part 2. 3270 green screens	11
Chapter 3. The ZYX 3270 Parts application	13
3.1 3270 emulation primer	14
3.2 ZYX Parts	15
3.2.1 Login	15
3.2.2 Starting the application	15
3.2.3 ZYX Parts main menu	16
3.2.4 ZYX Parts Order Form	17
3.2.5 Searching ZYX Parts	18
3.2.6 Search results	19
Part 3. 3270 applications using HATS	23
Chapter 4. Creating a default HATS application	25
4.1 HATS wizards and editors	26
4.2 Creating a HATS project	26
4.2.1 Configure connection settings	27
4.2.2 Selecting the default template	28
Chapter 5. Default HATS application	31
5.1 ZYX Parts with default transformation	32
Chapter 6. Creating a customized HATS application	39
6.1 Modifying the default template	40
6.2 Customizing the default rendering options	42
6.3 Customizing the logon screen	43
6.4 Bypassing the command screen	48
6.5 Customizing the parts main menu	49
6.6 Customizing the Parts Order Form	52
6.7 Customizing the Parts Search screen	53
6.8 Customizing the search results screens	55
6.9 Importing and exporting a HATS project	56
Chapter 7. Customized ZYX Parts	59

7.1 Customized ZYX Parts review	60
Part 4. HATS deployment	63
Chapter 8. WebSphere runtime on z/OS	65
8.1 WAS V5 for z/OS improvements	66
8.2 zSeries hardware and z/OS	66
8.2.1 Central processors and logical partitions	66
8.2.2 Parallel Sysplex®	67
8.2.3 Address spaces and tasks	67
8.2.4 z/OS components	68
8.3 The WebSphere programming model	69
8.3.1 Java overview	69
8.4 Application server model	69
8.4.1 Regions and instances	70
8.4.2 Servers and nodes	70
8.5 Putting it together: a typical customer installation	72
8.6 Performance components	74
8.6.1 The TCP/IP network	74
8.6.2 zSeries server	74
8.6.3 z/OS	75
Chapter 9. HATS and WebSphere for z/OS	77
9.1 Why to run HATS on z/OS	78
9.1.1 The z/OS difference: architecture matters	78
9.1.2 Prioritization of work on z/OS using WLM and IRD	78
9.1.3 WebSphere behavior in a sysplex	79
9.1.4 Anatomy of an enterprise cluster	79
9.1.5 z/OS benefits summary	80
9.2 Deploying HATS applications to WAS v5.1 on z/OS	80
9.3 zSeries Application Assist Processor (ZAAP)	89
9.3.1 What is a ZAAP	89
9.3.2 Why use a ZAAP	89
9.3.3 How does a ZAAP work	90
9.3.4 Prerequisites for the ZAAP	90
9.4 Performance settings and values	91
9.5 Enabling graphics support for HATS on z/Series	91
9.6 z/OS version updates	92
9.6.1 z/OS 1.5 update	92
9.6.2 z/OS 1.6 update	92
9.6.3 z/OS.e on zSeries z800 server	92
9.7 Configuring the Display Terminal for zSeries® when running WebSphere Application Server v5.0.x	92
Part 5. Appendixes	95
Appendix A. What's new in HATS v5.0.4	97
New functions in v5.0.4	97
New functions in v5.0.3	102
Additional options for subfile recognition and rendering	103
Component level in the project settings (application.hap)	103
Default rendering level in the project settings	103
Transformation level	104
New selection list component setting (APAR IC42366)	106

New setting for the field widget (APAR IC41065)	106
HATS portlet support in a Web Service Remote Portlet (WSRP) configuration.	107
Appendix B. APARs fixed in HATS Service Pack 4	109
Problems fixed in Service Pack 4	109
Additional problems	110
Problems fixed in Service Pack 3	111
Additional problems	112
Appendix C. Lab 1: Creating a HATS application.	115
Default project	116
Screen customizations	128
Screen customizations summary	129
Welcome screen	130
Sign-on Complete and blank screens	137
CICS MENU transaction OPERATOR INSTRUCTIONS screen	145
CICS MENU transaction FILE BROWSE screen	157
Extra credit	169
CICS MENU transaction FILE INQUIRY screen	170
Appendix D. Lab 2: Exposing legacy applications as Web services	175
Naming conventions	176
Create a HATS project	176
Create macros	179
Connect macro	179
Data macro	181
Disconnect macro	187
Set connection parameters	188
Create integration object	193
Create Web service support files	195
Related publications	209
IBM Redbooks	209
Other publications	209
Online resources	209
How to get IBM Redbooks	210
Help from IBM	210
Index	211

Archived

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing, IBM Corporation, North Castle Drive Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.


This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrates programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

Trademarks

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

@server®	DFS™	Manager™
Redbooks (logo)  ™	DFSMSrmm™	PR/SM™
Eserver®	DFSORT™	Redbooks™
ibm.com®	IBM®	RACF®
z/OS®	IMS™	RMF™
z/VM®	Language Environment®	S/390®
zSeries®	Notes®	Sysplex Timer®
AIX®	OS/390®	Tivoli®
CICS®	Parallel Sysplex®	WebSphere®
DB2 Universal Database™	Perform™	
DB2®	Processor Resource/Systems	

The following terms are trademarks of other companies:

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

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

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

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

Preface

This IBM® Redbook provides IT managers, application developers, system administrators and architects with an overview of Host Access Transformation Server v5 for z/OS®. The topics in this book use as examples HATS applications that are developed to front-end 3270 programs, as well as HATS applications that can be deployed to WebSphere® on z/OS.

This redbook is also intended to provide an update for HATS v5.0.4. Many of these improvements are available on the distributed platforms as well. Therefore, this book should be of value to any HATS v5.0.4 user.

The benefits of HATS can best be seen by using examples. The ZYX Corporation Parts Ordering System is used in this book to demonstrate a traditional 3270 program, and a base HATS application is created to demonstrate the ease and speed with which a legacy application can be presented through a Web browser. HATS customizations can be done using the wizards in HATS, or by manually modifying project components. Some of the more common customizations are illustrated in this text.

Deployment examples and considerations are then discussed, using the sample ZYX application. The discussion includes the performance, scalability and reliability benefits of running HATS applications on z/OS.

At the end of the book, readers should have a fundamental understanding of the product and be able to create, customize, and deploy a HATS application. The book is presented in five sections, given that the reader base will be diverse:

- ▶ Part 1 includes an executive and product overview
- ▶ Part 2 provides a review of a legacy 3270 application
- ▶ Part 3 describes using HATS to front-end 3270 applications
- ▶ Part 4 describes HATS deployment and considerations, using examples
- ▶ Part 5 provides appendices and two laboratory exercises

The team that wrote this redbook

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

Scott G. Peters is currently a Senior Certified IT Specialist with the IBM Software Group. He joined IBM in 1989 as a Systems Engineer in Chicago. He graduated from the University of Illinois in Champaign/Urbana with a Bachelor of Science in Computer Science. In his fifteen years at IBM, he has acted as a consulting programmer, project manager, solution architect and software proponent.



Figure 1 Scott Peters

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

Rick Hardison, Raleigh, NC for the two laboratory exercises

Kyle Croutwater, Raleigh, NC

James Carmichael, Raleigh, NC

Bruce R Green, Raleigh, NC

Gareth Patterson, San Francisco, CA

Joe DeCarlo, Manager for Special Projects for the IBM Technical Support Organization, San Jose, CA

Become a published author

Join us for a two- to six-week residency program! Help write an IBM Redbook dealing with specific products or solutions, while getting hands-on experience with leading-edge technologies. You'll team with IBM technical professionals, Business Partners and/or customers.

Your efforts will help increase product acceptance and customer satisfaction. As a bonus, you'll develop a network of contacts in IBM development labs, and increase your productivity and marketability.

Find out more about the residency program, browse the residency index, and apply online at:

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

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

- Use the online **Contact us** review redbook form found at:

ibm.com/redbooks

- Send your comments in an e-mail to:

redbook@us.ibm.com

- Mail your comments to:

IBM Corporation, International Technical Support Organization
Dept. HZ8 Building 662
P.O. Box 12195
Research Triangle Park, NC 27709-2195

Archived



Part 1

Introduction

In this part we introduce Host Access Transformation Services (HATS) from an executive perspective. Readers not interested in the development details may subsequently view Chapters 3, 5 and 7, which illustrate a legacy application presented with an emulator, base HATS application, and a customized HATS application.

Archived

Executive overview

Speed to market, flexibility and cost reduction are concerns common to all IT executives. With significant resource investments in monolithic legacy systems, IT organizations struggle with the balancing of old and new technology. While the rewards associated with rebuilding legacy applications are high, so are the risks. Host Access Transformation Services (HATS) was developed to help customers deal with this challenge.

In the e-business On Demand era, business requirements change frequently. To remain competitive, businesses are developing new, strategic Web-based applications. Java™ adoption continues to accelerate as an open programming model, but these applications typically require expensive, highly skilled IT resources. Unfortunately, IT budgets are not keeping pace with these needs, thereby forcing customers to seek more cost-effective and productive ways of deploying new Java technology-based applications.

Today there are over 200 billion lines of application code written in COBOL, with over 5 billion being added annually. Over 70% of data is now being stored in mainframe repositories. The premier platform for running high availability business applications has been and will continue to be the mainframe, at least for the foreseeable future.

HATS provides a strategic platform for enhancing and extending legacy applications leveraging open, industry standard technologies, essentially bridging the gap between new and old programming models. HATS allows for the rapid deployment of legacy applications through a browser, thus eliminating the requirement for terminal emulators without requiring modifications to the legacy code. Utilizing the value of WebSphere Studio and WebSphere Application Server, HATS allows for the creation of server side J2EE applications without any Java programming skills. The value is extended to developers, who can exploit the many features found in these products.

Increased productivity and reduced training costs can be realized by providing a Web-based interface to legacy applications using HATS. When used in a Service Oriented Architecture, HATS can facilitate improved profitability and speed to market by improving strategic agility. With HATS, customers can realize appreciable business benefits, with minimum investment and risk.

1.1 What is HATS

IBM WebSphere Host Access Transformation Services (HATS) Version 5 for zSeries® gives you the tools you need to quickly and easily extend host applications to business partners, customers, and employees. HATS exposes 3270 and 5250 applications as HTML through ubiquitous Web browsers, while converting legacy screens to a Web “look and feel”. HATS can also make it easy to improve the workflow and navigation of your host applications, without any access or modification to source code.

The HATS rules-based transformation engine makes it possible to extend host applications to the Web within hours of installing the software. HATS is a zero-footprint, zero-download, Web-to-host solution; the only software required on the client is a Web browser.

A HATS solution fundamentally consists of three components: the HATS developer, which is responsible for creating the HATS application; the HATS server, which is the WebSphere Application Server that runs the HATS application; and the client workstation, which uses a browser to access the HATS application, as shown in Figure 1-1.

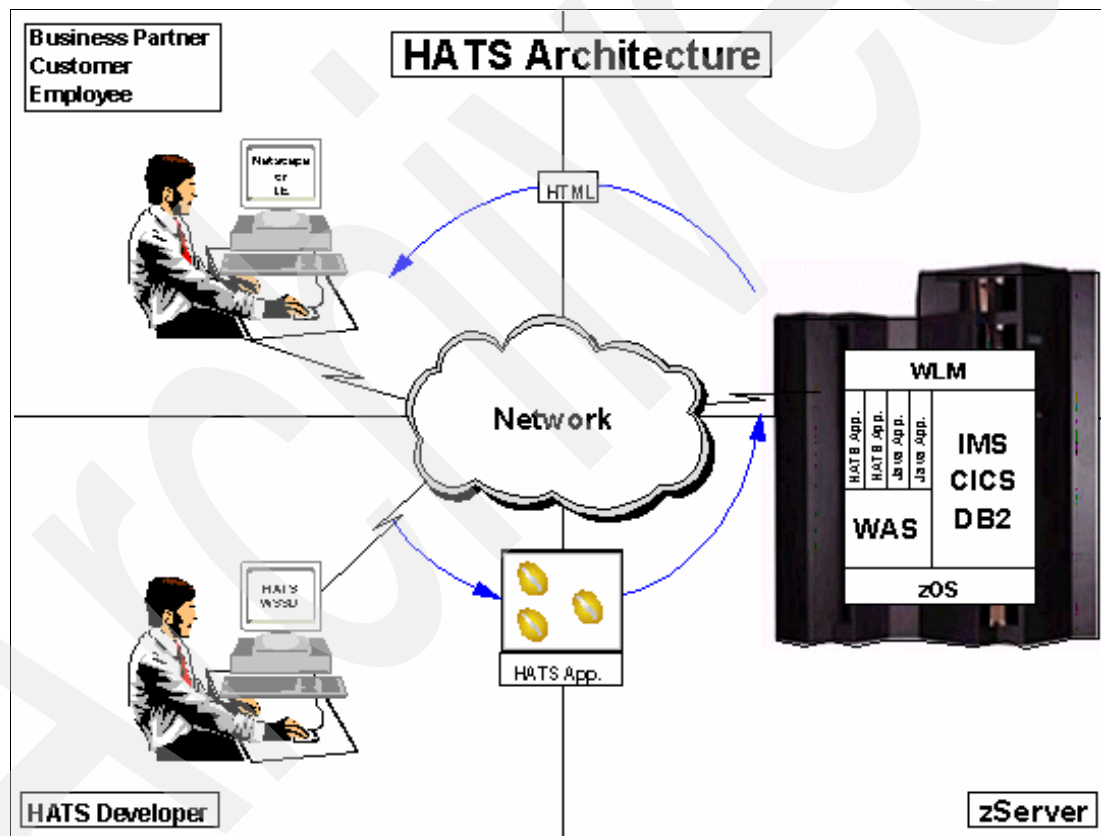


Figure 1-1 HATS architecture

The power of HATS lies in its ability to accurately recognize the components of host screens and transform them in real time to a Web interface according to a set of predefined rules. It is easy to customize the rules according to specific application requirements. HATS can add a variety of elements to host screens, such as drop-down lists, hot links, tables, buttons, valid value lists, tabbed folders, and graphs. HATS can also add HTML elements such as logos, graphics, backgrounds, and Web links.

HATS has macro support that provides programmed navigation through multiple host screens to help improve the productivity and ease of use of legacy applications. Examples of

productivity improvements include skipping and combining screens, prefilling text entry fields, and storing and retrieving data in global variables. HATS can provide such programmed access to a single host application, or can integrate screens from multiple host applications into a single Web interface. HATS can also use macros created in IBM WebSphere Host On-Demand or IBM WebSphere Host Publisher.

The HATS Studio is fully integrated within the Eclipse-based IBM WebSphere Studio. It offers an intuitive interface for customizing the rules for transformation of host screens. The HATS applications are deployed to WebSphere Application Server for zSeries, and take advantage of the extensive security and reliability features of that platform.

Archived

Product overview

In this chapter we provide an overview to Host Access Transformation Services (HATS) Version 5. This release of HATS leverages the power of WebSphere Studio Application Developer V5.1 and is well integrated with the WebSphere portfolio of products. Web Services enables HATS as a plugable asset into a Service Oriented Architecture, which can ultimately enable faster innovation, increased productivity and business resilience. These and other exciting features are revealed in this chapter.

Highlights of HATS Version 5:

- ▶ Programmed access to host transactions through standard Web services interfaces
- ▶ Integration of multiple host applications into a single Web interface
- ▶ Support for VT business objects
- ▶ Added support for customization of default rules
- ▶ Automated migration from Host Publisher V4 projects

2.1 HATS overview

The rules-based implementation of HATS V5 for zSeries can make it possible to extend virtually any 3270 or 5250 application to the Web within a single day of installing the software. The HATS rules define how the various host components are to be transformed and represented as HTML. For example, the rules define whether selection lists should be represented as hot links, buttons, drop-down menus, radio buttons, or tables. The HATS rules are applied on the fly to host screens to generate the corresponding Web pages for end users. These rules can eliminate the need to customize individual host screens and significantly reduce the time to market for new Web-to-host projects. Virtually any host screen can be rendered according to HATS' default rules, or can be partially or fully customized according to the needs of the specific application.

HATS provides the flexibility to assign different rule sets to different end-user communities. For example, you can give a single host application a variety of looks and workflows that are appropriate for different end-user groups. Alternatively, you can use a single rule set on different applications, which will enable you to reuse your work across the various host applications in your organization.

Support for WebSphere Portal running on WebSphere Application Server on the zSeries platform will be through a provided iFrame portlet. HATS works with IBM Tivoli® Access Manager and third party products to provide support for single sign-on.

HATS provides the ability to create connection pools, which can be used at run time to cache connected, logged-on, and ready host connections. A predefined number of connections can remain active in the pool, supporting requests from any user. This can help eliminate the overhead of establishing a connection, logging on, and disconnecting for each host request. You can define the minimum and maximum number of connections in the pool, as well as determine whether HATS should wait for an available connection or create a new, non-pooled connection if all predefined connections are in use.

HATS supports a broad range of security features, in addition to leveraging the security provided by WebSphere Application Server. Secure Sockets Layer (SSL) and Secure HTTP (HTTPS) provide security between the host application, the mid-tier server, and the end user. HATS will appear as a typical user to your host applications, allowing you to continue to take advantage of existing access security systems, such as IBM Resource Access Control Facility (RACF®).

Local print support is an important capability when extending legacy applications to new users. HATS provides a user-friendly means of delivering this support by converting host print output into industry-standard PDF format. The resultant PDF file is sent to the end user's browser, where it can be printed locally or saved for later use.

HATS offers native keyboard support to allow users to continue to use the keyboard functionality to which they have become accustomed. Although running in a browser environment, HATS is able to provide direct keyboard support for functions employed by the host application, such as PF keys, enter keys, clear keys, and system-request keys.

HATS Studio is fully integrated within the Eclipse-based IBM WebSphere Studio. The Eclipse platform is an industry-standard application development environment, providing the benefits of a common framework and reusable skill set for development of Web-based applications. Integration with WebSphere Studio provides a common tooling family for all your e-business needs. WebSphere Studio application development features provide a variety of additional benefits, such as team development facilities that enable code management across multiple developers.

HATS integration with WebSphere Studio provides a flexible and extensible environment in which to integrate host applications with Java-based applications. Screens and data from multiple host sources can be combined with Java-based applications to create new WebSphere applications. Transactions with host systems can be encapsulated into reusable business objects, such as Web services, Java beans, or Enterprise Java Beans. HATS and WebSphere Studio wizards can be used to create Web pages that call these new business objects. These Web pages can use traditional Model 1, or the newer, open standard Struts standard for Model 2-based Web pages. WebSphere Studio wizards can also be used to create SQL queries and business objects (Web services, Java beans, or Enterprise Java Beans) to execute these SQL queries.

HATS, in conjunction with WebSphere Studio, simplifies the creation of standard Web services interfaces to provide access to host applications. Web services protocols, such as Simple Object Access Protocol (SOAP) and Web Services Description Language (WSDL), are an efficient and reusable means of providing standardized access to your host systems, helping you lower the cost of maintaining and deploying connectors to these systems.

The run time components of HATS are generated by the HATS Studio and deployed to IBM WebSphere Application Server. The WebSphere platform provides support for the workload management features required for enterprise-class scalability and availability. Load balancing and fail-over support functions, such as vertical and horizontal cloning, are handled by the WebSphere Workload Manager.

Users familiar with HATS v4 will find many new enhancements in HATS v5. Table 2-1 lists some of the more significant improvements.

Table 2-1 HATS V5 enhancements

Feature/Enhancement	Benefit
Support for multiple back end systems (5250/3270/VT) and global variable support across multiple host sessions	Provides greater flexibility in simplifying end-user experience
Integrated support for WebSphere Portal Server including Click-to-Action, Cooperative Portlets, Credential Vault	Allows for full integration at the glass of legacy and new applications, creating a consistent users experience single sign-on environment
Enhanced components and widgets, Global Rules, enhanced screen recognition, pluggable default rendering	Allows richer HTML customizations with fewer rules
Enhanced support for Struts and Web Services	Legacy business logic can be exposed using industry standard SOAP/WSDL/Struts/JSPs
Support for Linux® and Linux on zSeries	Allows even greater flexibility and scalability for HATS application deployment
Centralized administration of host systems and applications	Provides a secure, centralized method for administering HATS applications
Web Express Logon with Tivoli Access Manager	Facilitates end-user access to multiple systems by enabling single sign-on
Enhanced macro recording and editing	Enables developers to create powerful macros to combine screens, streamline application flow, and create business objects (HIOs)
Host Publisher V4 to HATS V4 and HATS V5 automated migration tools	Allows Host Publisher V4 and HATS V4 customers to easily migrate to HATS V5

Archived



Part 2

3270 green screens

In this part we explore the use of Host Access Transformation Services (HATS) to Web-enable the ZYX Electronics parts ordering application. Like many corporations, ZYX Electronics runs its mission-critical ordering application on z/OS for unsurpassed availability, reliability and security. ZYX Electronics would like to Web-enable its ordering system, using a new portal that will allow them to enter into new markets and increase revenue. We start by introducing the legacy 3270 parts application to establish a functional baseline.

Archived

The ZYX 3270 Parts application

In this chapter we review a standard 3270 application, the ZYX Electronics Corporations parts ordering system, called ZYX Parts. This mission-critical application is made available to business partners using a 3270 emulator. We provide an example of performing a part search against ZYX Parts and demonstrate the following:

- ▶ Existing user interface
- ▶ Workflow for common business function
- ▶ Skills required for navigation
- ▶ Usability issues

3.1 3270 emulation primer

Before examining the application, a review of 3270 terminals and emulation is in order. The 3270 protocol is generically used to apply to information system-dependent displays that utilize the IBM-architected 3270 data stream to interact with application on S/390®-type hosts. The 3270 dependent displays, or terminals, were the mainstay of the computer industry. In order to function, they require a controller to provide most of their functions (CUT) and physical connectivity (CUT and DFT) to the host.

The software implementation of the functionality found in a 3270 dependent display is commonly referred to as 3270 emulation. One of the major advantages of 3270 emulation is that the requirement for a controller, in most cases, is removed since the PC, adapters and the emulation software provide all that is needed to make the physical connection as well as the functions to interact with the host 3270 applications.

The 3270 data stream that is used for interaction between the 3270 peripheral devices (displays, emulators and controllers), is two-tiered: the base 3270 data stream and its extensions. The extensions (commonly called extended data stream) provide the enhanced capabilities for expressing:

- ▶ Color
- ▶ Character set (single and double byte character set)
- ▶ Extended highlighting (for example blinking and underscore)
- ▶ Field validation (validation of data within a field)

The extensions that are supported by an emulator are conveyed to the host application using the 3270 data stream query process. Traditionally the 3270 data stream was intended for hierarchical SNA networks where the emulation program was a PU Type 2.0 node supporting LU Types 0, 2, 3 or 6. The 3270 data stream is the only data stream used on LU-LU session types 2 and 3. It is optional on LU-LU session types 0 and 6.

The flattening of networks to TCP/IP created a non-SNA requirement to the architecture to provide support for the 3270 data stream. TN3270 and subsequently TN3270E were developed and are the primary 3270 emulation type in the marketplace today.

In 1998, RFC 2355 for TN3270E was written to extend the functionality of TN3270. A number of enhancements were made, most notably the addition of certain function keys (including ATTN and SYSTEQ), SNA response handling and LU name specificity. Also, contention resolution functionality was added which can improve communications performance, resulting in reduced response time. HATS takes full advantage of these improvements when the TN3270E protocol is selected.

3.2 ZYX Parts

In the following sections we discuss the ZYX Parts application.

3.2.1 Login

Today most users have many different systems that require authentication. They are required to manage a significant number of user ids and passwords for both internal and external applications. ZYX Parts users must begin their interaction with the application by first logging in to the ZYX 3270 system as shown in Figure 3-1.

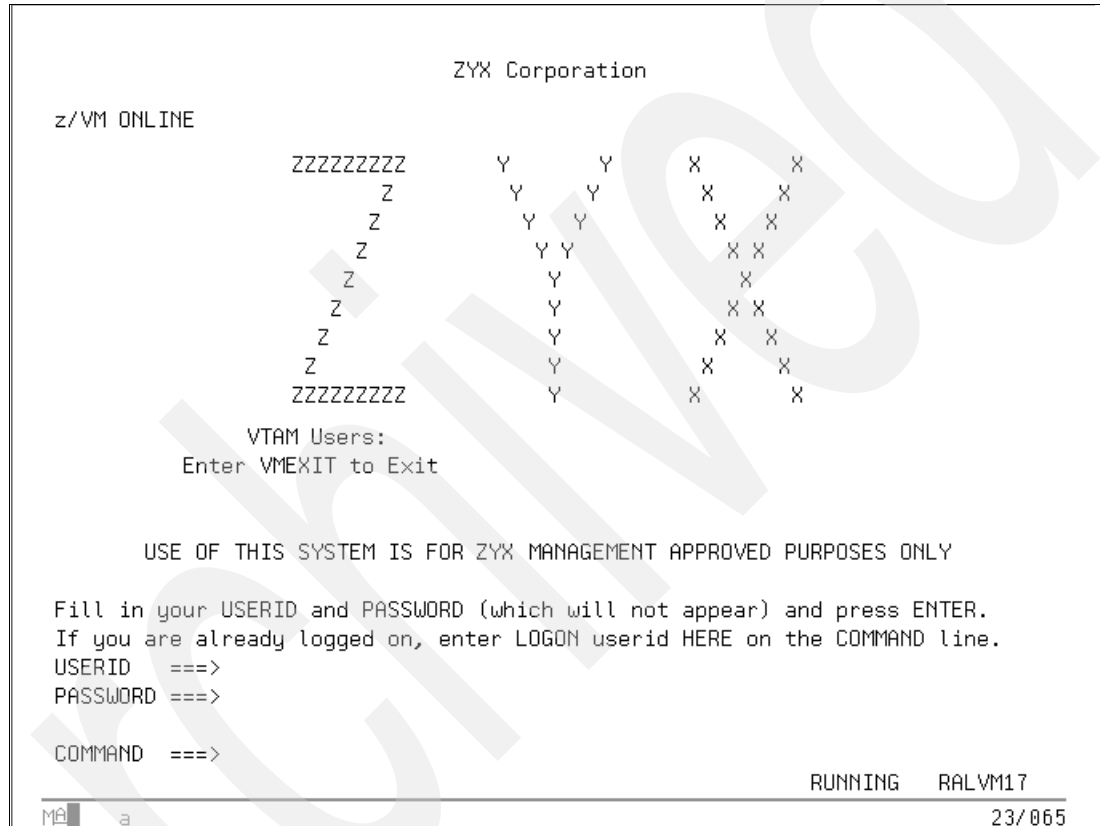


Figure 3-1 Logon screen

The challenges of security increase every day. Of course the primary concerns relate to protecting access to sensitive applications and data. It may also be desirable to provide functional components of applications to different constituencies (end-user groups). For example, in the case of ZYX Parts, the business would like to extend access to inventory information over the Internet. This data is currently only available to users granted full access to ZYX Parts, as only one level of authentication is available.

3.2.2 Starting the application

Following a successful logon, users are positioned at a standard 3270 screen. Users then enter the name of the application they would like to run. For new users, this is perhaps the most intimidating screen. It provides no navigational assistance and is of little value to most users. In this example we start the ZYX Parts application by entering `zyxparts` and pressing Enter, as shown in Figure 3-2 on page 16.

```
ICH70001I LAST ACCESS AT 20:32:43 ON THURSDAY, SEPTEMBER 30, 2004
z/VM Version 4 Release 4.0, Service Level 0402 (64-bit),
built on IBM Virtualization Technology
There is no logmsg data
FILES:  NO RDR,  NO PRT,  NO PUN
LOGON AT 20:50:29 EDT THURSDAY 09/30/04
WWVM z/VM CMS 20 005 05/20/04
Ready; T=0.09/0.10 20:50:29

zyxparts_
MA a
RUNNING RALVM17
23/001
```

Figure 3-2 VM command screen

3.2.3 ZYX Parts main menu

The ZYX Parts Main Menu, as shown in Figure 3-3 on page 17, is a standard 3270 application screen. It provides function keys, an ordinal menu with descriptions, and a command line. There are a number of options on this screen, but users generally use only option 1. Option 1 is the business function we use in this example.

Basic help is provided by selecting the F1 key. The help is limited to static text; in some cases, it assists by providing valid values for input fields.

To continue, the user enters a 1 in the entry field in row 5 and presses Enter.

ZYX Parts Current Order Form

Type in Product numbers, pressing Enter after each entry.

Press the Search key to search for unknown product numbers

or, press the Catalog key to view the product catalog.

When your order is complete, press Enter to submit.

\$ Cap: 0

\$ Exp: 0

Lines: 0

More: +

Qty	Product Num	Use	Description	Estimated \$ Each
—	_____	—		
—	_____	—		
—	_____	—		
—	_____	—		
—	_____	—		
—	_____	—		
—	_____	—		
—	_____	—		

Current order is empty. Press GetOrder key to retrieve a previous order.

Command ==> _____

F1=Help
F2=Set 2
F3=Exit
F4=Prompt
F5=GetOrder
F6=Details

F7=Backward
F8=Forward
F9=Search
F10=Catalog
F11=Justify
F12=Cancel

MA a
13/002

Figure 3-4 Order screen

Legacy 3270 applications are predominantly navigated by using function keys. The function key panel at the bottom of the screen is essentially the “steering wheel” for the application. In this example we will be performing a part search. This function is initiated by pressing F9.

3.2.5 Searching ZYX Parts

There are a number of 3270 screen sizes in production today, and most of them operate with 24 rows and 80 columns, as in this example. Today’s high resolution workstations provide significantly more “real estate” by comparison. On the ZYX Parts Search page, as shown in Figure 3-5 on page 19, you can see that the screen is full but only contains three application input fields.


```

ZYX Parts Search

Type search keywords, then press Enter.
Examples of keywords are: 386  mouse  8MB  25MHz

Search
Category . . . : All Products

with Keywords. . PC2100 memory_____

NOTE: For most searches you do need not change default search criteria.
Criteria . . . . 3  1. Product description only
                  2. Product number only
                  3. Product number, description, and price

You may specify this catalog search to be limited to
Criteria . . . . 1  1. Products which are currently available
                  2. All products, including Withdrawn or
                   NSI (Not Shipped to Internal customers)

Command ==> _____
F1=Help      F3=Exit      F12=Cancel

MA  a 09/034

```

Figure 3-5 Search screen

The input fields predominantly used on this screen are Search or Search with Keywords. The other fields are generally left at their default values. In our example we will search for PC2100 computer memory. We enter this into the Keywords field and press Enter to continue.

3.2.6 Search results

The search results are displayed in the Parts Products screen as shown in Figure 3-6 on page 20. This panel provides part information including the part number, description and availability. To get more detail about a product the end user must navigate to the selection field preceding the product number and enter a forward slash (/), and then press Enter or the F6 key. Since the display is limited by 80 columns, a Long function is provided to shift the description laterally so that it can be read in its entirety.

ZYX Parts Products			
Type a "/" to select a product, then press Enter to order. To return to current order form, press Exit key.		Products 1 To 14 of 17	
		More: +	
		ESTIMATED	
Product Num	Description	Price	Avail
. L50L50	1GB PC2100 CL2.5 ECC DDR SDRAM DIMM Memory for	1	1-3WK
. P20P20	1GB PC2100 CL2.5 ECC DDR SDRAM RDIMM Memory	2	4-6WK
. K00K00	1GB PC2100 CL2.5 ECC DDR SDRAM UDIMM Memory	5	4-6WK
. L50L50	128MB PC2100 CL2.5 ECC DDR SDRAM DIMM Memory	0	4-6WK
. K00K00	128MB PC2100 CL2.5 NP DDR SDRAM SODIMM Memory	9	1-3WK
. L33L33	128MB PC2100 CL2.5 NP DDR SDRAM UDIMM Memory	9	1-3WK
. L50L50	2GB PC2100 CL2.5 ECC DDR SDRAM DIMM Memory for	1	4-6WK
. P20P20	2GB PC2100 CL2.5 ECC DDR SDRAM RDIMM Memory	2	4-6WK
. L50L50	256MB PC2100 CL2.5 ECC DDR SDRAM DIMM Memory	9	4-6WK
. K00K00	256MB PC2100 CL2.5 ECC DDR SDRAM UDIMM Memory	2	4-6WK
. K00K00	256MB PC2100 CL2.5 NP DDR SDRAM SODIMM Memory	5	1-3WK
. L33L33	256MB PC2100 CL2.5 NP DDR SDRAM UDIMM Memory	5	1-3WK
Products are displayed alphabetically by description.			
Command ==>			
F1=Help	F3=Exit	F4=Sort Cat	F5=SortDesc
F6=ProdInfo	F7=Backward	F8=Forward	F9=Long
F10=Sort Num	F12=Cancel		
MA a		09/002	

Figure 3-6 Search results screen

The screen is also limited by the number of records that can be displayed. The common solution to this issue is to virtualize *pages* by displaying sets of records using the same screen layout. The ZYX Parts Products screen provides information to users regarding their position in a record set similar to most 3270 applications. The record identifier, number of records, and expression of location using more or less indicators is common to 3270 applications.

Pressing F8=Forward on the ZYX Parts Products screen is equivalent to the page down functionality found in most applications, as shown in Figure 3-7 on page 21. In our example, we selected search criteria that resulted in only two pages of data. Other search criteria could yield significantly more records.

ZYX Parts Products			
Type a "/" to select a product, then press Enter to order. To return to current order form, press Exit key.			
		Products 14 To 17 of 17	
		More: -	
		ESTIMATED	
Product Num	Description	Price	Avail
. P20P20	512MB PC2100 CL2.5 ECC DDR SDRAM RDIMM Memory	0	4-6WK
. K00K00	512MB PC2100 CL2.5 ECC DDR SDRAM UDIMM Memory	0	4-6WK
. K00K00	512MB PC2100 CL2.5 NP DDR SDRAM SODIMM Memory	2	1-3WK
. L33L33	512MB PC2100 CL2.5 NP DDR SDRAM UDIMM Memory	7	1-3WK
Command ==>			
F1=Help	F3=Exit	F4=Sort Cat	F5=SortDesc
F6=ProdInfo	F7=Backward	F8=Forward	F9=Long
F10=Sort Num	F12=Cancel		
MA a		22/015	

Figure 3-7 Search results screen - next page

There are a number of indicators showing that we have reached the last row in our data set. The most obvious is the lack of records populating the screen. Another indicator is located next to the More: field in the upper right corner of the screen.

This simple example represents a standard 3270 application and a common business transaction. In the chapters that follow we demonstrate how to enhance this business process, without modifying the legacy application source, to create a completely new end-user experience.

Archived

3270 applications using HATS

In this part we create a default and customized Host Access Transformation Services (HATS) application. The default application demonstrates the ability of HATS to dynamically transform legacy screens to HTML and display them in a browser. The customized application illustrates some of the more common improvements that can be made to legacy applications using HATS.

Archived

Creating a default HATS application

This chapter provides an introduction to creating a Host Access Transformation Services (HATS) application for ZYX Parts. The concepts presented here are basic. After HATS is installed, base applications can be created in minutes.

Note: The examples in this book use WebSphere Studio Application Developer Version 5.1.2 and Host Access Transformation Services v5.0.4.

- ▶ Introducing HATS wizards and editors
- ▶ Creating a base HATS project
- ▶ Configuring the communication settings
- ▶ Choosing a default template

4.1 HATS wizards and editors

Extending legacy applications to the Web requires a variety of different skill sets. End users are experts in application navigation, but rarely have Java development skills. Java developers may not be seasoned Web page designers. HATS combines easy-to-use wizards with the industry's leading Java integrated development environment, WebSphere Studio. This allows someone without advanced skills in any of these disciplines to create HATS applications, while leveraging a strategic platform.

The welcome page presented in WebSphere Studio following the installation of HATS, as shown in Figure 4-1, provides an outline of wizards, editors and tutorials that will sequentially guide you through the creation and customization of your first project. As your skills develop, you will discover short-cuts and fast paths to many HATS functions. Many of these tricks can be learned from the “tips” that are presented by default after installation.

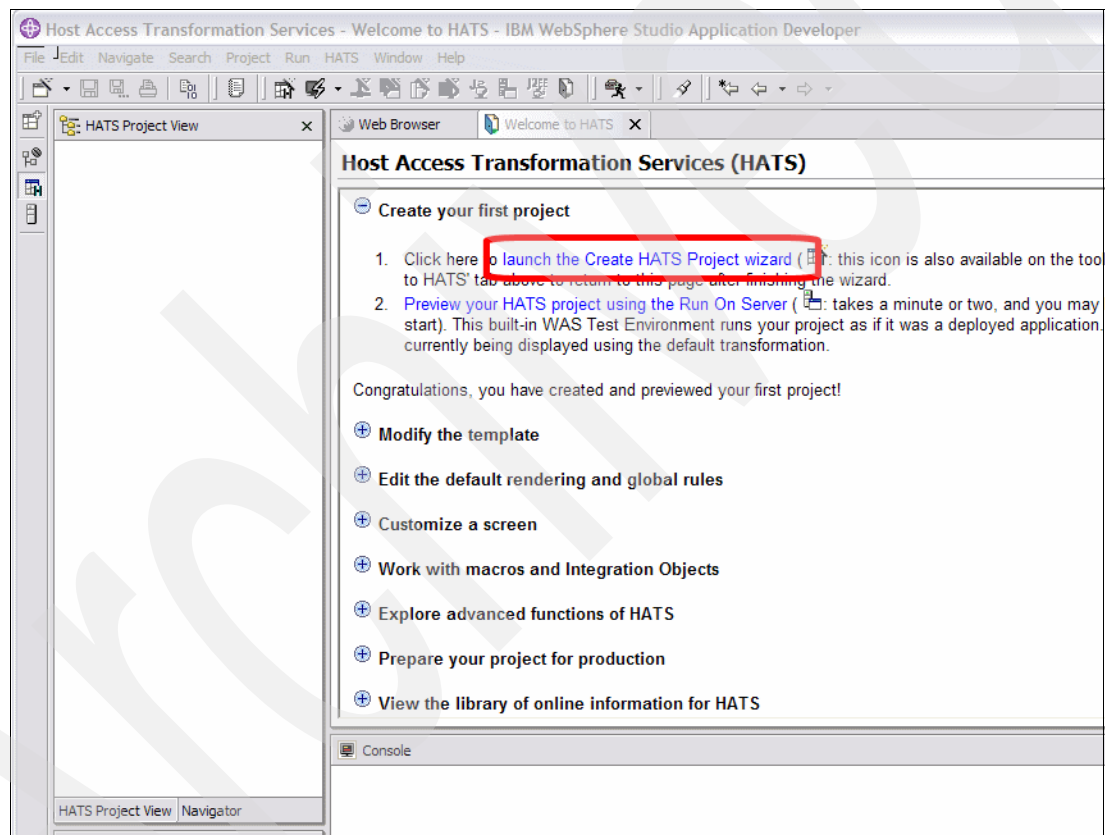


Figure 4-1 Welcome to HATS panel

4.2 Creating a HATS project

The short-cut can be found on the tool bar:



The first step in building a HATS application is creating the project. This can be done a number of ways but it is easiest to start with the link found in the welcome page. Alternatively, you could select **HATS -> New -> Project**.

Note: The HATS menu option is only available in the HATS perspective.

This will bring up the Create a Project panel, as shown in Figure 4-2. Enter the name of the HATS project and then, optionally, a description. To change the default location of the project, uncheck the use default location check box. The location field and browse buttons are then enabled and you can specify a path manually.

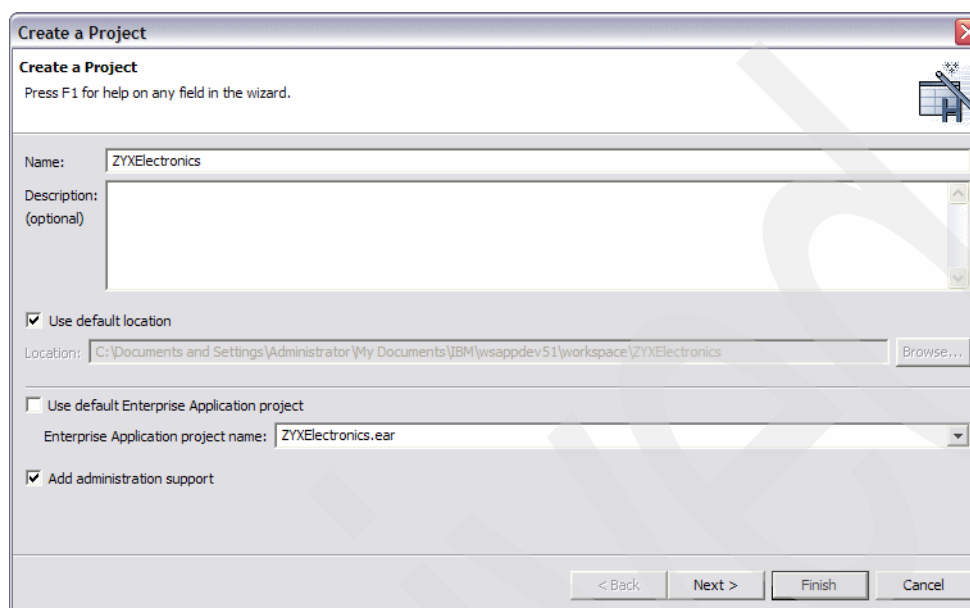


Figure 4-2 Create a Project panel

In this example we will uncheck the use default enterprise application project box and change the default name from HATS.ear to ZYXElectronics.ear. It is good practice to provide a unique ear file for each project and to keep the ear file name and the project name the same. Each ear file is approximately 16 M in size. The name of the ear file is important, since it is used to identify the HATS application when running on WebSphere Application Server.

Note: In HATS Studio, you cannot choose a different ear file once the project is created. However, you can move projects from one ear file to another, using the WebSphere Studio application.xml editor.

HATS provides a number of administrator functions for the purposes of connection management, tracing and logging. These functions are new to HATS Version 5. To enable these features, make sure the check box for adding administrative support is checked. This is the default and is recommended, as it provides useful debugging capabilities. When this panel is complete, click **Next >**.

4.2.1 Configure connection settings

HATS applications require a telnet connection to provide access to the legacy system. The first entry field on this panel requires the host name and port of the telnet gateway that can be used to access the legacy system, as shown in Figure 4-3 on page 28. This can be found in your current emulator settings, or by contacting your system administrator. The IP address of the telnet gateway can also be entered here, but the domain name is preferable. The traditional port used by 3270 devices is Port 23.

Create a Project

Connection Settings
Configure host connection settings for this new project.

You can specify advanced connection values later in the connection editor.

Host name:

Port:

Type:

Code page:

Screen size:

< Back Next > Finish Cancel

Figure 4-3 Configure Connection Settings panel

The two protocols for accessing zSeries applications are TN3270 and TN3270E. If the type is unknown, it is recommended that you select TN3270E, as it can sub-negotiate a TN3270 connection. This will allow the HATS application to take advantage of the improvements implemented in RFC 2355, as discussed in “3270 emulation primer” on page 14.

Select the appropriate code page and screen size, then click **Next >**.

4.2.2 Selecting the default template

The final step in creating a HATS project is selecting the default transformation template. The templates are JSPs that are used to create the aesthetic canvas for the display of your 3270 application. The product comes with a number of templates, but also allows you to create your own. Further customization can be done after the project is created.

Note: HATS ships with a Classic Terminal template. The use of this theme will render 3270 applications in a browser with the same look and feel of a 3270 terminal.

The theme selection is done using the template pull-down box. A large preview window is provided to assist in the template selection, as shown in Figure 4-4 on page 29. When you click Finish, the HATS Project Wizard will create the HATS project.

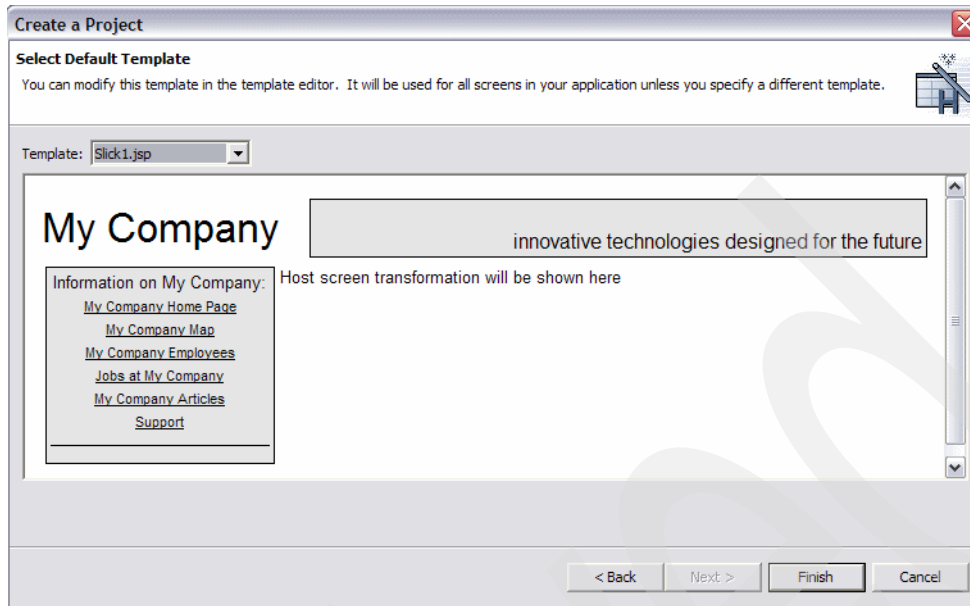


Figure 4-4 Select Default Template panel

The default HATS application created is functional and ready to be run. While it is desirable to customize your HATS application, it is exciting to see how fast legacy systems can be enabled using the tool. This can be done using the next item in the HATS Welcome panel: Preview your HATS project using the Run On Server, as shown in Figure 4-5. Refer to Chapter 9, “HATS and WebSphere for z/OS” on page 77, for more information.

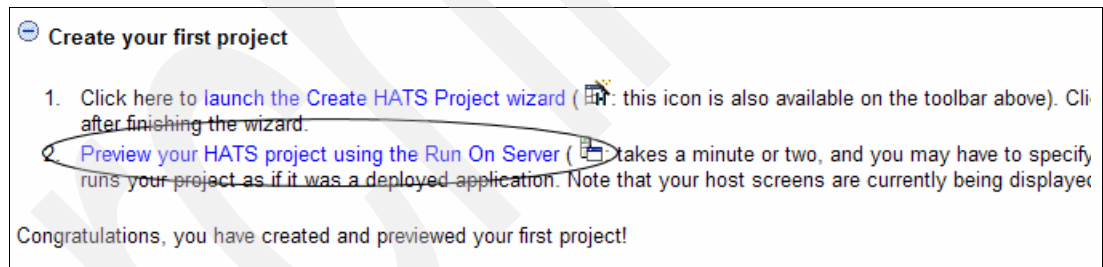


Figure 4-5 Preview your HATS project

Archived

Default HATS application

Now that we have created our base application, we will examine the results. In this chapter we run the base application created in Chapter 4. It is important to keep in mind that no customization has taken place yet; that task will be performed in Chapter 6, “Creating a customized HATS application” on page 39.

Here, we examine the default Host Access Transformation Services (HATS) transformations of the ZYX Parts application.

5.1 ZYX Parts with default transformation

The purpose of reviewing the default transformations is to set expectations for the functional baseline of HATS. The default transformation functionality of HATS provides a number of benefits; perhaps the most significant is the ability to enable legacy applications to be delivered via a browser in a period of minutes. Rebuilding legacy systems with new technologies is expensive and creates risk. By using the HATS default transformation, you can save time and money and leverage your proven legacy applications.

The default transformation feature of HATS reduces risk in a number of ways. By applying a consistent look and feel to all legacy screens, enterprise modernization can be done in bite-size chunks. This means that IT departments can develop at a rate sustainable by resources and budget. In addition, if project plans change, the default HATS transformation can still provide a customized appearance.

Lastly, there is often a risk that a screen or business function from the legacy system will be omitted from the new application. If HATS developers overlook an application or screen, it will still be rendered using the HATS default transformation.

The ZYX Corporation logon screen looks very similar when displayed using the default transformation as shown in Figure 5-1. A HATS transformation is used to define how legacy components will be rendered in the browser. By default, all of the legacy screen assets are displayed. The HATS default template provides the elements surrounding the host transformation, including graphics, URL links and buttons. We review these items in more detail in Chapter 6, “Creating a customized HATS application” on page 39.

My Company

innovative technologies designed for the future

Information on My Company:
[My Company Home Page](#)
[My Company Map](#)
[My Company Employees](#)
[Jobs at My Company](#)
[My Company Articles](#)
[Support](#)

Reset
Default
Refresh
Disconnect
Turn Keyboard Off

z/VM ONLINE

WELCOME TO IBM GLOBAL SERVICES

RRR RESEARCH TRIANGLE PARK

R R

RRR III

R R I INFORMATION

R R I

R R I SSS

III S SYSTEMS

SSS

SSS

SSS

VTAM Users:

Enter VMEXIT to Exit

Customer Service Center: 888-IBM-HELP

USE OF THIS SYSTEM IS FOR IBM MANAGEMENT APPROVED PURPOSES ONLY

Fill in your USERID and PASSWORD (which will not appear) and press ENTER.
If you are already logged on, enter LOGON userid HERE on the COMMAND line.

USERID ==>

PASSWORD ==>

COMMAND ==>

RUNNING RALVM17

Figure 5-1 Logon screen

From this screen we can log in just as we would using a terminal or emulation program.

Once again we are at a 3270 screen, as shown in Figure 5-2 on page 33, that provides little guidance for the end user. Without adequate training or documentation, users struggle with what to do next.

This screen is used to launch a variety of applications, based on the user group. While system commands are available, they are generally unused and their functionality can be obfuscated.

```
ICH70001I LAST ACCESS AT 20:50:29 ON THURSDAY, SEPTEMBER 30, 2004
z/VM Version 4 Release 4.0, Service Level 0402 (64-bit),
built on IBM Virtualization Technology
There is no logmsg data
FILES:  NO RDR,  NO PRT,  NO PUN
LOGON AT 01:03:36 EDT FRIDAY 10/01/04
WWVM z/VM CMS 20 005 05/20/04
Ready; T=0.09/0.11 01:03:36
```

RUNNING RALVM17

Figure 5-2 VM command screen

The default HATS screens have the same geographic layout as the legacy screens, and look very much the same using the default transformations. On the ZYX Parts Main Menu screen, shown in Figure 5-3 on page 34, we can see in greater detail some of the default rendering options that HATS provides. Here the menu screen component of the application has been converted to links.

Note:

Be aware of the following terminology:

- *Components* are the legacy assets that exist on the 3270 screens (for example, command lines, input fields, function keys, and so on).
- *Widgets* are the GUI assets that are used to render the legacy components (for example, buttons, calendars, check boxes, drop-down boxes, and so on).

ZYX Parts Main Menu

Type the number of your choice, then press Enter.

1	1. Shop	- Browse or order from catalog, or retrieve saved orders
	2. Inquire	Status of approved orders (option to cancel & escalate)
	3. Nominate	Nominate a product for the catalog
	4. Feedback	Send comments, suggestions, questions
	5. Information	ZYX Parts bulletin boards, services
	6. Start here	An overview of ZYX Parts

Command ==>

[F1=Help](#) [F3=Exit](#) [F12=Cancel](#)

Figure 5-3 Main menu

To continue in our example, we click the **Shop** link (or enter menu option 1 into the input field), then press Enter.

The ZYX Parts order screen is the most utilized screen in the application. Here are found the greatest number of function keys and input fields, as shown in Figure 5-4 on page 35. The fields are rendered as GUI input boxes—which is not only more aesthetically pleasing, but also makes it much easier for users to identify the input fields.

The input boxes can be used in the same functional manner as the input fields provided by terminal emulation. Users can tab or use the mouse to navigate between fields.

ZYX Parts Current Order Form

Type in Product numbers, pressing **Enter** after each entry.

Press the **Search key** to search for unknown product numbers

or, press the **Catalog key** to view the product catalog.

When your order is complete, **press Enter** to submit.

\$ Cap: 0

\$ Exp: 0

Lines: 0

More: +

Qty	Product Num	Use	Description	Estimated \$ Each
<input type="text"/>	<input type="text"/>	<input type="text"/>		
<input type="text"/>	<input type="text"/>	<input type="text"/>		
<input type="text"/>	<input type="text"/>	<input type="text"/>		
<input type="text"/>	<input type="text"/>	<input type="text"/>		
<input type="text"/>	<input type="text"/>	<input type="text"/>		
<input type="text"/>	<input type="text"/>	<input type="text"/>		
<input type="text"/>	<input type="text"/>	<input type="text"/>		
<input type="text"/>	<input type="text"/>	<input type="text"/>		
<input type="text"/>	<input type="text"/>	<input type="text"/>		
<input type="text"/>	<input type="text"/>	<input type="text"/>		

Current order is empty. Press **GetOrder** key to retrieve a previous order.

Command ===>

[F1=Help](#)

[F7=Backward](#)

[F2=Set 2](#)

[F8=Forward](#)

[F3=Exit](#)

[F9=Search](#)

[F4=Prompt](#)

[F10=Catalog](#)

[F5=GetOrder](#)

[F11=Justify](#)

[F6=Details](#)

[F12=Cancel](#)

Figure 5-4 Order screen

Another dramatic improvement can be seen with the rendering of the function keys. The default transformation functionality of HATS automatically converts function keys to a number of GUI-based components. In our example, the function keys have automatically been converted to links.

Note: HATS provides function key support via the browser. This feature is configurable and can be customized to allow remapping.

Review the ZYX Parts Search screen, as shown in Figure 5-5 on page 36, and the ZYX Parts Products screen as shown Figure 5-6 on page 36 and Figure 5-7 on page 37. Note that they provide the same functionality as with an emulator, but offer functional and aesthetic improvements.

ZYX Parts Products

Type a "/" to select a product, then press Enter to order. To return to current order form, press Exit key.

Products 14 To 17 of 17
More: -

Product Num	Description	ESTIMATED Price Avail
<input type="checkbox"/> P20P20	512MB PC2100 CL2.5 ECC DDR SDRAM RDIMM Memory	0 4-6WK
<input type="checkbox"/> K00K00	512MB PC2100 CL2.5 ECC DDR SDRAM UDIMM Memory	0 4-6WK
<input type="checkbox"/> K00K00	512MB PC2100 CL2.5 NP DDR SDRAM SODIMM Memory	2 1-3WK
<input type="checkbox"/> L33L33	512MB PC2100 CL2.5 NP DDR SDRAM UDIMM Memory	7 1-3WK

Command ==>

F1=Help
F8=Forward
F3=Exit
F9=Long
F4=Sort Cat
F10=Sort Num
F5=SortDesc
F12=Cancel
F6=ProdInfo
F7=Backward

Figure 5-7 Search results screen

Archived

Creating a customized HATS application

In this chapter we review some common customizations that can be done with Host Access Transformation Services (HATS). This is not a tutorial, but rather an overview; refer to *HATS Programmers Guide* and *HATS User's and Administrator's Guide* for more detailed information. No Java coding is required for the customizations that are done in this chapter.

In our scenario, the ZYX Corporation would like to extend access to the ZYX Parts application to a new set of business partners. This will allow them to reach into new markets and become more competitive. They would like to create an interface that can be deployed over the Internet and can be integrated into a corporate portal.

In order to present the level of work and skill required to customize a legacy 3270 application, we perform the following tasks with the ZYX Parts application:

- ▶ Modifying the default transformation template
- ▶ Customizing the logon screen
- ▶ Consolidating screens
- ▶ Customizing the main menu screen
- ▶ Customizing the parts order screen
- ▶ Customizing the parts search results screen

6.1 Modifying the default template

The default template, as we discussed earlier, is used to customize the generic appearance of the legacy application. It is applied to all screens, with the exception of screens that are customized. The template is a Java Server Page (JSP) that uses Cascading Style Sheets (CSS).

Templates can be created in a number of ways:

- ▶ Using the Create a Template wizard found in the Welcome panel, as shown in Figure 6-1
- ▶ Opening the default template from the Welcome panel
- ▶ Opening the template marked default in the HATS Project View

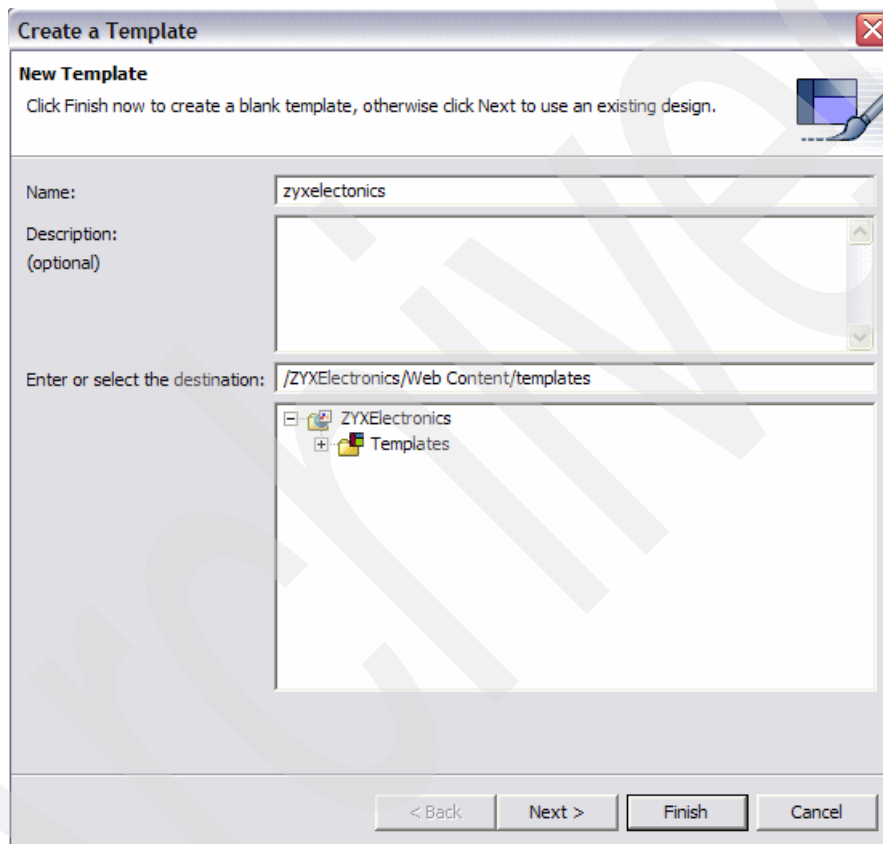


Figure 6-1 New Template panel

The first step in creating the template is to select a source. To model your template using a provided template, use the option Prefill the template from an existing project template, shown in Figure 6-2 on page 41.

If you have an existing Web site that you want to model your template with, select the option Prefill from an existing Web Page (URL or file). If you want to design a new template, select the option Create a blank template.

In this example we will be modeling this template after slick1.jsp, which is provided with HATS. After selecting the source option, click Finish.

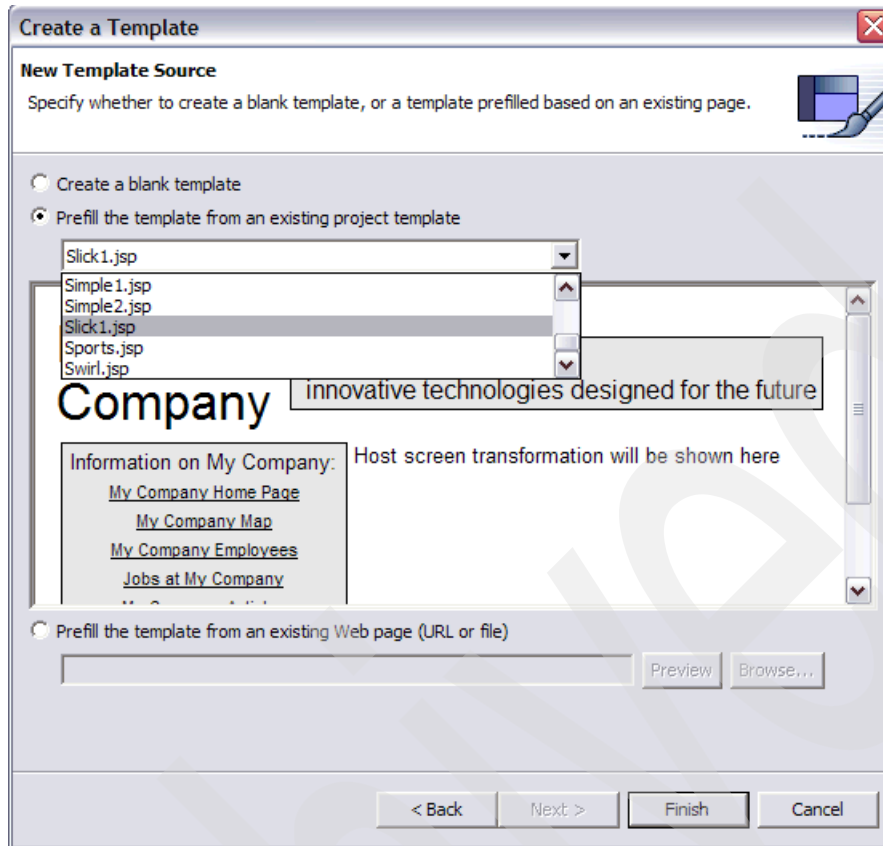


Figure 6-2 New Template Source panel

The new template is created and then displayed using the WebSphere Studio JSP editor. For more information on using the WebSphere Studio JSP Editor, refer to the IBM WebSphere Studio Application Developer InfoCenter:

<http://publib.boulder.ibm.com/infocenter/ad51help>

In this example we modified the company name, added a graphic, changed the URL links and changed the banner text, as shown in Figure 6-3 on page 42. These are all simple changes to make in the JSP editor. After customizing the page, save the template using **File -> Save**.

Tip: In WebSphere Studio, an asterisk (*) next to the filename indicated in the tab means that changes were made to the file. You can use Ctrl + S as a shortcut for saving changes.

Note: If your application is running, you can simply click the browser refresh button and the changes will be applied dynamically. This is just one example of how the WebSphere Studio integrated development environment can dramatically improve programmer productivity.

The customized template for this example can be seen by selecting the Refresh button, as shown in Figure 6-3 on page 42.

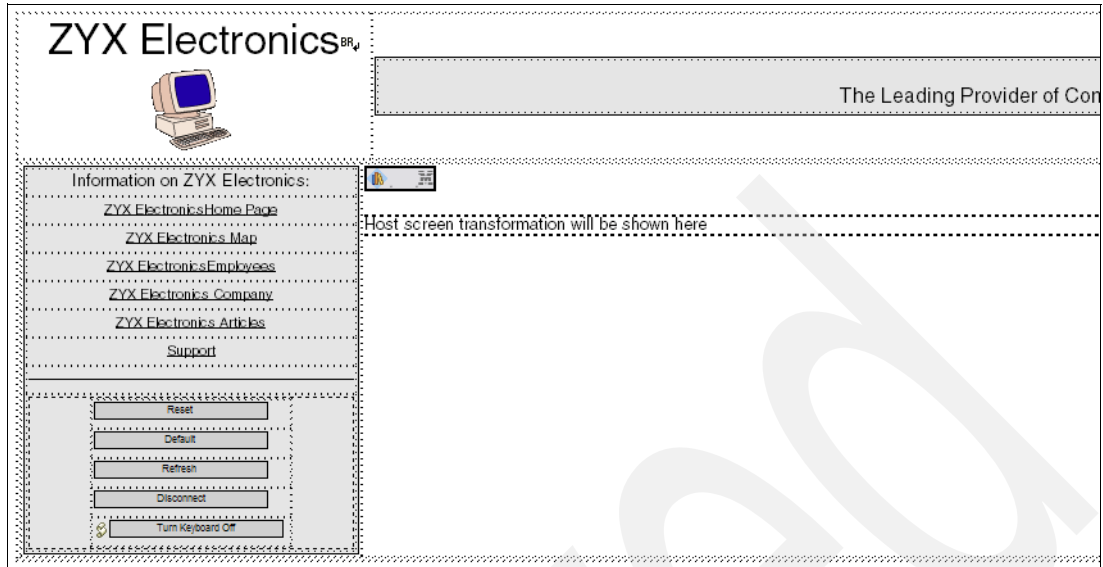


Figure 6-3 ZYX Parts default template

6.2 Customizing the default rendering options

The default rendering panel provides a prioritized list of components. The placement of the rendered components is based on their respective position in the list, as shown in Figure 6-4. It is a good practice to leave the Transform remaining text and input fields last.

Transformations can be enabled/disabled, configured, and added from this panel. Editing is done by selecting the transformation, then the Edit button. A screen capture is required to continue using the component customization wizard. Use the mouse to draw a “rubber band” around the screen region that will contain the components. Then use the selection panel to configure the component to widget mapping. In this example, we do not modify any of the default rendering options.

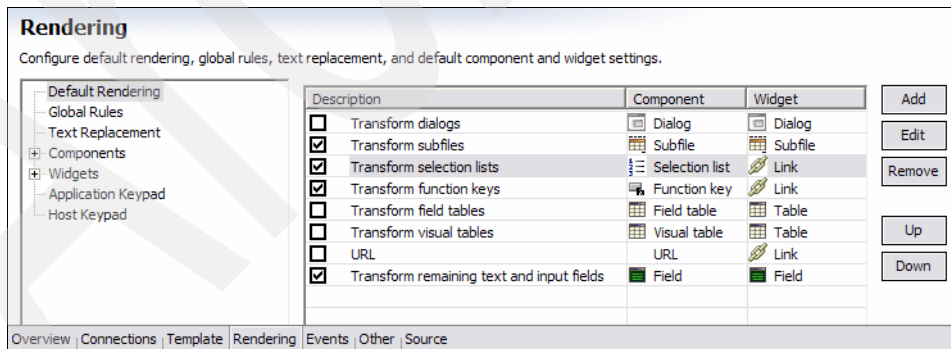


Figure 6-4 Default rendering configuration panel

6.3 Customizing the logon screen

Screens are customized in HATS by using the screen customization wizard. This tool will guide you through the steps required to customize a screen. The project components created are the *screen event*, *screen capture* and *transformation*.

The screen event component contains the following:

- ▶ Screen recognition criteria
- ▶ Actions that can be taken include:
 - Apply transformation
 - Execute business logic
 - Extract data from screen
 - Send data to screen
 - Set global variables
 - Show URL
 - Forward to URL
 - Play a macro
 - Perform a macro transaction
 - Send a key
 - Disconnect
- ▶ Text replacement

The screen customizing wizard can be launched from the host terminal window. First select the project, then select **HATS -> Open HATS Host Terminal**. From within the Host Terminal window, select the first icon on the tool bar and then screen customization, as shown in Figure 6-5.

The short-cut
can be found
on the tool bar:

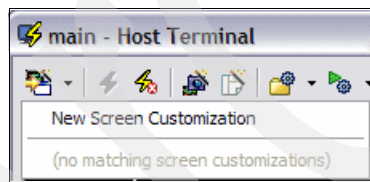


Figure 6-5 New Screen Customization option

This will bring up the Create Screen Customization panel, as shown in Figure 6-6 on page 44. Here is where you enter the name of the new customization.

Figure 6-6 Create Screen Customization panel

The Select Screen Recognition Criteria panel is then used to identify the host screen that is being customized. HATS provides a number of conditional alternatives which are used to uniquely identify screens, including the following:

- ▶ Total number of fields
- ▶ Number of input fields
- ▶ Row and position of cursor
- ▶ String text and position including
 - Anywhere on screen
 - At a specified row and column
 - Within a rectangular region

When this panel is completed, selecting Next will bring up the Select Actions window. For this example we kept the initial layout setting of blank, and selected **Finish**.

The panel that follows is used to select the legacy components that are to be rendered in the screen customization. Screen elements are selected by drawing a rubber band around the desired elements. Multiple components can be selected at once or individually by repeating the process. In this example we selected the userid and password fields, including the prompt text.

The final step in creating the screen customization is mapping the legacy component to a HATS widget. The Insert Host Component panel provides two selection windows and two preview windows, as shown in Figure 6-7 on page 45. The panel will only allow the selection of valid components and widget combinations based on the screen elements selected.

We selected the visual table component from the Insert Host Component window. A table provides a structured way to lay out fields on the screen. The only valid widget for this component selection is Table. After selecting the widget, we clicked **Finish**.

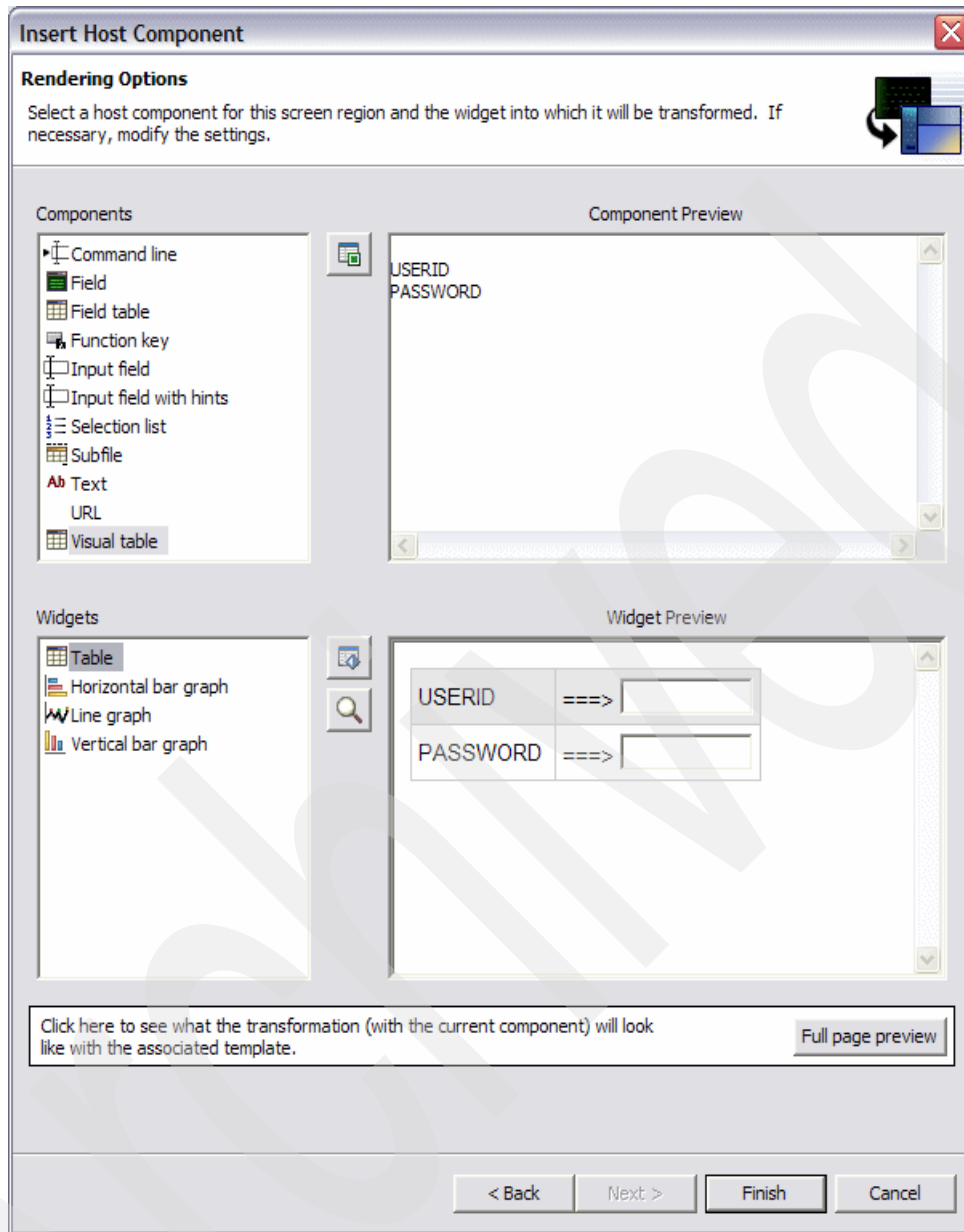


Figure 6-7 Rendering Options panel - Visual table

The wizard then automatically brings up the new transformation in the WebSphere Studio JSP editor. At this time the HATS developer can use the design or source view of the editor to further customize the page. New screen elements can be added by using the WebSphere studio features or the HATS Tools. The HATS Tools can be used to:

- ▶ Insert host components
- ▶ Edit host components
- ▶ Insert tabbed folders
- ▶ Insert key to play a macro
- ▶ Insert host keys
- ▶ Insert application keys
- ▶ Insert Host Integration Objects

In this example we added a host button to the logon screen by selecting **HATS Tools -> Insert Individual Host Key**. The Insert Host Key panel provides the ability to add keys that will be rendered with buttons on the transformation, as shown in Figure 6-8.

Note: HATS components must be added between the HATS form tags.

```
<!-- Start of the HATS form. -->
<HATS:Form>

<!-- Insert your HATS component tags here. -->

<HATS:HostKeypad />
</HATS:Form>
```

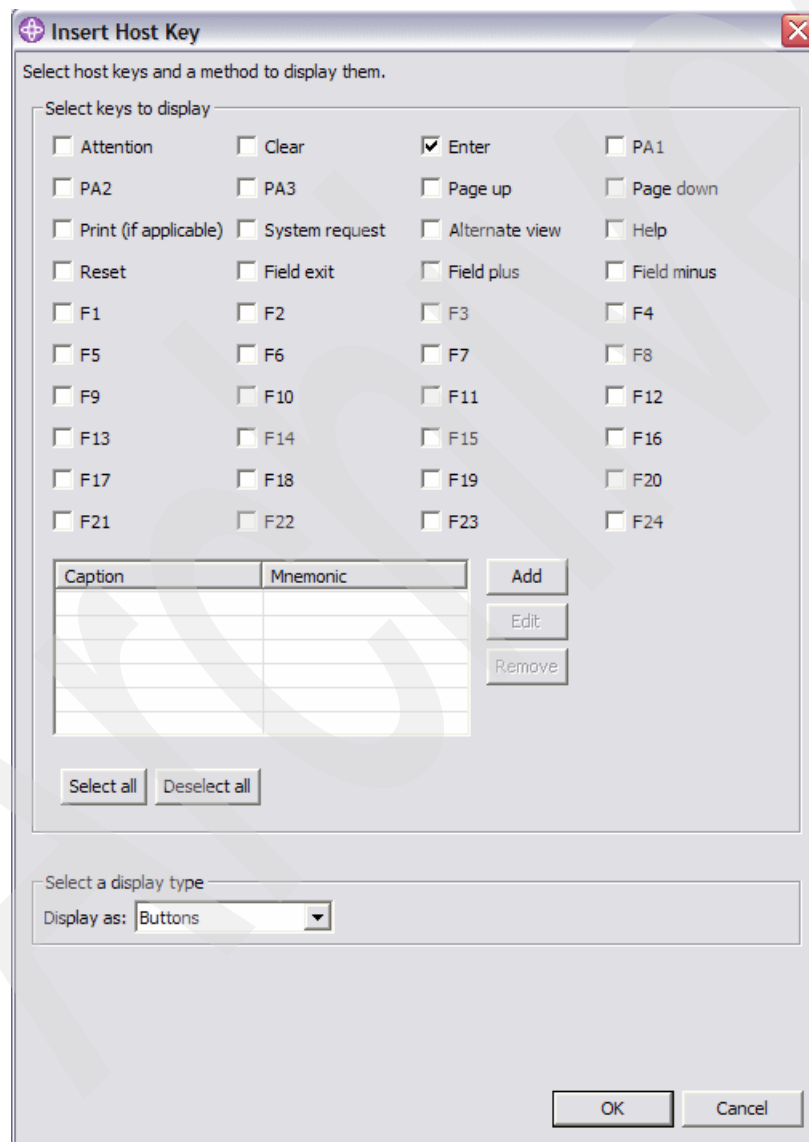


Figure 6-8 Insert Host Key panel

Function key names are often abbreviated because of the space limitation on the 3270 screen. HATS gives us the ability to customize the button name, color and other characteristics. In this example we changed the name from Enter to Logon.

The attributes can be changed by first selecting the button. The menu mouse button (right mouse button for right-handed users) will bring up a menu which contains an attributes option. Selecting this option will present the attributes configuration panel, as shown in Figure 6-9.

Figure 6-9 Button Attributes panel

The last formatting step for the logon screen is to horizontally align the screen elements that were added. This can be done by selecting these elements and then selecting **Format -> Align -> Horizontal Center**.

After editing the transformation is completed, save and exit the JSP editor—in less than 10 minutes the appearance of the ZYX Corporation logon screen has been dramatically improved! The changes can then be viewed using the WebSphere Studio Test Environment.

The short-cut
can be found
on the tool bar:



Notes:

- ▶ A new screen customization requires you to refresh the project on the server. This can be done in WSAD by selecting the project and the mouse menu button (right button for right-handed users). Then select Run on Server.
- ▶ The KBS.js file contains the key code mapping and can be found in the <project>/webApplication/common directory.

Tip: The cursor can be configured to auto advance to the next field when the current input field is completely filled in. This is enabled in the <project>/webApplication/common/lxgwfunctions.js file by changing the var autoAdvance = false statement to true and is only available for IE browsers.

6.4 Bypassing the command screen

Customizing the 3270 Command screen is initiated in a similar manner to the logon screen. The primary difference is the *action* that is applied to the screen customization event. Rather than applying a screen transformation, HATS will automatically enter the name of the application and “press Enter” using a macro.

In this example we use a macro called startzyxparts that sends the characters *zyxparts* to the screen followed by the Enter key (for more information on creating macros, refer to *Advanced Macro Guide*). We then create a screen customization and select the advanced check box, as shown in Figure 6-10.

Note: Since the screen will not be displayed, the check box Apply a transformation must be unchecked.

It may be necessary to bypass system messages or other unsolicited screens. These can be handled by providing a screen customization event for each type of screen. For example, a screen customization can be created for system messages that either displays, obfuscates, or ameliorates the message text.

Create a Screen Customization

Select Actions

These actions will be performed when a host screen matches the recognition criteria for this screen customization. You can also add or modify actions later using the screen customization editor.

☐ Apply a transformation

☒ Create new transformation

Initial Layout

☐ Prepopulated with host components

☒ Blank

☐ Use existing transformation

Template:

ZYX Electronics

The Lead

☒ Play a macro

Macro:

☒ Add advanced actions

Figure 6-10 Select Actions panel

6.5 Customizing the parts main menu

The ZYX Parts Main Menu screen contains a menu with six options. As usual, the first step for customizing a screen is using the screen customization wizard discussed in “Customizing the default rendering options” on page 42. After identifying the screen, we use the “rubber band” to identify the menu selection contents on the screen.

The next step is selecting the components and widgets that will be used to render the menu. In this example we will convert the legacy style selection list to buttons. This can be done using the selection list component that comes with HATS, as shown in Figure 6-11.

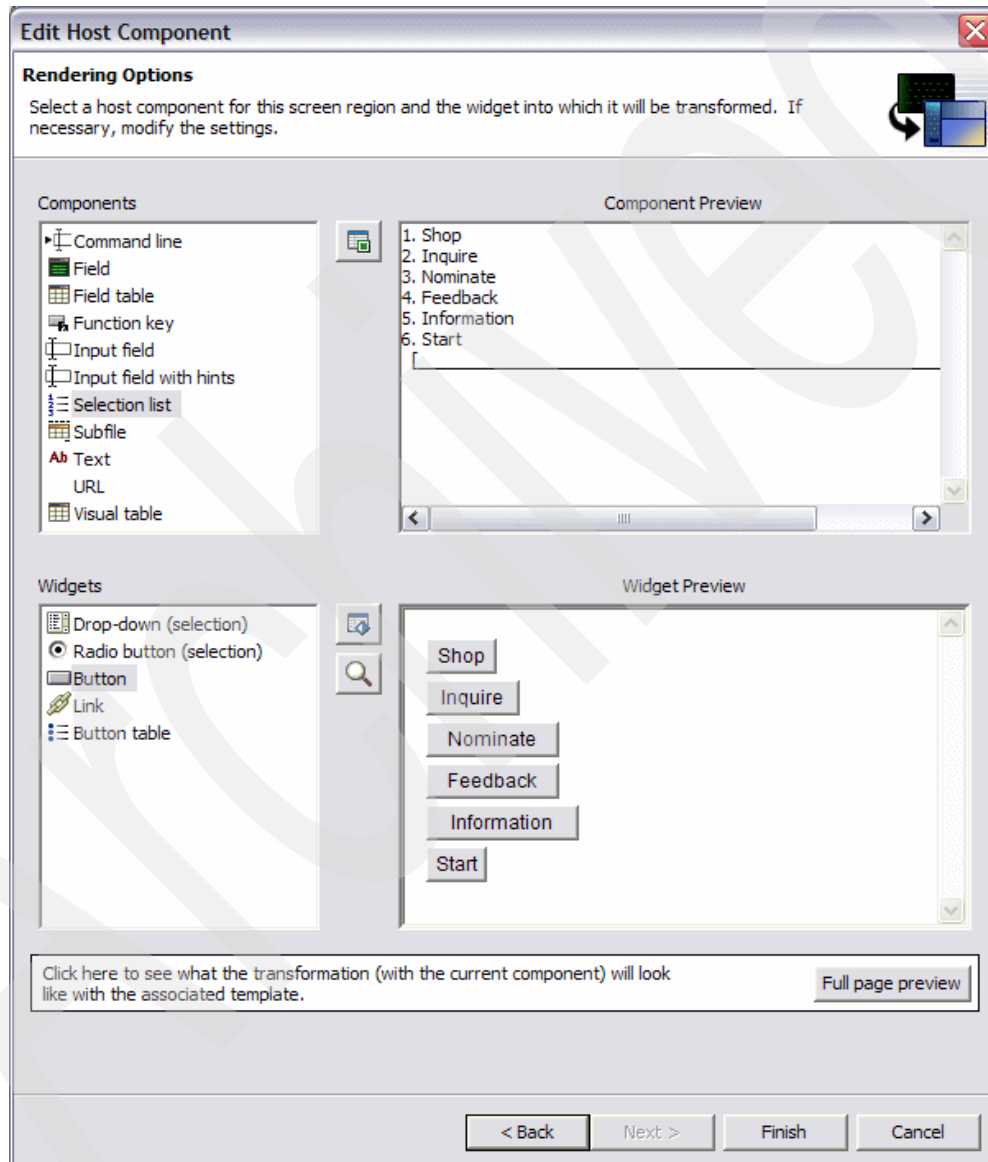


Figure 6-11 Rendering Options panel - Selection list

The short-cut
for the widget
settings:



Now is a good time to confirm the button style class. Selecting the widget settings icon brings up the widget settings pane, as shown in Figure 6-12 on page 50. The default button style class is HATSBUTTON. A new class can be specified, or the class can be changed.

The Button Settings panel is closed by selecting **OK**. The Edit Host Component panel is closed by selecting **Finish**.

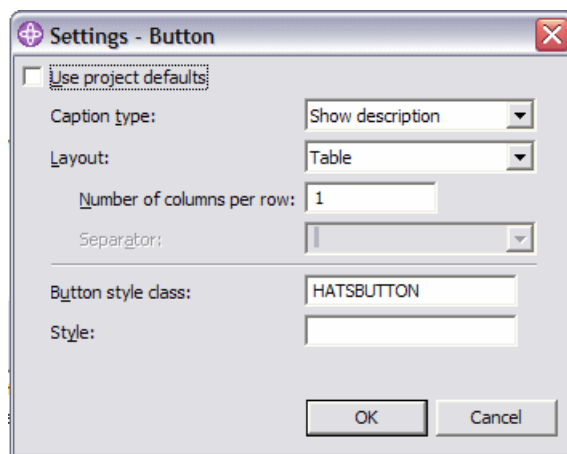


Figure 6-12 Button Settings panel

The buttons created by default have varying widths, based on the description length. It is often desirable to have buttons of the same width. This can be customized by using cascading stylesheets. The first step is identifying which cascading stylesheet to edit. This can be found in the template file, which in this example is `zyxelectronics.jsp`. Once opened, the template file source can be viewed in the WSAD JSP editor by selecting the source tab.

Stylesheets are applied in a cascading manner, with the last file listed having the highest precedence. Customizations can be made in any of the stylesheets. Example 6-1 shows the `slickWhiteBackground.css` file will be customized.

Example 6-1 Default template - `zyxelectronics.jsp`

```
<LINK rel="stylesheet" href="../../common/stylesheets/monochrometheme.css" type="text/css">
<LINK rel="stylesheet" href="../../common/stylesheets/slickWhiteBackground.css"
type="text/css">
</HEAD>
```

The stylesheet can be found in the HATS Project View panel under the **Project Name** -> **Web Content** -> **Web Content** -> **Common** -> **Stylesheets** folder. Open the file and WSAD will automatically start the style sheet editor. We then add (or modify) the width parameter for the Button style class `HATSBUTTON`.

There are a number of width units that can be used, but `em` and `px` are commonly used. The `em` tag provides width relative to font size. The `px` tag refers to pixels and is dependent on resolution, as shown in Figure 6-13 on page 51.

Tip: Pressing `Ctrl + Space Bar` in the WSAD editors provides code assistance. This will help with the class customization.

Note: HATS also uses cascading style sheets to map host attributes (such as color) to the JSP rendering.

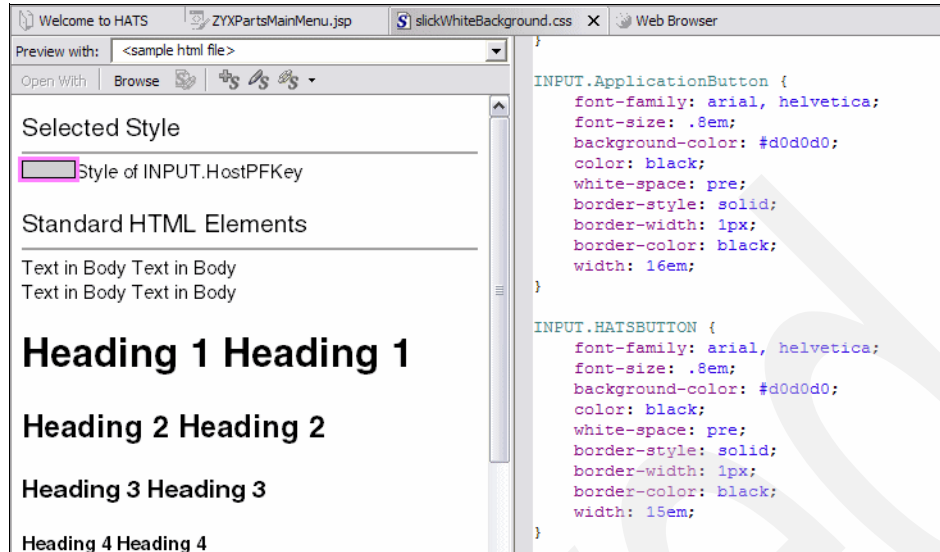


Figure 6-13 WSAD style sheet editor

The last step in customizing the logon screen is adding an exit button. This is done by selecting **HATS Tools -> Insert Individual Host Key** from the menu bar. The F3 function key is selected from the Insert Host Key panel. The button caption is then changed to Logoff.

6.6 Customizing the Parts Order Form

The only function of the Parts application order form screen that needs to be enabled is Search. This function provides product description, price and availability. Therefore, all other functions on the order form screen will be obfuscated from the user.

This is done in a manner similar to the way in which we handled the 3270 Command screen in section 2. In this case, however, rather than applying a screen transformation, we will automatically pass the F9 function key to initiate the search function.

When creating the screen customization for this screen, the check box `Apply transformation` should be unchecked, as we will not be displaying the screen. Only the `Add advanced` function check box should be selected.

On the subsequent panel, `Add` is selected to add a new action. The `Send key` action is selected from the `Select an Action` panel, as shown in Figure 6-14.

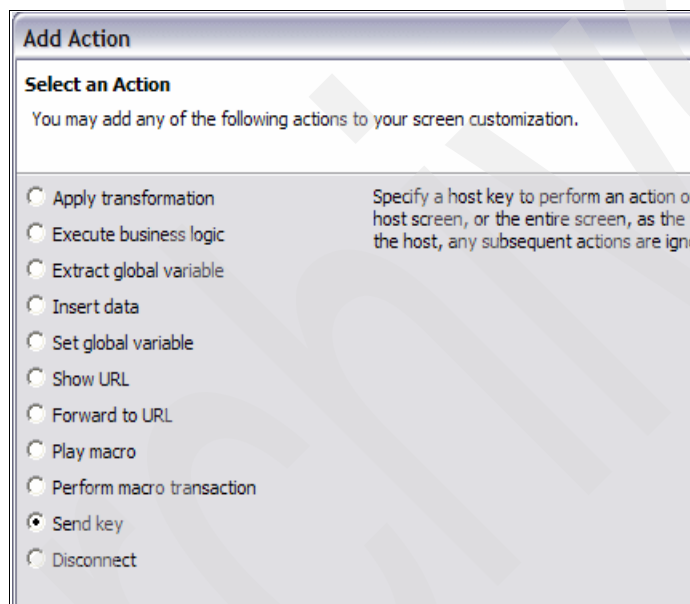


Figure 6-14 Add Action - Select an Action panel

The Define Action properties panel allows us to configure the key that will be automatically sent to the host when users reach the Order Form screen. Selecting PF and PA Keys will provide a list of valid function keys in the drop-down list box, as shown in Figure 6-15 on page 53.

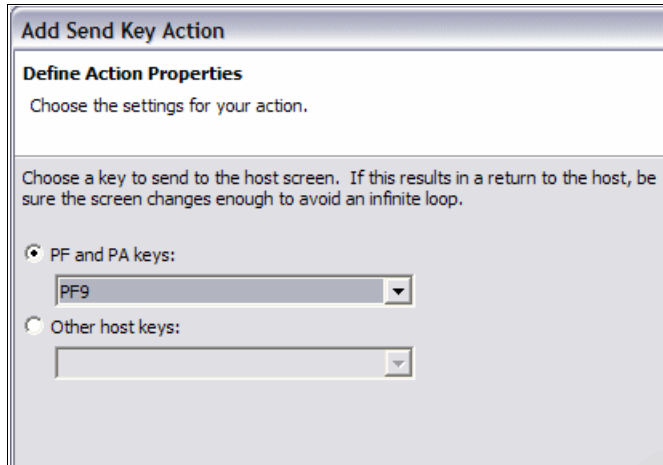


Figure 6-15 Add Send Key Action - Define Action Properties panel

These simple steps have resulted in a screen customization that will automatically initiate the search function when the Parts Order Form screen is reached. This reduces the number of screens the new business partners will have to traverse and prevents them from accessing other functions of the parts ordering application.

6.7 Customizing the Parts Search screen

The ZYX Parts Search screen provides the primary inputs to this work stream. The search keywords and criteria are critical components of this transaction. In this section we convert the legacy selection lists to pull-down lists, improving the usability and aesthetics of this screen.

In order to customize the Parts Search screen, we create a screen customization as in the previous sections. On the rendering options panel, the Selection list component is selected. In the Widgets list the drop-down widget is selected. Optionally, the widget can be tested in the widget preview window, as shown in Figure 6-16 on page 54.

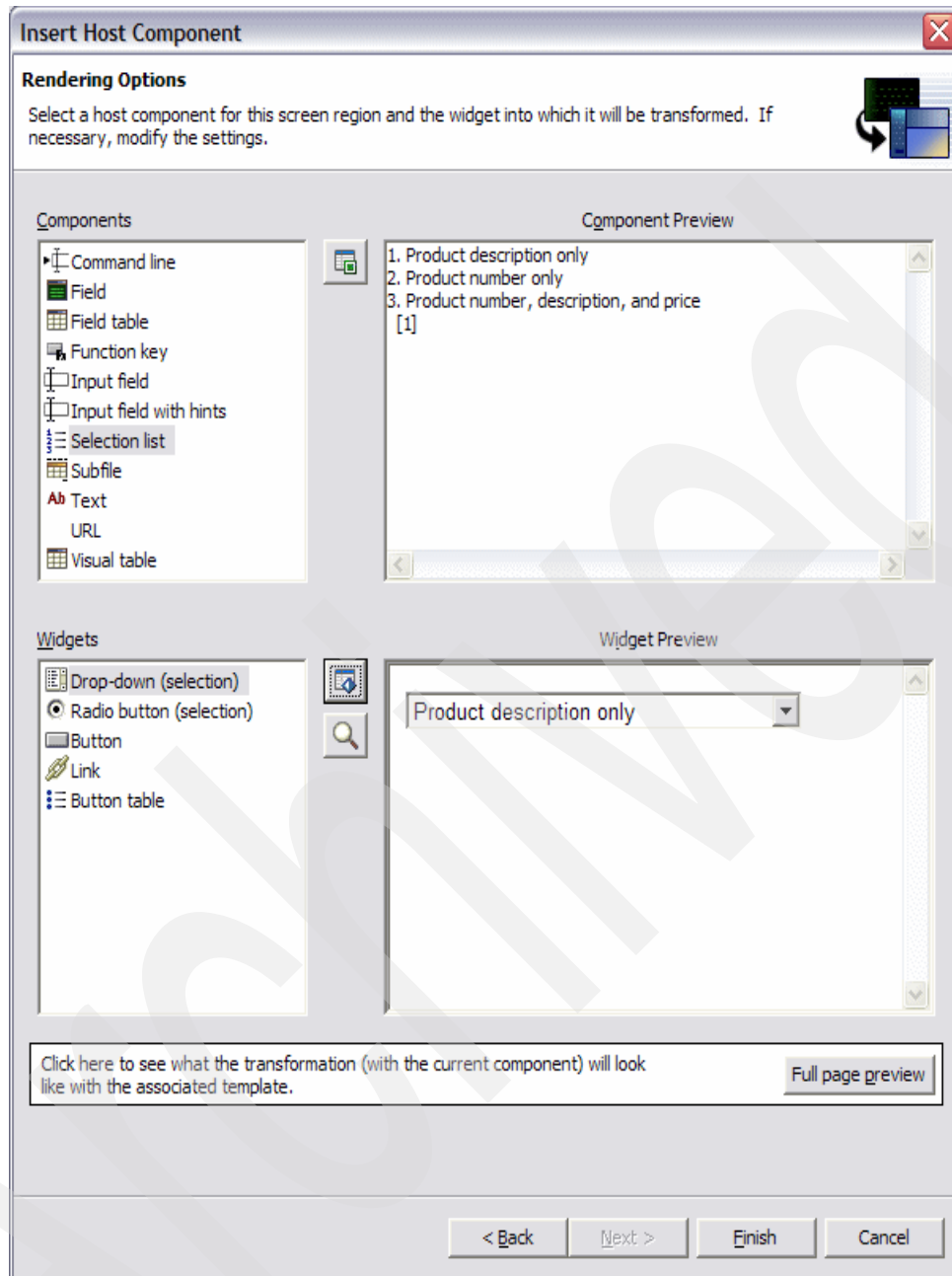


Figure 6-16 Insert Host Component - drop-down list

This process is repeated for the other search criteria. This can be done by selecting **HATS Tools -> Insert Host Component** function from the menu bar. Additional customizations can be made using the JSP editor in the resulting transformation.

The input field is added to the screen using the insert HATS component function as well. A table is added using the **Table -> Table** function from the WSAD tool bar, as shown in Figure 6-17 on page 55.

The short-cut
for inserting a
component:



Figure 6-17 Insert Table panel

The search entry field and drop-down list boxes can be placed in the table using drag-and-drop. Rows can be labeled simply by highlighting the cell and entering text. The row and column background colors can be changed using the attributes panel.

The lines can be removed from the input fields by adding the following line to the Project Settings source, as shown in Example 6-2.

Example 6-2 Removing input field lines

Add:

```
<setting name="stripUnderlinesOnInputs" value="true"/>
```

After the following line:

```
<class name="com.ibm.hats.widget.DefaultWidget">
```

The last element added to the screen is a cancel button. This button is inserted using the insert host key function and selecting the F12 key. The button is then labeled Cancel using the attributes panel.

6.8 Customizing the search results screens

The final screen to be customized in this application is the search result screen. This screen can contain pages of information. HATS provides wizards that facilitate combining multiple screens of data into one. In this example we combine the resulting search records into one table. This can be done by creating a macro that uses the looping feature of HATS. The macro will automatically traverse the records until the end of the list is reached.

The short-cut
for the widget
settings:



Macros can be created from the record macro feature in the Host Terminal window. Start creating the macro from the screen just before the first screen of data. At the data screen, select the loop icon from the tool bar. The macro is completed by following the wizard directions at the bottom of the screen.

The macro key is then added to the Parts Search screen transformation. This is done using the **HATS Tools -> Insert Host Component** function located on the menu bar. When this macro is run, the resultant table will be displayed using the default transformation.

This completes the customizations for the ZYX Parts Ordering System.

6.9 Importing and exporting a HATS project

WebSphere supports a number of team development tools including Clear Case and CVS. It is also possible to share and back up HATS projects using the import and export functionality of WebSphere Studio.

The first step in exporting a HATS project is selecting the project from the Navigator tab from the HATS perspective in WebSphere Studio. Then select **File -> Export** to open the export wizard, as shown in Figure 6-18.

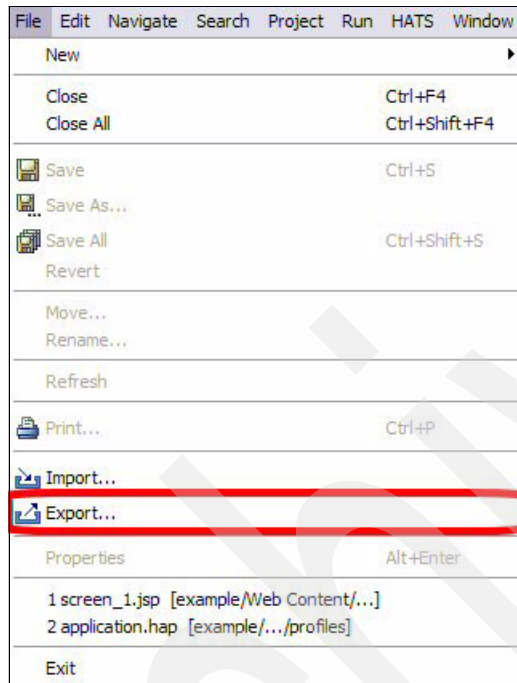


Figure 6-18 Export project

Select the Zip file option to save the file in the Zip file format and click **Next**. The subsequent panel is used to select the resources that will be exported, as shown in Figure 6-19 on page 57. Expand the project and check all of its subdirectories, as well as all the files in the project (in the right-hand pane). Note the following:

- ▶ Do *not* select the ear file and the ear project.
- ▶ Click **Create only selected directories**.
- ▶ Do *not* click Create directory structure for files.

In the Zip file field, type in the export destination of the file. (You can also select an export destination by clicking Browse.)

Select **Finish** to complete the export. It is a good idea to verify the success of the export by viewing the contents of the file by opening the file created and confirming that the transformations, screen captures, and other components are included.

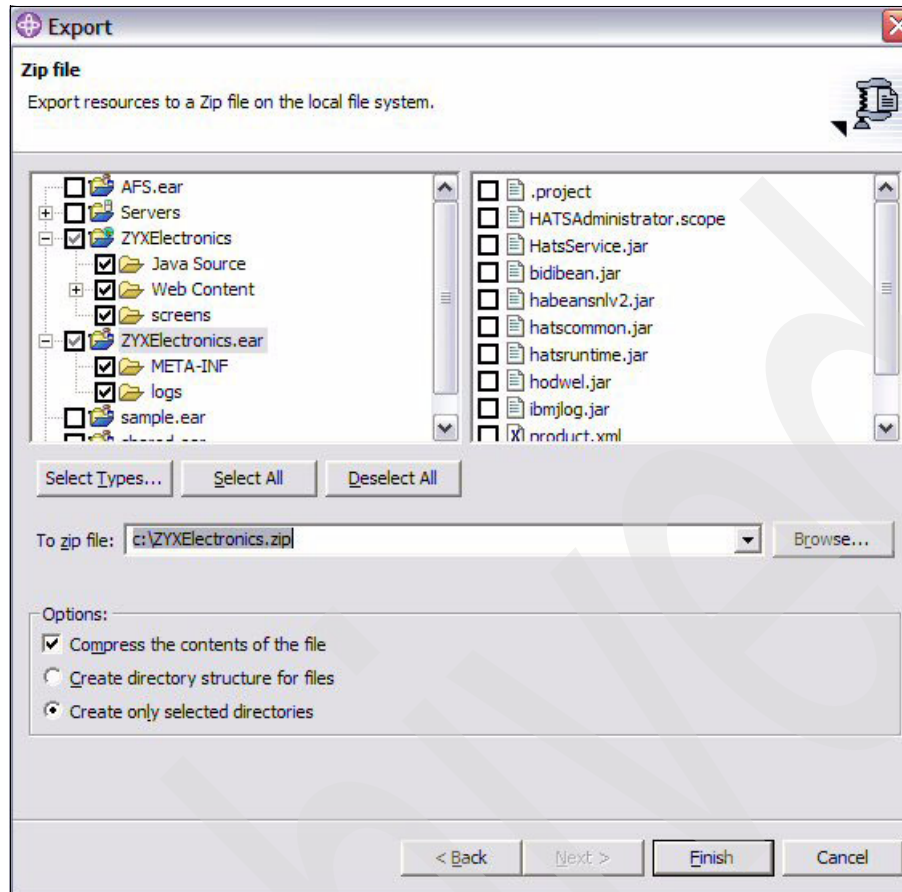


Figure 6-19 Export Zip file wizard

Importing a HATS project is just as simple. Before a project can be imported, a new HATS project must be created using the same name as the one that is being imported. If you use a different project name, you will get an error message that indicates that the context root does not match.

Selecting **File -> Import** will open the import wizard. Again, there are many options here. The Zip file option should be selected, followed by **Next**. This will bring up the Import Zip file panel, as shown in Figure 6-20 on page 58. In the Zip file field, type in the location of the Zip file. (You can also select the location by clicking Browse.) Enter the newly created HATS project name in the Folder field, or click Browse and select the name.

Select **Finish** to complete the import process. When the file is being imported, click **Yes to all** to overwrite existing files. The new project will now contain the original (imported) HATS project.

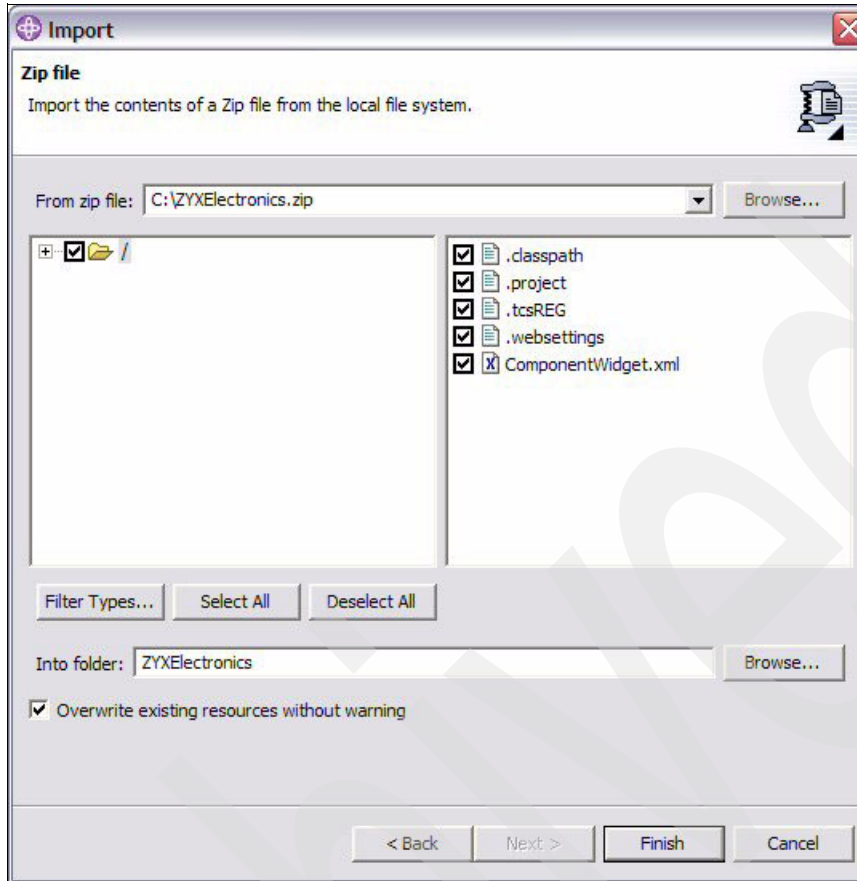


Figure 6-20 Import Zip file wizard

Customized ZYX Parts

In this chapter we will review the customized ZYX Parts Host Access Transformation Services (HATS) application created in Chapter 6, “Creating a customized HATS application” on page 39. This application was built in a matter of hours using the tools provided with HATS and WSAD. The original screen flow can be found in Chapter 3, “The ZYX 3270 Parts application” on page 13.

7.1 Customized ZYX Parts review

All screens in this example use the default template. The template provides the banner, corporate logo and URL link table. This establishes the backdrop for the host transformation. Each of the ZYX Parts screens has unique screen transformations, unlike the default transformations illustrated in Chapter 5, “Default HATS application” on page 31.

The logon screen was customized to present the userid and password fields in a table, creating a dramatically different look for the ZYX Parts ordering system. A logon button was added to provide the same functionality as the Enter key, as shown in Figure 7-1. This screen could be eliminated completely when integrated with the WebSphere Portal Server *credential vault*, which enable single sign-on. That is, once users log into the portal, they can automatically be logged into the ZYX Parts application.

ZYX Electronics							
The Leading Provider of Com							
<p>Information on ZYX Electronics:</p> <ul style="list-style-type: none">ZYX Electronics Home PageZYX Electronics MapZYX Electronics EmployeesZYX Electronics CompanyZYX Electronics ArticlesSupport	<p>Please log into the ZYX Parts Order System</p> <table border="1"><tr><td>USERID</td><td>====></td><td><input type="text"/></td></tr><tr><td>PASSWORD</td><td>====></td><td><input type="password"/></td></tr></table> <p>Logon</p>	USERID	====>	<input type="text"/>	PASSWORD	====>	<input type="password"/>
USERID	====>	<input type="text"/>					
PASSWORD	====>	<input type="password"/>					

Figure 7-1 New logon screen

The screen that would normally follow is a 3270 command screen. Users typically enter the application names on this screen. The customized ZYX Parts application now automatically navigates this screen using macros. This is the first step in reducing workflow and increasing ease of use.

The next screen a user would encounter is the ZYX Parts main menu. Using the HATS selection list component, we automatically converted the menu to buttons. It is important to note that the button descriptions and selection ordinals are dynamic, as shown in Figure 7-2. When a button on this screen is pressed, the corresponding option number and the Enter key are sent to the host screen. In our example, we click the Shop button.

ZYX Electronics	
The Leading Provider of Com	
<p>Information on ZYX Electronics:</p> <ul style="list-style-type: none">ZYX Electronics Home PageZYX Electronics MapZYX Electronics EmployeesZYX Electronics CompanyZYX Electronics ArticlesSupport	<p>ZYX Parts Main Menu</p> <ul style="list-style-type: none">ShopInquireNominateFeedbackInformationStartLogoff

Figure 7-2 New main menu

The screen following the main menu in the ZYX Parts application is the order form screen. All information required for this scenario can be obtained using the search function. A screen customization created with HATS simulates pressing F9. This prohibits the user from accessing other functions of the ZYX Parts application and reduces the workflow. As a result, the application is more secure and easier to use.

The user is then presented with the parts search screen, as shown in Figure 7-3. The inputs for this screen have been presented in a customizable table. The search field is converted to an entry box and the two criteria selection lists have been converted to pull-down lists. A pull-down list has the option of passing the Enter key after a selection is made. In this example, a search macro button has been added instead.

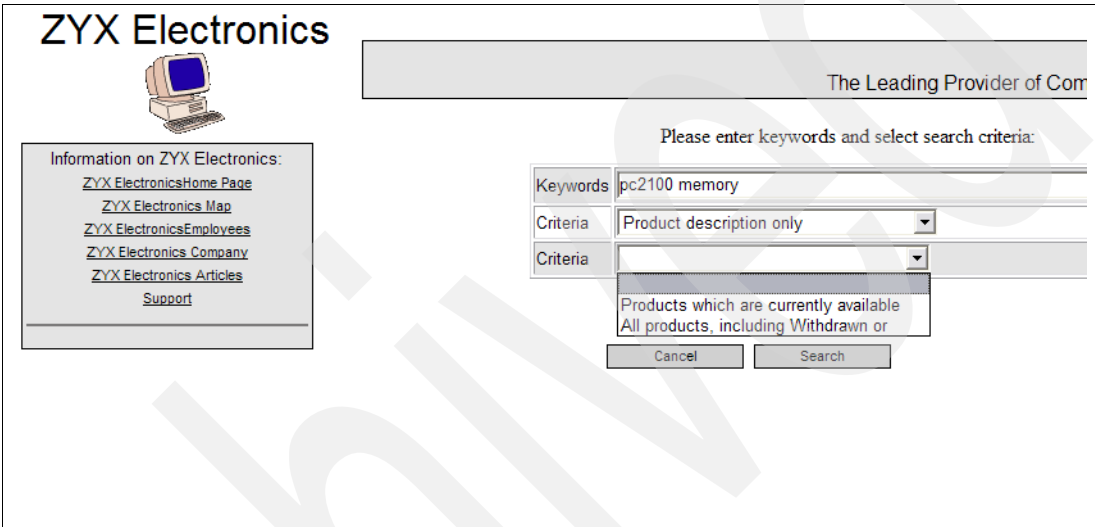


Figure 7-3 New product search page

The macro initiates the search and automatically iterates through the resultant data screens. A table is built containing all of the records returned and displayed on one page, as shown in Figure 7-4 on page 62. The table, by default, highlights every other line, thus improving readability. The user can now see more than twelve records on the screen simultaneously, a limitation of the legacy application. This is an example of leveraging the increased real estate provided by a GUI.

WSAD provides a number of screen elements for displaying data. HATS global variables can be used to populate a variety of other GUI screen objects.

Pressing the Continue button will return users to the search page.

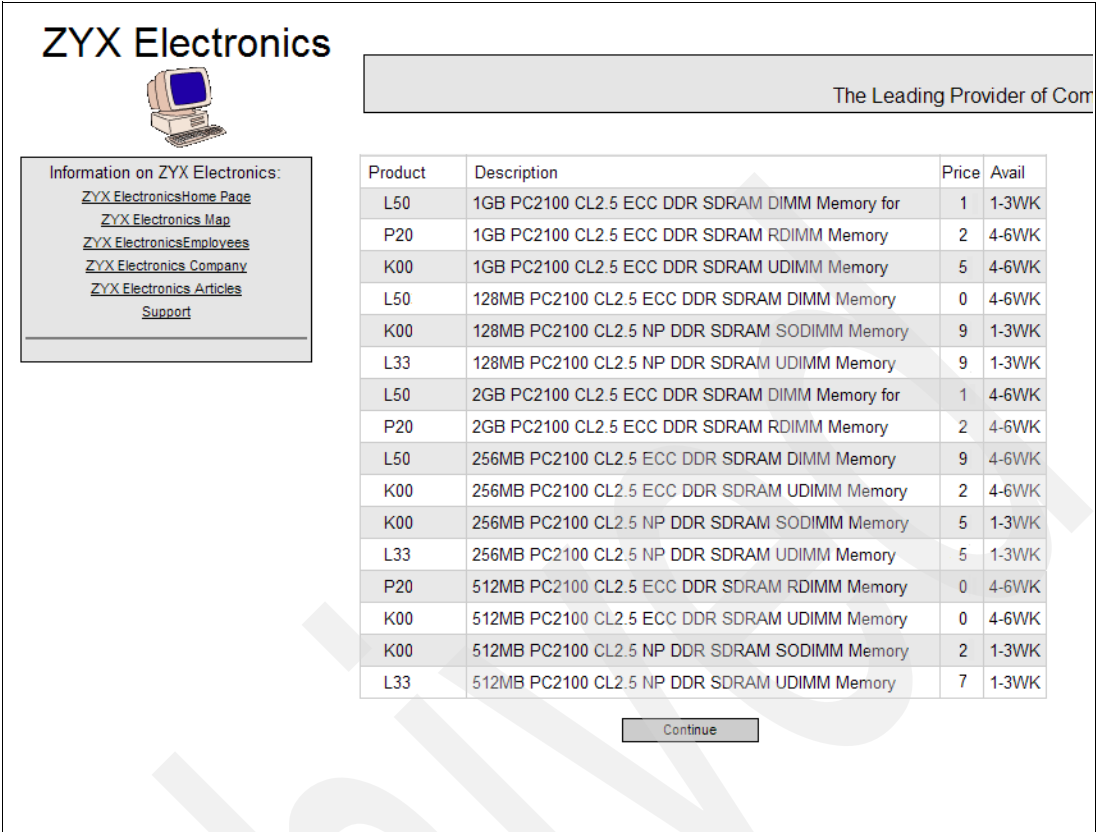


Figure 7-4 New search results page

HATS deployment

This part discusses the Host Access Transformation Services (HATS) and WebSphere on z/OS runtime environment. The zSeries platform provides the ultimate in high availability and scalability. We will discuss the anatomy, benefits and performance characteristics of this platform. In addition, we will illustrate the deployment of the application that was created in Part 3, “3270 applications using HATS” on page 23 to WebSphere on z/OS.

Archived

WebSphere runtime on z/OS

In this chapter we describe how the runtime components of WebSphere fit together and run together in a z/OS sysplex. WebSphere is an extensive collection of applications, so we confine ourselves to those components necessary to run (not develop) applications in production. Those same components are the ones of interest to us in monitoring performance.

We cover the following topics:

- ▶ Improvements in WAS V5 for z/OS
- ▶ zSeries hardware and z/OS
- ▶ The WebSphere programming model
- ▶ The application server model - how WebSphere application servers work under z/OS
- ▶ A typical customer installation
- ▶ Performance components

8.1 WAS V5 for z/OS improvements

The release of WebSphere Application Server (WAS) V5 for z/OS was highly anticipated, in large part due to its conformance to the WAS family programming model. This has removed many of the issues that led to challenges in moving applications from prototype to production environments. However, the convergence with the distributed WAS code base did not stop there.

The systems management model and tools in WAS V5 are also aligned with its distributed derivatives. The SMEUI and SMS are gone, replaced by a browser and JMX-based administration interface. As a result, there is no z/OS-specific learning curve for distributed WebSphere customers. In fact, WAS for z/OS V5 is now simpler to manage.

WAS V5 for z/OS is also simpler to install and requires a smaller installation image. The LPA footprint has been reduced from 160 MB to ~30 MB. The private area footprint in the Control Region has been reduced from 80 MB to less than 30 MB. The private area footprint in the Server Region has been reduced from 300 MB to 100 MB. The number of server processes has also been reduced.

It is easy to see that WAS for z/OS V5 has more modest hardware requirements than its predecessor. Many of the software prerequisites, including DB2® and LDAP, have also been relaxed. There are, of course, many other enhancements. For those considering z/OS as a runtime platform for Host Access Transformation Services (HATS) applications, these are the most germane. The improvements to WAS V5 for z/OS have paid off and its delivery has been very well received.

In the following section, we provide a brief introduction to zSeries hardware and to z/OS for the benefit of readers who are unfamiliar with the mainframe environment.

8.2 zSeries hardware and z/OS

WebSphere on z/OS is in some ways different from WebSphere on the distributed platforms, and much of that difference is due to the unique nature of the zSeries architecture.

The zSeries architecture is optimized for a mixed workload. Therefore, you will find the WebSphere servers sharing their mainframe with databases, transaction processing, development, batch jobs, and almost everything else. z/OS ensures that each piece of work is allocated the resources and the priority it needs to fulfill the installation's service objectives.

In performance terms, this means that poor response time in WebSphere could be due to excessive resource usage by another application. Furthermore, this could be “working as designed”—if the business has determined that Web applications should *not* have the highest priority, then at peak times WebSphere transactions could receive poor service. This is not a performance issue, it is a business issue.

8.2.1 Central processors and logical partitions

Each physical zSeries server comes with one or more processing units (PUs), some storage, and a Channel Subsystem that communicates with the outside world. Each PU can perform one of a number of functions, depending on the microcode loaded into it.

The PUs that run the operating system are called central processors (CPs). However, there is not a one-to-one mapping between instances of the operating system and the CPs. Rather, each copy of the operating system runs in its own *logical partition* (LPAR).

An LPAR has assigned to it a certain proportion of the total computing power, a certain amount of storage, and a certain number of channels. Moreover, these proportions can be dynamically adjusted as the workload changes.

When we talk about work running on a mainframe, we talk in terms of LPARs because that is how the mainframe is seen by an application: One LPAR = one system image = one interface between the application and the server. But keep in mind that one LPAR could be using as many as 16 CPs, or only half a CP.

Many operating systems, such as z/OS, z/VM® and Linux run in zSeries LPARs. In this redbook, we are only concerned with z/OS.

8.2.2 Parallel Sysplex®

To achieve high availability for a z/OS application, you need that application to run in multiple LPARs, ideally distributed between multiple physical servers. However, if you also want optimum performance, you need to ensure very close coordination between those LPARs. These two principles give rise to the concept of the sysplex. A *sysplex* comprises multiple z/OS LPARs (spread across one or more physical servers) that have three things in common:

- ▶ Shared disk space, containing (at the very least) the files (*data sets*, in z/OS parlance) that define the way the sysplex LPARs cooperate.
- ▶ A means of communication, called the *cross-system coupling facility* (XCF), that allows the sysplex LPARs to keep in touch and up-to-date with all interesting events.
- ▶ A Sysplex Timer® that keeps the physical servers' clocks in synchronization. The timer is not required if the whole sysplex is in the same physical server, as there is only one clock.

The Coupling Facility is a very high-speed shared storage area that can be used by z/OS instances for holding critical data. Using the Coupling Facility allows in-storage data to survive the failure of a z/OS instance, since any other instance can retrieve the data and continue to process it.

The Coupling Facility is, in fact, just another LPAR, running a special operating system optimized for just the one purpose. Coupling Facilities are connected to z/OS LPARs via high-speed connections. In a production environment there are usually at least two Coupling Facilities, for redundancy.

A typical customer environment might include several physical zSeries servers, containing a production sysplex, a development sysplex, and a test sysplex. The production sysplex might comprise the following:

- ▶ A z/OS LPAR using two CPs in server A
- ▶ A z/OS LPAR using three CPs in server B
- ▶ A Coupling Facility in server C
- ▶ A z/OS LPAR using one CP in server D
- ▶ A Coupling Facility in server D

The development and test sysplexes would probably have less computing power, and only one Coupling Facility each.

8.2.3 Address spaces and tasks

Turning to a single z/OS in a single LPAR, we now take a look at how work gets run within this environment.

When you run a job or start an application under z/OS (for example, a WebSphere application server), the operating system creates an *address space*. This is simply a piece of virtual storage that is assigned to the application for the duration of its existence, and the application runs within it. Up to 2 Gb of virtual storage is available for each address space, at least until 64-bit addressing is implemented. As well as user applications, many of z/OS's own system tasks run in their own address spaces.

The address space is the higher of the two layers of work management in z/OS. The lower unit is the *task*. The individual task is what gets dispatched by z/OS when it is ready to do work and has the highest priority of all the ready tasks. When an address space is started, the application that was invoked is assigned a task, and can work under the auspices of that task. If it needs to perform multiple units of work concurrently, it can request z/OS to create new tasks and assign them to the units of work. For example, if the WebSphere Application Server is handling several client requests at the same time, you would expect to see several tasks running in its address space.

8.2.4 z/OS components

Many of the major functions of z/OS run within address spaces rather than within the supervisory kernel. They have special authority and privileges. As far as WebSphere performance is concerned, two of the most important z/OS components are Workload Manager and UNIX® System Services.

Workload Manager

One of the most important performance-related components of any z/OS sysplex is Workload Manager (WLM). An instance of WLM runs on each z/OS LPAR, and together they build a picture of the workload currently running in the sysplex.

The installation defines to WLM the performance policies (goals) that apply to various items of work. WLM uses these goals, together with its knowledge of the running workload, to:

- ▶ Manage workload distribution and balancing, which includes scheduling new address spaces to handle increasing workload
- ▶ Distribute resources to competing workloads

UNIX System Services

To ease the portability of applications from other platforms to zSeries, z/OS includes a component called UNIX System Services (USS). It behaves as an operating system within an operating system, and provides:

- ▶ UNIX APIs
- ▶ A hierarchical file system (HFS) similar to that used on UNIX
- ▶ A UNIX-like command shell

Applications running under z/OS can make use of the traditional z/OS APIs, the UNIX APIs, or both.

In UNIX, applications run as *processes*; each process can comprise multiple *threads*. In z/OS, these are generally mapped to address spaces and tasks (TCBs), respectively.

WebSphere makes extensive use of the USS programming interfaces.

8.3 The WebSphere programming model

WebSphere Application Server Version 5 follows the specifications laid down in the Java 2 Platform Enterprise Edition (J2EE), Version 1.3. The J2EE V1.3 specification is itself based on the Java language programming platform, and it defines many aspects of enterprise Java applications including:

- ▶ The internal design of enterprise Java applications
- ▶ The mechanisms used by the application to communicate to other systems
- ▶ The process for deploying code into a server environment

8.3.1 Java overview

Java is an object-oriented programming language developed by Sun Microsystems Inc. It has the look and feel of C++, but is easier to use than C++. It includes a comprehensive set of APIs, ranging from desktop GUI to database access, to make a programmer's life easier. Since Java was introduced in 1995, it has been used extensively to build simple and complex applications.

Java programming model

In order to understand the J2EE programming model, it is important to understand the basics of the Java programming model. Java enforces object-oriented programming. Each type of object is represented by a construct called *class*, which has methods and variables. Classes can have subclasses, and together they build the class hierarchy. Another important piece is *interface*, a device that unrelated objects use to interact with each other. Interfaces only have *methods*, which can be implemented differently by different classes.

For code clarity and manageability reasons, Java has the concept of a package. A *package* groups logically related classes in the same way that a file system directory groups logically related files. There is no runtime performance impact with regard to packages.

In general, Java class files are packaged into one or more Java Archive (jar) files. This is a file format that is based on the popular ZIP file format for aggregating many files into one.

Java runtime execution

Java classes are not stored as object code in the sense that the hardware understands it. They are stored as bytecode to be interpreted by the *Java Virtual Machine* (JVM). Although this uses more runtime resources than traditional compilation, its great advantage is portability. Although the JVM is always platform-specific, the classes it runs are platform-independent. JVM also has the Just-In-Time (JIT) compile function, which dynamically compiles bytecode into execution code to improve performance.

But in this redbook we are not dealing simply with Java applications—we are talking specifically about z/OS WebSphere applications, which add another degree of sophistication. And WebSphere adheres to the J2EE standard for developing, deploying, and running enterprise applications. So what is J2EE?

8.4 Application server model

So now that we have some J2EE applications set up in the correct way, we need an environment within z/OS that will execute these applications.

8.4.1 Regions and instances

A J2EE application container, which includes both a Web container and EJB container as shown in Figure 8-1, runs *in a process*, in UNIX terminology. This translates to an address space in z/OS language, called the *server region*. Since we require some measure of availability and scalability, we create multiple server regions and manage the distribution of work to them via another address space called the *control region*. The combination of a control region and its managed server regions is called a *server instance*. Figure 8-1 illustrates this.

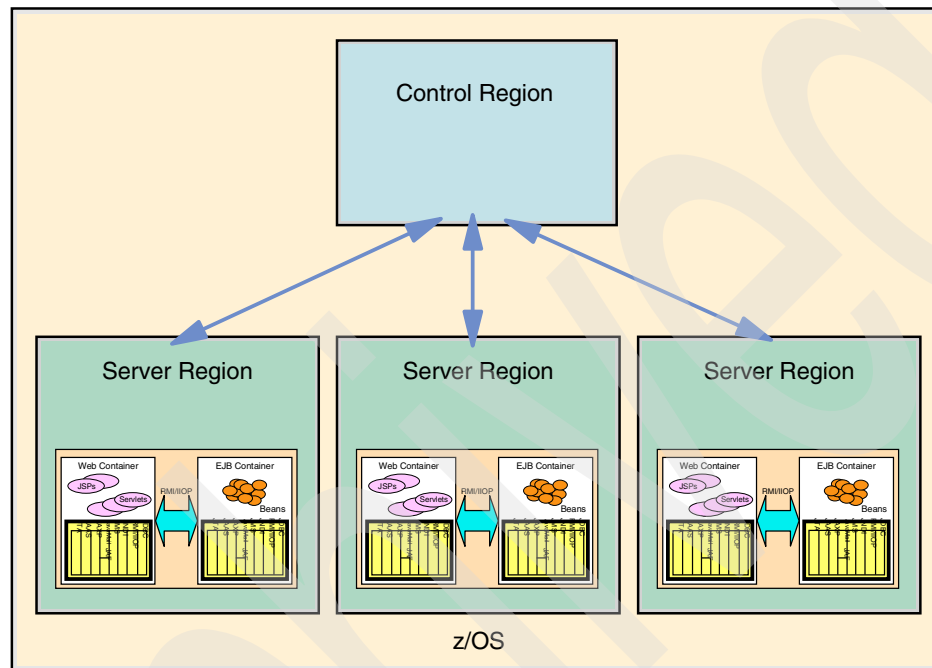


Figure 8-1 WebSphere Application Server instance

The control region is the end point of the communication (TCP connection) from the client. Its job is to distribute requests to the server regions. It places the requests on a WLM queue, from which WLM takes them and gives them to the server region that it deems best able to meet the performance goal. WLM is also able to start new server regions if the existing ones are near their capacity limit.

Within each server region (address space), many client requests may be handled concurrently. Each request runs as a *thread* in UNIX terminology (or a *subtask*, in z/OS terminology).

8.4.2 Servers and nodes

Now we have a control region plus a group of server regions. This still does not constitute high availability. The next stage is to clone the server instances, so that a number of control regions accept requests from the network, passing them on to the appropriate server regions within their own domains. The group of server instances is called a *server*. It is the entire server that presents itself to the clients, so that clients perceive a complete collection of control regions and server regions as a single entity with a single host name.

The server instances may coexist on the same LPAR, but it is common practice to have a server instance per LPAR, thus utilizing the high availability functions of the sysplex to provide the optimum service. Figure 8-2 on page 71 illustrates this concept.

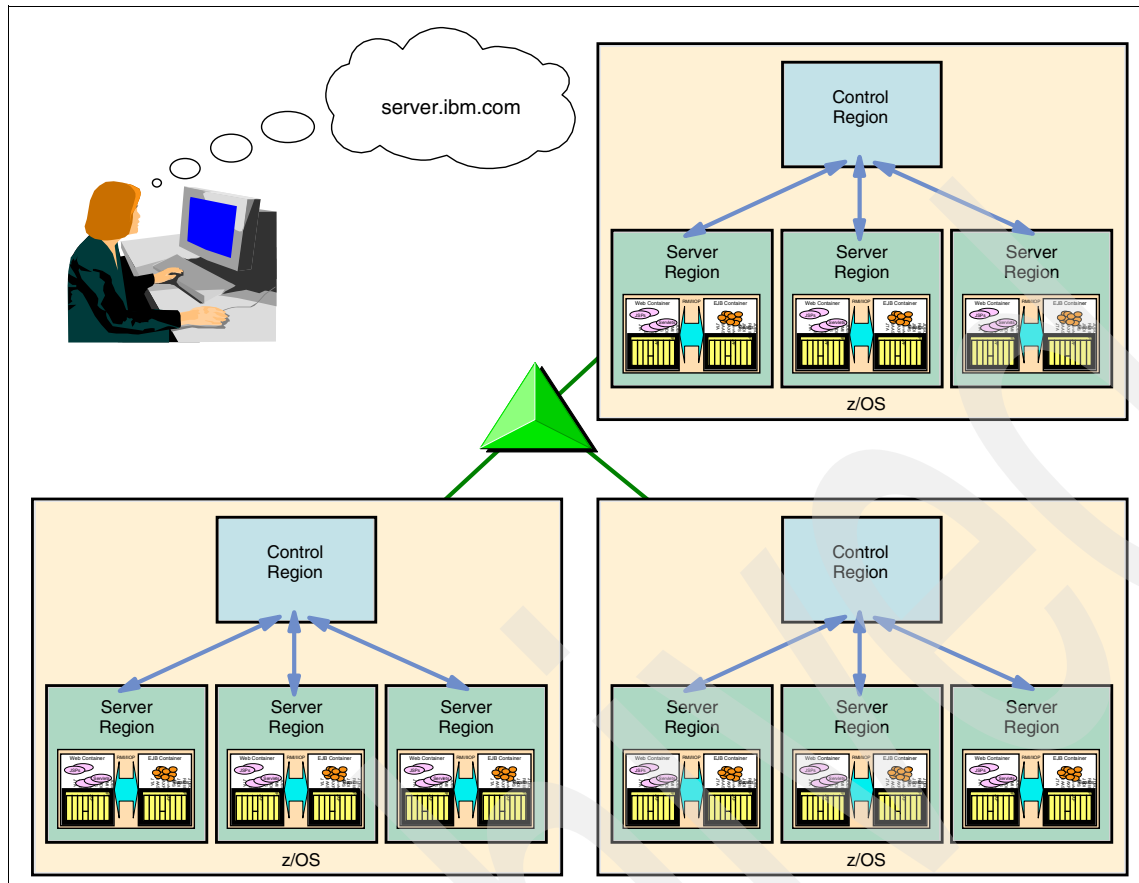


Figure 8-2 WebSphere server and server instances

A further refinement is to separate different e-business applications across different servers. Now there are multiple servers in the sysplex, comprised of multiple server instances running on each LPAR. The whole collection of servers is called a *node*; see Figure 8-3 on page 72.

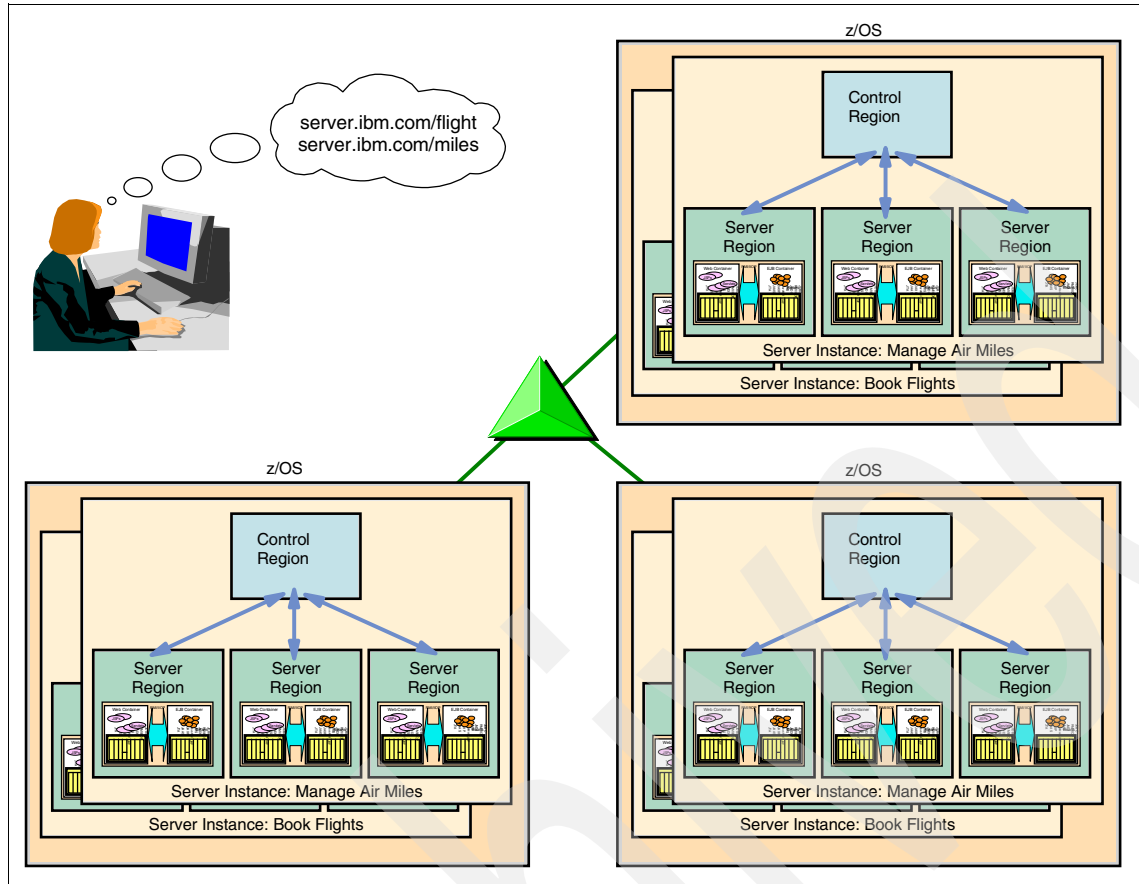


Figure 8-3 Multiple WebSphere servers in a sysplex node

The issues we now face are:

- ▶ How to present the same image (host name) to the clients from each server
- ▶ How to distinguish between different servers in the sysplex
- ▶ How to select the correct server instance if there is a session affinity between the client and the server—for example, if consecutive HTTP connections are part of the same transaction and must be handled by the same instance

Now we need a sophisticated connection distribution mechanism that can take account of these things, as well as input from WLM. The recommended method is described in 8.5, “Putting it together: a typical customer installation” on page 72.

8.5 Putting it together: a typical customer installation

A WebSphere installation on z/OS can be as clever and as sophisticated as you like, but it still needs clients, and client access, in order to fulfill its purpose. In other words, it needs a TCP/IP network and some form of intelligent load distribution mechanism. The diagram shown in Figure 8-4 on page 73 depicts a setup that combines high availability with WLM-assisted workload balancing.

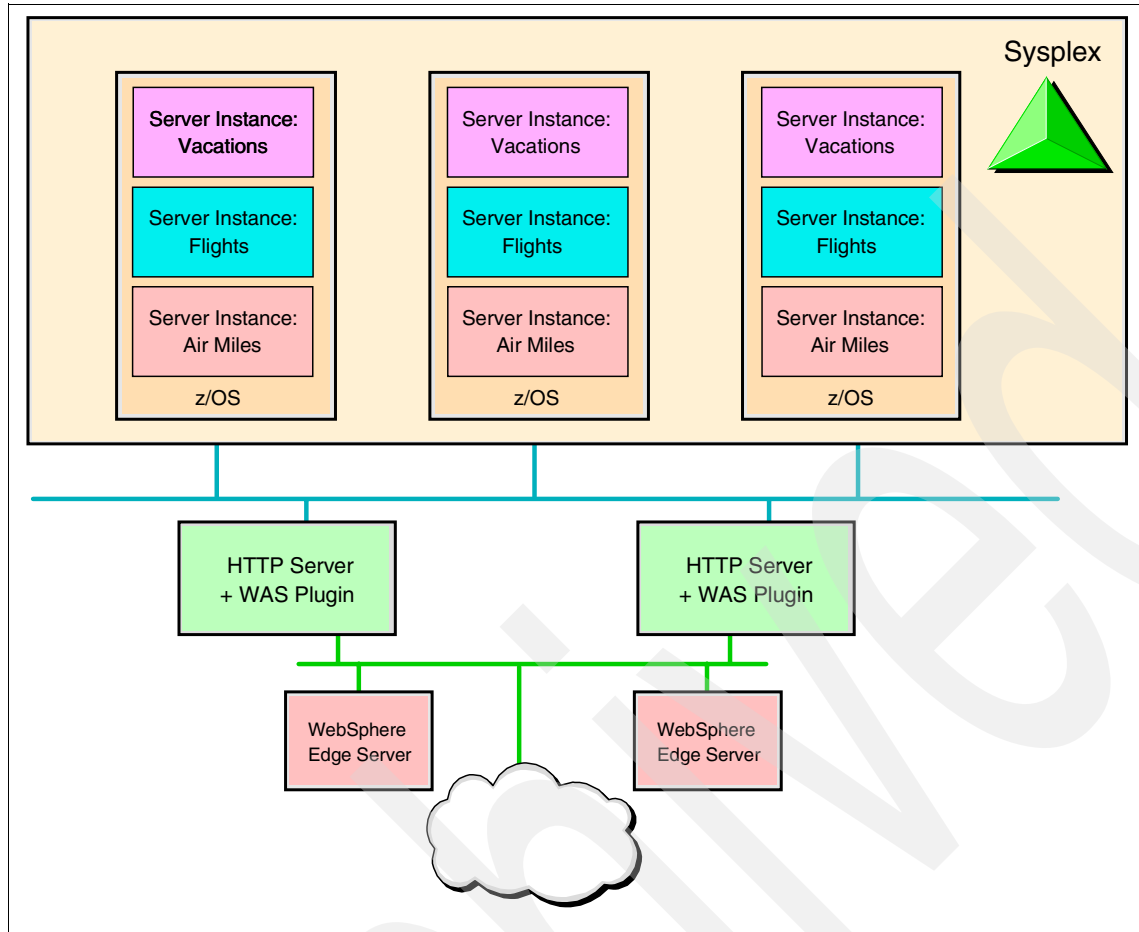


Figure 8-4 Typical WebSphere installation on z/OS

Here there are several distinct servers running on the z/OS sysplex, all listening for HTTP connections. The requirements are:

- ▶ To distribute incoming connections between the servers, based on WLM advice
- ▶ To check for session affinity, and to override WLM-based distribution if an affinity exists between a client and a server instance
- ▶ To distinguish between separate servers based on the client's input—usually a particular URI
- ▶ To ensure high availability

This is accomplished by a combination of the following:

- ▶ Sysplex Distributor on z/OS, which balances incoming connections among available server instances based on WLM input. However, it distinguishes between server instances by port number, not by URI. Also, it does not provide any affinity between client and server; it cannot recognize when a client needs to access the same server on consecutive HTTP connections.
- ▶ The IBM HTTP servers with the WebSphere plug-in on the outboard servers, which can check the incoming URI and translate it to a port number. They can also check cookies to determine if session affinity exists, and if so, bypass Sysplex Distributor, forwarding requests directly to specific server instances.

- ▶ The IBM WebSphere Edge Servers (one primary and one backup), which distribute incoming connections between the HTTP server instances. They can also act as caching proxies, relieving the application servers of the tedious task of serving static pages.

Thus, a client connecting to the IP address of the WebSphere Edge Servers is assured of service from the correct z/OS server, and of continuing service if any component on the path fails.

In an environment where Internet access is available to the WebSphere sysplex, there would also be firewalls in the picture; these have been omitted for simplicity.

Refer to the IBM Redbook *Enabling High Availability e-Business on zSeries*, SG24-6850, for an in-depth discussion of this configuration.

8.6 Performance components

From the preceding discussion, you will have concluded that WebSphere Application Server on z/OS is not the simplest environment in which to perform performance investigations. There are very many factors that could adversely affect response times. Some of them can be easily identified by the tools described in this book, and some of them cannot. In this section we inspect the route taken by a typical Web transaction to identify the potential performance bottlenecks.

Note: To determine the number of processors required to run the anticipated workload, have your IBM representative contact TechLine (or have your Business Partner contact PartnerLine) and request a sizing from SIZE390. You will be sent a questionnaire that you and your IBM representative can fill out to provide TechLine with the information they need about your WebSphere applications in order to do an accurate CPU sizing. We strongly recommend using SIZE390.

8.6.1 The TCP/IP network

The network is the first thing a user's request sees when it leaves the browser. It is also the least responsive to any tuning done by the installation, since much of it is outside your control. There are two things you can do to help:

- ▶ Provide a good design of the environment immediately outside the sysplex: fast routers, fast switches, fast adapters (OSA Express), and efficient routing.
- ▶ Optimize the z/OS TCP/IP stack; ensure that TCP buffer sizes are large enough, that MTU sizes are as large as possible, and that enough sockets are available for all the connections that need to be handled.

8.6.2 zSeries server

The obvious consideration is that WebSphere applications need hardware resources, memory, and CPU power. Less obvious factors are:

- ▶ Java applications use IEEE floating point instructions extensively. Prior to the 9672 G5 servers (predecessors to the zSeries), these instructions were emulated and performance could be adversely affected.
- ▶ If your Web site uses SSL encryption, this too is a heavy user of processing power. Hardware features available on the zSeries servers give them an advantage.

8.6.3 z/OS

As soon as the user's request hits the zSeries hardware, everything from that point onwards is under the control of z/OS until the response is sent back. Recommendations include:

- ▶ Turn off all tracing unless absolutely necessary.
- ▶ Turn off recording of systems management facility (SMF) records other than the ones you need. Some of the WebSphere performance tools make use of SMF, so a measure of SMF recording may be required.
- ▶ Put frequently used Language Environment® modules into the link pack area (LPA).

UNIX System Services

WebSphere is a UNIX System Services (USS) application, meaning that it runs in the UNIX environment under z/OS. In configuring USS, you tell it how many processes (address spaces), threads (tasks), sockets, and users it is expected to handle. WebSphere uses large numbers of these things.

USS also uses a hierarchical file system (HFS) similar to that implemented on UNIX and PC platforms. You should ensure that search paths for required files are optimized. Also, if you are sharing HFS files between LPARs, make as many files as possible read-only. Writing to shared files incurs a significant overhead. Also, make sure HFS files are mounted locally when possible.

Workload Manager

Workload Manager (WLM) is responsible for delivering the correct service level to each application and to each user, as determined by the installation. "Correct" is defined by the goals that you configure in WLM. Goals are generally of two kinds:

- ▶ Response time
- ▶ Velocity, meaning percentage of requested processor time allowed

The trick with WLM is to map a given piece of work to a defined goal. Recommendations include:

- ▶ Classify all regions except the WebSphere server regions as high velocity.
- ▶ The server regions should be given a reasonable velocity for starting up and other work, like garbage collection. The real application work is handled under the application environment. Classify this with a suitable response time with a percentile goal.
- ▶ In the WLM definition, do not limit the number of server address spaces that can be started for a subsystem instance in an LPAR. Specify `No Limit` in the definition for your application environment. You can place suitable limits on the number via WebSphere itself.

Security

Security in a WebSphere environment is administered by the Resource Access Control Facility (RACF) or one of its equivalent products. Security is costly in resources, so the principle to adopt is to define only what is necessary.

Enable only those classes (RACF authorization groupings) that you need. In particular, if you are not using EJB security roles (which define the users that can invoke individual methods), disable the appropriate facility class.

LDAP

The z/OS LDAP server is used by WebSphere, and runs multiple threads (subtasks) to service requests. It must be configured with enough threads to support the workload.

JVM

The Java Virtual Machine interprets (or, as recommended for best performance, compiles Just In Time) the Java classes. Java spreads its work around in its storage (the Java Heap) and periodically cleans it up (garbage collection). Too small a heap size will lead to frequent garbage collection and poor performance. Recommendations include:

- ▶ Monitor the garbage collection cycles and define a sufficient heap size.
- ▶ Run with the JIT compiler active.
- ▶ Set CLASSPATH (the Java equivalent of a list of concatenated libraries) to point to the most frequently used classes first, and to omit classes not used.
- ▶ Keep up to date with PTFs, since many of them have performance enhancements.

WebSphere

The structure of the WebSphere server environment itself has many options that allow you to optimize performance within the available hardware resources. For example:

- ▶ How do you split the applications between servers?
- ▶ How many LPARs/instances for each server?
- ▶ How many server regions allowed per instance?
- ▶ Do you let a server region run multiple threads, or process one transaction at a time within each server region?

Some general recommendations for WebSphere are:

- ▶ Put as much of the code as possible into the LPA, and the rest in the link list. This will eliminate unnecessary searching of libraries.
- ▶ Make sure that enough storage (both real and virtual) is available. WebSphere is a heavy user of storage.

Containers run and manage the EJBs, JSPs, and servlets. Many of the properties associated with the containers can be tuned to improve efficiency. In particular, the behavior of pools of resources can be adjusted in terms of when pool elements get reused.

HATS and WebSphere for z/OS

In this chapter we discuss running Host Access Transformation Services (HATS) applications on a z/Series server. It is expected that you also use *HATS User's and Administrator's Guide* and the *HATS Programmer's Guide* as sources of information on HATS, and that you review the HATS Readme.

This chapter includes the following information:

- ▶ Why to run HATS on z/OS
- ▶ Deploying HATS applications to WAS v5.1
- ▶ Overview of zAAP
- ▶ Recommended settings for performance
- ▶ Enabling graphics support for HATS on z/OS and OS/390®
- ▶ Configuring the display terminal

9.1 Why to run HATS on z/OS

WebSphere on z/OS merges the best of 30 years of mission-critical transaction monitors with the J2EE programming mode, thereby providing isolation, availability, consistency, resource management, and two phase commits (2PCs). It is the premier high availability platform in the industry today, providing near-zero downtime or 99.999% availability, with no more than 5 minutes per year of unplanned outages.

As a virtual server system, zSeries can run a diverse, changeable combination of workloads with a single set of shared system resources. More work is processed within a single server without over-configuring for complementary peaks. New operating system images can be started without affecting ongoing work. Consider the following points:

- ▶ *Availability* - The z/OS platform consistently delivers expected service regardless of unanticipated workload spikes or failures.
- ▶ *Selectivity* - WebSphere Application Server on z/OS provides the ability to guarantee service levels for specific types of customers and workloads as defined by business needs.
- ▶ *Integrated* - The composition and integration with multiple z/OS resource managers provides optimal performance, better availability and faster recovery.
- ▶ *Secure* - The industry's most stringent processes, tools and techniques for access control and asset protection can be found on the z/OS platform.
- ▶ *Efficient* - A lower total cost of ownership can be obtained through a reduction in trained system programmers and fuller utilization of existing capacity.

9.1.1 The z/OS difference: architecture matters

Consider the following differences:

- ▶ Each distributed server is dedicated to a particular workload.
 - Each server is sized for the maximum expected demand.
 - Additional dedicated servers are defined for failover, development, and test.
 - Peaks seldom coincide but resources cannot be shared, resulting in significant unused capacity.
 - Utilization is very low (5-10% average) over long periods.
- ▶ z/OS is based on a unique architecture that shares enterprise data and dynamically allocates resources across multiple servers and heterogeneous workloads.
 - It offers a more efficient architecture for a smaller number of high-volume applications.
 - WebSphere applications are transactionally co-located with EIS resources.
 - z/OS is legendary as a self-configuring, self-healing, self-optimizing and self-protecting platform.

9.1.2 Prioritization of work on z/OS using WLM and IRD

Workload Manager (WLM) differentiates z/OS in the marketplace with its ability to manage multiple, diverse enterprise workloads. This is done using policies based on Quality of Service (QoS) objectives. Resources are automatically allocated, adjusted, and reallocated to meet objectives. WLM will manage LPARs, CPUs, channels, I/O subsystems and DASD, TCP/IP connections and servers, thus ensuring 100% utilization of capacity.

Intelligent Resource Director (IRD) further differentiates z/OS, with its ability to manage resources across multiple partitions in a server. PR/SM™, IRD and WLM work together to ensure that the resources of the server are correctly balanced to enable work to complete within stated policy goals.

Processor resources, data bandwidth and I/O queueing decisions are perfectly balanced across the servers to manage diverse workloads within the parameters of the stated business goals.

Here are some example business goals that could be defined:

Transaction type

- ▶ Web “buy” versus “browse”
- ▶ B2B
- ▶ Batch payroll
- ▶ Test

User/user type

- ▶ Top 100 clients
- ▶ Typical clients
- ▶ Executive
- ▶ Design team
- ▶ Developers

Time periods

- ▶ 1AM - 4AM
- ▶ Mon. - Fri.
- ▶ Weekends
- ▶ End of quarter

9.1.3 WebSphere behavior in a sysplex

The sysplex is designed to run heterogeneous workloads. It can run WebSphere and traditional OLTP and DB applications simultaneously, at 100% utilization. Work is automatically balanced within a system to complete high priority work according to stated business goals.

New WebSphere servers are started automatically to accommodate spikes; they are automatically be quiesced when no longer needed. If a given system is overloaded, it will be temporarily bypassed in favor of less busy systems. If a system is unavailable, it will not receive new work. If a system fails, other systems will take over the work and the failed system will be recovered. When the sysplex is running at capacity, resources are adjusted to favor the more important workloads.

9.1.4 Anatomy of an enterprise cluster

WebSphere components deployed to a sysplex scale in a near-linear fashion and survive software outages without loss of application availability. Massive scalability and availability can be obtained using up to 32 zSeries servers, which can be coupled to create a single logical system. Servers can be located within a 40km campus and can contain up to 16 active and 4 spare processors. Clusters can contain up to 512 processors.

Data is shared across all z/OS images. The Coupling Facility (a special zSeries processor) maintains shared data integrity. Sysplex services include high speed messaging, caching and

lock management. Work is directed towards the sysplex and not an individual server; therefore it can run in any image, on any server.

9.1.5 z/OS benefits summary

- ▶ Unmatched scalability and availability using Parallel Sysplex advanced clustering
 - Ability to provide near-linear scale in a cluster
 - Ability to survive a software subsystem outage without loss of availability of the system
- ▶ State-of-the-art resource management
 - Ability to differentiate and prioritize work based on Service Level Agreements
 - Within a system (heterogeneous or homogeneous), and across systems
- ▶ Advanced resource recovery services
 - Optimized support for application models which require 2PC across IMS™, CICS®, MQ Series and DB2
- ▶ Non-disruptive change to software components (z/OS, WAS, resource managers)
- ▶ Recovery termination management that detects, isolates, corrects, and recovers from software errors
- ▶ Deep, end-to-end security integration
 - “Mainframe” security controls for user access to TCP/IP stack, ports, network
 - Integrated intrusion detection services (port scanning, stack attacks, flooding detection)

9.2 Deploying HATS applications to WAS v5.1 on z/OS

In this section we detail the procedure for deploying HATS applications to WebSphere v5 on z/OS. The Web-based administration console released with WebSphere v5 provides the WebSphere on z/OS administrators with the same look and feel as the distributed environment.

The first step in running a HATS application is assembling the project. This can be done by selecting the project followed by **menu -> Assemble HATS Project** as shown in Figure 9-1.

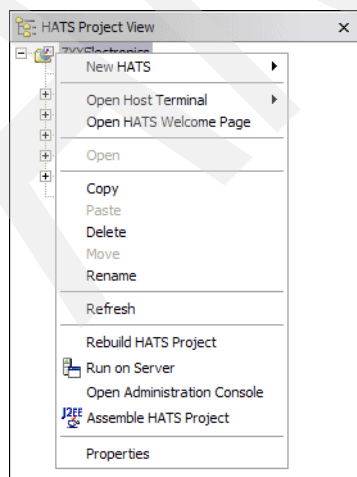


Figure 9-1 Assemble HATS project

The user must then select a file name for the assembled HATS project. The file must end with the EAR extension and be saved to the local file system. Enter a file name that is consistent with your project name and select **Finish** as shown in Figure 9-2.

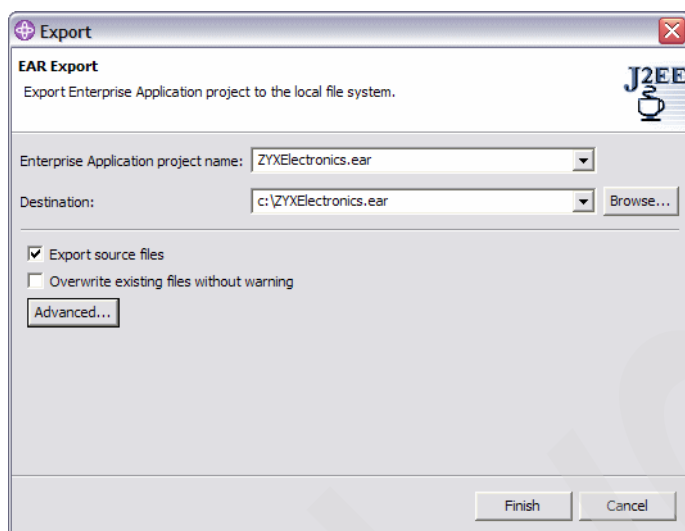


Figure 9-2 Export panel

The next step is logging into the WebSphere Application Server Administration Console as shown in Figure 9-3. If you do not have a userid and password, you will need to contact the system administrator for login credentials.

WAS Version v5.x provides four authorization types when global security is enabled:

- ▶ **Monitor** - Least privileged user that can view the WebSphere Application Server configuration and current state.
- ▶ **Configurator** - Monitor privilege plus the ability to change the WebSphere Application Server configuration.
- ▶ **Operator** - Monitor privilege plus the ability to change the run-time state, such as starting or stopping services.
- ▶ **Administrator** - Operator plus Configuration privilege and the permission required to access sensitive data including the server password, LTPA password, keys, and so on.

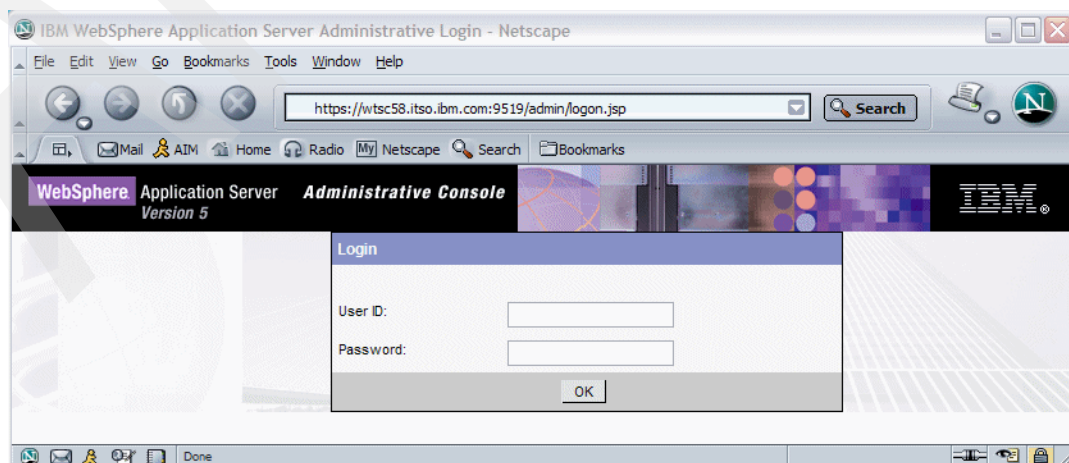


Figure 9-3 Admin console sign-on screen

After logging in, the admin console is displayed. The administration capabilities can be found by navigating through the explorer in the left window panel. The WebSphere Application Server version information can be found on the right side of the panel, as shown in Figure 9-4.

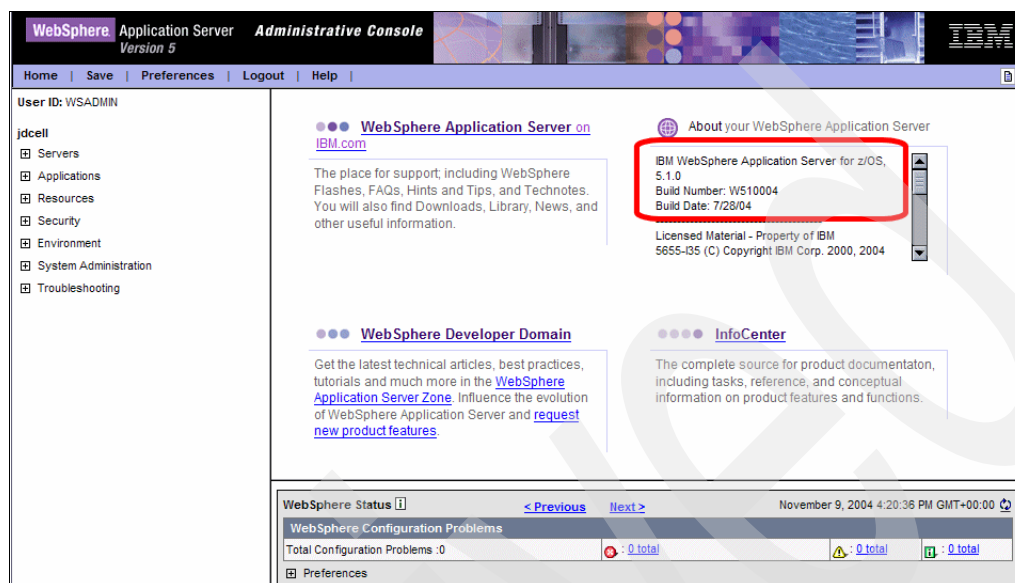


Figure 9-4 WAS version information

The HATS application can now be installed to the WebSphere Application Server by selecting **Applications -> Install New Application**. You can enter the fully qualified file name manually, or you can use the browse feature to graphically select the application, as shown in Figure 9-5 on page 83.

Note: The application can be installed from the server or from the local machine. When installing from local copy using a browser, a BB000271E HTTP REQUEST EXCEEDED error may be received. This can be the result of the default http buffer not being large enough for a HATS application. If this error condition is encountered, try increasing the protocol_http_large_dta_inbound_buffer variable and then restart the server.

The second field on this panel, Context Root, represents the portion of the URL immediately following the hostname and is used to determine which Web application will service an incoming request. The HATS studio has already stored this information in the EAR file and this field can be left blank. To continue, select **Next**.

Note: Each application running in a server instance requires a unique context root.

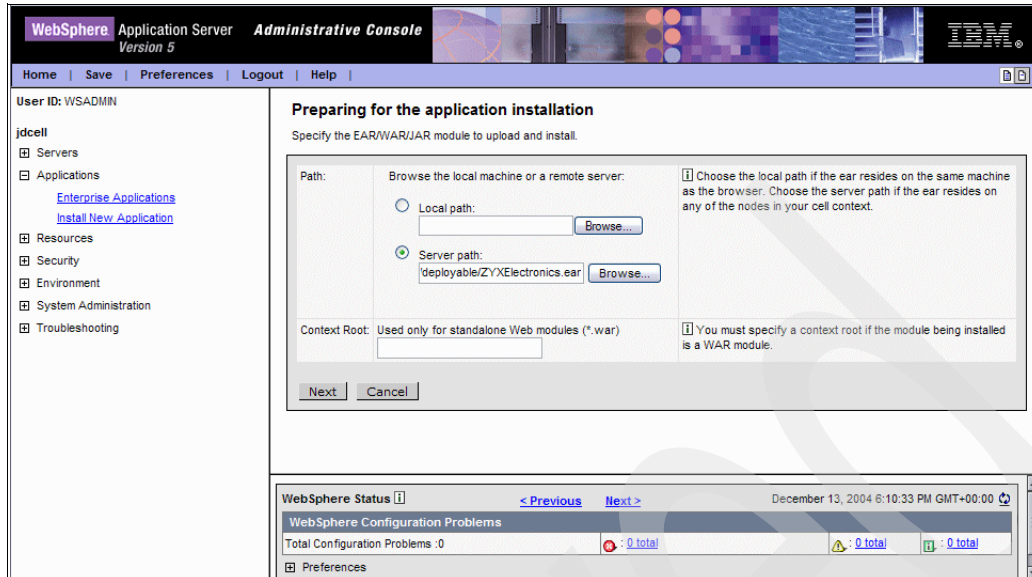


Figure 9-5 Select application to install

On the panel Preparing application for installation, the defaults can be taken as shown in Figure 9-6. Selecting **Next** continues the installation process.

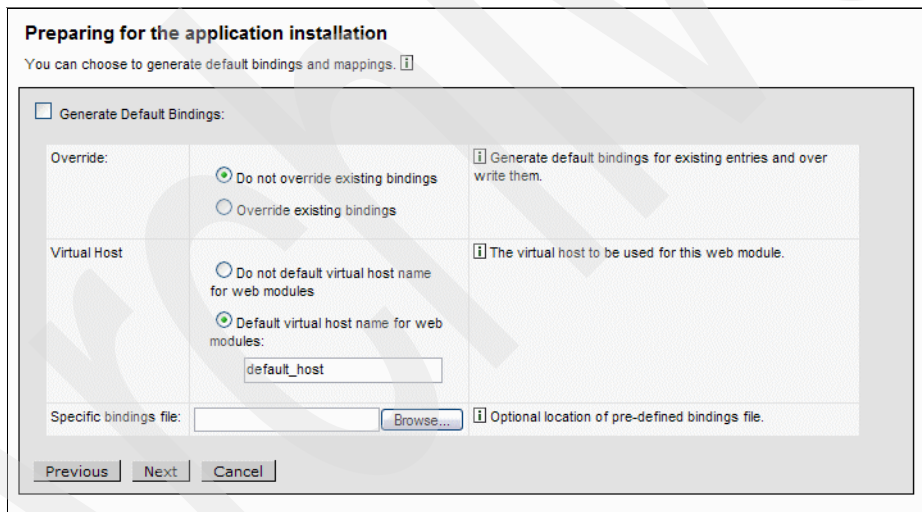


Figure 9-6 Preparing for installation panel

The balance of the installation is divided into five steps, each providing a configuration panel. The defaults can be taken, but it may be desirable to customize some of the installation options. Let's review the installation steps in more detail.

Step 1 provides customization options for application deployment. For example, the Pre-compile JSP feature will automatically compile the JSPs improving the initial application response time, as shown in Figure 9-7 on page 84. By default, this feature is not enabled.

The Application Name field should be populated with the same name that was provided during the HATS assembly process. Select **Next** to continue to Step 2.

Note: The user can jump directly to any of the installation configuration panels by selecting the *Step* link. In most cases users can jump directly to the summary panel (Step 5).

Install New Application
Allows installation of Enterprise Applications and Module

→ **Step 1: Provide options to perform the installation**

Specify the various options available to prepare and install your application.

AppDeployment Options	Enable
Pre-compile JSP	<input checked="" type="checkbox"/>
Directory to Install Application	<input type="text"/>
Distribute Application	<input checked="" type="checkbox"/>
Use Binary Configuration	<input type="checkbox"/>
Deploy EJBs	<input type="checkbox"/>
Application Name	ZYXElectronics.ear
Create MBeans for Resources	<input checked="" type="checkbox"/>
Enable class reloading	<input type="checkbox"/>
Reload Interval	<input type="text"/>

Next Cancel

[Step 2](#) Map virtual hosts for web modules
[Step 3](#) Map modules to application servers
[Step 4](#) Map security roles to users/groups
[Step 5](#) Summary

Figure 9-7 Application installation - options

The next installation panel allows for the specification of the virtual host where the application's Web modules will be installed as shown in Figure 9-8. Use the Virtual Hosts pull-down list to select an alternative destination. In this example, the default host is selected.

Install New Application
Allows installation of Enterprise Applications and Module

[Step 1](#) Provide options to perform the installation

→ **Step 2: Map virtual hosts for web modules**

Specify the virtual host where you want to install the Web modules contained in your application. Web modules can be installed on the same virtual host or dispersed among several hosts.

☒ Apply Multiple Mappings

Web Module	Virtual Host
<input type="checkbox"/> ZYXElectronics	default_host

Previous Next Cancel

[Step 3](#) Map modules to application servers
[Step 4](#) Map security roles to users/groups
[Step 5](#) Summary

Figure 9-8 Application installation - map virtual hosts

Step 3 allows for the mapping of application modules with application servers. WebSphere Version 5 introduced the concepts of *nodes* and *cells*. The HATS application can be selectively installed to the desired nodes using the apply button on this panel, as shown in Figure 9-9 on page 85.

Install New Application
Allows installation of Enterprise Applications and Module

[Step 1](#) Provide options to perform the installation
[Step 2](#) Map virtual hosts for web modules

→ **Step 3: Map modules to application servers**

Specify the application server where you want to install modules contained in your application. Modules can be installed on the same server or dispersed among several servers.

Clusters and Servers: WebSphere:cell=jdcell,node=jdnodea,server=jdsr01a
WebSphere:cell=jdcell,node=jdnodea,server=jdsr03a
WebSphere:cell=jdcell,node=jdnodea,server=jdsr04a

<input type="checkbox"/> Module	URI	Server
<input type="checkbox"/> ZYXElectronics	ZYXElectronics.war,WEB-INF/web.xml	WebSphere:cell=jdcell,node=jdnodea,server=jdsr01a

[Step 4](#) Map security roles to users/groups
[Step 5](#) Summary

Figure 9-9 Application installation - map modules

The installation process provides the administrator with the ability to manage administration security by assigning roles to users and groups. A *monitor* can view system state and configuration, but cannot make any changes. An *operator* has the capabilities of a monitor and can additionally make operational changes, such as starting and stopping a server. The *administrator* has the capabilities of the operator, but can also make configuration changes. Use the panel in Step 4 to customize security roles for the HATS application, as shown in Figure 9-10.

Note: For more information on enabling HATS application security, Web Express Login and other HATS security topics, refer to the HATS Info Center at:

<http://publib.boulder.ibm.com/infocenter/hatshelp/index.jsp>

Install New Application
Allows installation of Enterprise Applications and Module

[Step 1](#) Provide options to perform the installation
[Step 2](#) Map virtual hosts for web modules
[Step 3](#) Map modules to application servers

→ **Step 4: Map security roles to users/groups**

Each role defined in the application or module must be mapped to a user or group from the domain's user registry.

<input type="checkbox"/> Role	Everyone?	All Authenticated?	Mapped Users	Mapped Groups
<input type="checkbox"/> HATSAdministrator	<input type="checkbox"/>	<input type="checkbox"/>		
<input type="checkbox"/> HATSOperator	<input type="checkbox"/>	<input type="checkbox"/>		
<input type="checkbox"/> HATSMonitor	<input type="checkbox"/>	<input type="checkbox"/>		

[Step 5](#) Summary

Figure 9-10 Application installation - map security roles

Step 5 in the configuration sequence provides a summary of the installation options, as shown in Figure 9-11 on page 86. This allows one last review of the deployment parameters

before they are committed to the server. The Previous and Back buttons can be used to review the selections.

Step 3 Map modules to application servers
Step 4 Map security roles to users/groups

→ Step 5: Summary

Summary of Install Options

Options	Values
Distribute Application	Yes
Use Binary Configuration	No
Cell/Node/Server	Click here
Create MBeans for Resources	Yes
Enable class reloading	No
Deploy EJBs	No
was.policy.data	<pre>// // Extra permissions can be added if required by the enterprise application. // // NOTE: Syntax errors in the policy files will cause the enterprise application FAIL to start. // Extreme care should be taken when editing these policy files. It is advised to use // the policytool provided by the JDK for editing the policy files // (WAS_HOME/java/jre/bin/policytool). // grant codeBase "file:\${application}" { permission java.io.FilePermission "\${app.installed.path}\${/}*", "read, write"; }; grant codeBase "file:\${jars}" { }; grant codeBase "file:\${webComponent}" { };</pre>
Application Name:	ZYXElectronics.ear
Reload Interval	
Directory to Install Application	
Pre-compile JSP	Yes
Application Name	ZYXElectronics.ear

Previous Finish Cancel

Figure 9-11 Application installation - summary

When **Finish** is selected, the application is installed on the application server as shown in Figure 9-12. There are only a few steps remaining before the application can be used.

Installing..

If there are EJB's in the application, the EJB Deploy process may take several minutes. Please do not save the configuration until the process is complete.

Check the SystemOut.log on the Deployment Manager or Server where the application is deployed for specific information about the EJB Deploy process as it occurs.

ADMA5016: Installation of ZYXElectronics.ear started.

ADMA5009: Application archive extracted at /tmp/app_1001ff6c4b1/ear

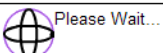
 Please Wait...

Figure 9-12 Installing Application screen

The changes must then be saved to the master configuration. Selecting the Save to Master Configuration option will present the save configuration panel, as shown in Figure 9-13 on page 87.

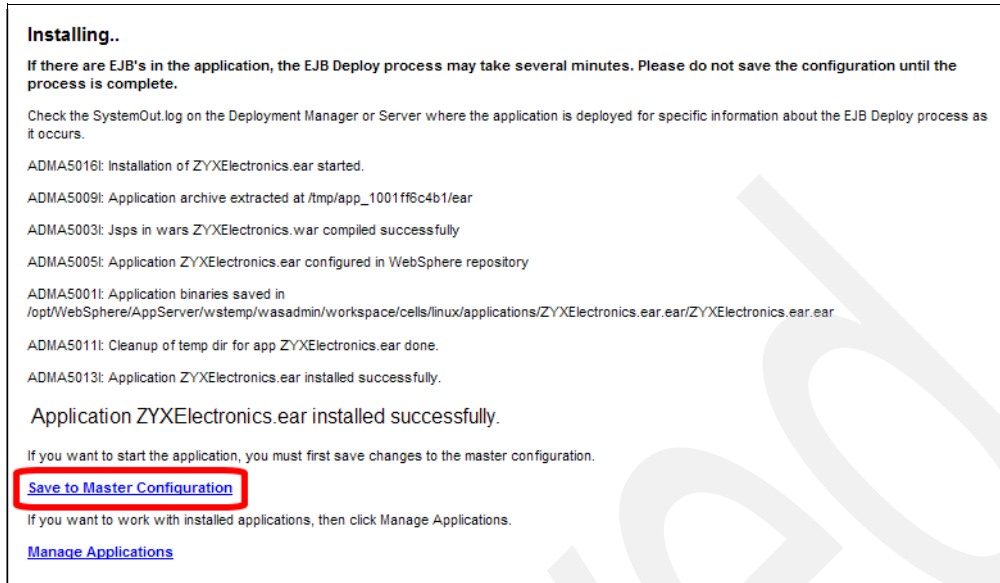


Figure 9-13 Save to master configuration

Confirm the save operation by clicking **Save** as shown in Figure 9-14.

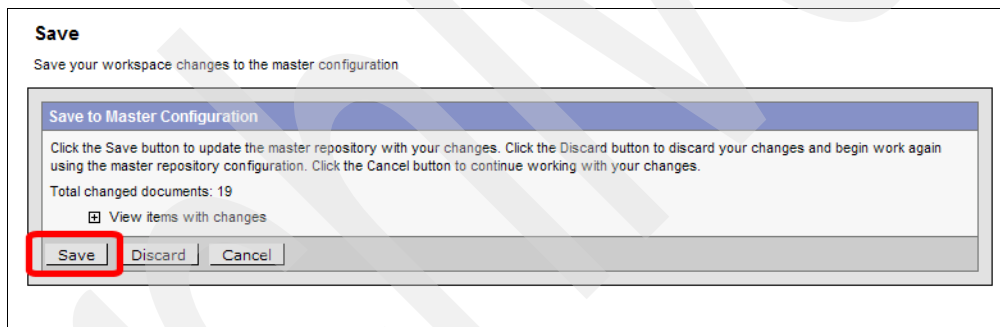


Figure 9-14 Save to master config confirmation

Now that the application is installed successfully, it can be run. **Select Applications** -> **Enterprise Applications** from the administration explorer to display the applications installed on the server. The applications are started by first selecting the corresponding check box and then clicking **Start**. The icons in the status column indicate the current state of the application as show in Figure 9-15 on page 88.

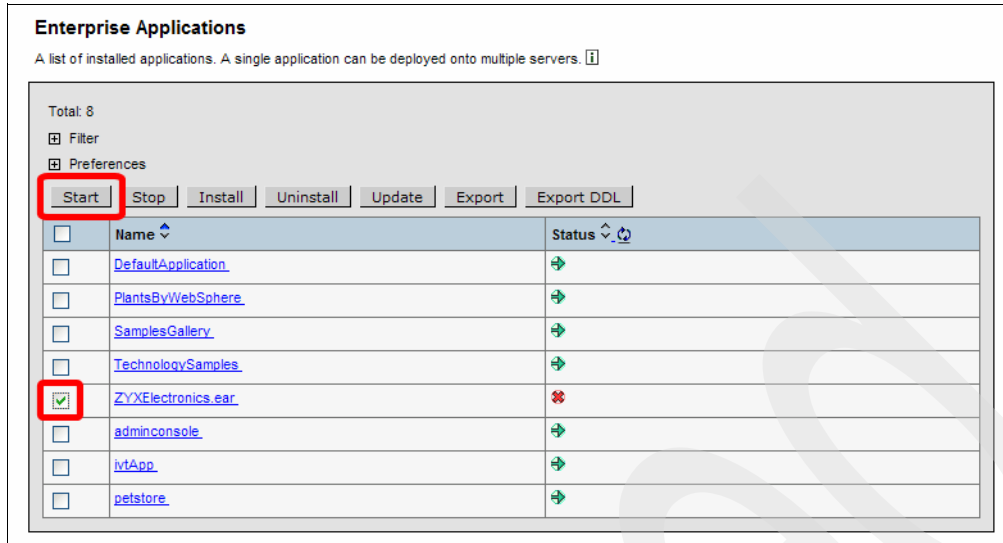


Figure 9-15 Select and start application

It is also necessary to update the Web server plug-in so that it can route incoming requests to the newly installed application. To update the plug-in select **Environment** -> **Update Web Server Plug-in**. Select **OK** to generate the plug-in as shown in Figure 9-16. If an external Web server is being used, the file must be copied to the appropriate servers.

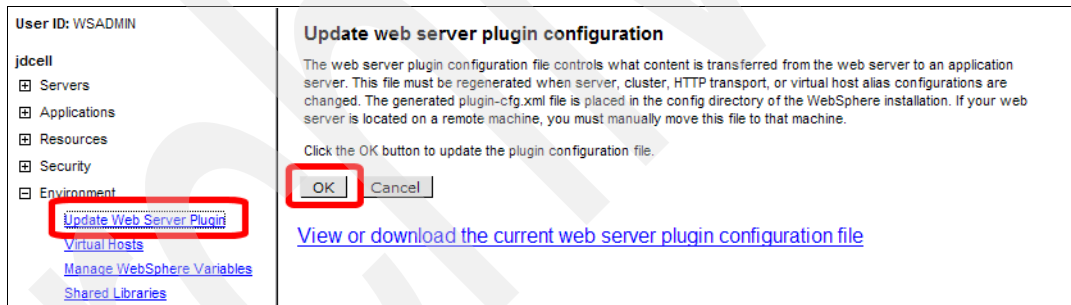


Figure 9-16 Update web server plug-in

The application can then be accessed from a Web browser. The URI format consists of the host name followed by the HATS application name. The following is an example:

http://zserver1/ZYXElectronics/

9.3 zSeries Application Assist Processor (zAAP)

Next we will cover the zAAP Application Assist Processor.

9.3.1 What is a zAAP

The IBM zAAP is an attractively priced specialized processing unit that provides a powerful z/OS Java execution environment for those who desire the integration advantages and quality of services benefits the zSeries platform provides. The IBM zAAP may provide the following benefits:

- ▶ Simplify and reduce server infrastructures by integrating Java applications and mission-critical data for higher performance, reliability, availability and security.
- ▶ Maximize the value of zSeries investment through increased system productivity by reducing the demands and capacity requirements on general purpose processors.
- ▶ Lower the overall cost of computing for WebSphere Application Server and other Java technology-based applications.

The IBM zAAP is very similar to a System Assist Processor (SAP). It cannot execute an initial program load (IPL), and can only assist the general purpose central processors (CPs) with the execution of Java programming under control of the JVM. However, unlike standard CPs, ICFs and IFLs, zAAPs can do nothing on their own. For this reason, IBM does not impose software charges on zAAP capacity.

Note: The zAAP can only help execute Java applications and application servers that use the IBM JVM. Execution of the JVM processing cycles on a zAAP is a function of the IBM SDK for z/OS.

9.3.2 Why use a zAAP

The IBM zAAP enables customers to strategically integrate their Java-based Web applications with their core business database environment by providing a more cost-effective, specialized z/OS Java execution environment.

zAAPs can enable you to run e-business Java Web applications next to mission-critical data for tightly integrated, highly secure and extremely efficient application and database serving. Execution of these applications within the same z/OS LPAR as the associated database subsystems can help simplify server infrastructures and improve operational efficiencies by reducing the number of TCP/IP programming stacks, firewalls, physical interconnections and their associated processing latencies.

In addition, because zAAPs are significantly less expensive than general CPs, customers can lower the cost of computing for z/OS Java-based application servers and all z/OS Java-based applications. By executing the Java cycles on a zAAP, you can reduce the demands and capacity requirements on general purpose CPs, which may then be available for reallocation to other zSeries workloads.

Furthermore, zAAPs allow customers to purchase additional processing power exclusively for Java workload execution without affecting the total MSU rating or machine model designation, as zAAPs do not carry a rated capacity. Consequently, IBM does not impose software charges on zAAP capacity. Additional IBM software charges will apply only when additional general purpose CP capacity is used.

Moreover, zAAPs may have the effect of reducing charges for subcapacity-eligible IBM software products by lowering the rolling 4-hour average MSUs for LPARs with assigned zAAPs.

Best of all, the IBM JVM processing cycles can be executed on the configured zAAPs with no anticipated modifications to the Java applications.

9.3.3 How does a zAAP work

When configured with general purpose processors (or CPs) within logical partitions running z/OS, zAAPs are designed to operate asynchronously with the general processors to execute Java programming under control of the IBM Java Virtual Machine (JVM). This can help reduce the demands and capacity requirements on the general purpose CPs, which may then be available for reallocation to other zSeries workloads.

The IBM JVM processing cycles can be executed on the configured zAAPs with no anticipated modifications to the Java application(s). Execution of the JVM processing cycles on a zAAP is a function of the IBM Software Developer's Kit (SDK) 1.4 product, z/OS 1.6, and the Processor Resource/Systems Manager™ (PR/SM™).

9.3.4 Prerequisites for the zAAP

zSeries Application Assist Processors (zAAPs) may be purchased and installed on z990 and z890 servers and future follow-on models only. In order to exploit a zAAP, the operating system must be migrated to z/OS 1.6 (or z/OS.e 1.6), the IBM SDK for z/OS, Java 2 Technology Edition, V1.4 with PTF (or later) for APAR PQ86689 must be used, and for WebSphere-based Java workloads, WebSphere Version 5.1 or above is required.

For customers planning to run WebSphere Application Server V5 under z/OS on a z890 or z990 server, you may be eligible to take advantage of some of the benefits that the zAAP provides for a limited time on your z890 or z990 prior to migrating to z/OS 1.6. Contact your local IBM or authorized business partner hardware sales specialist for more information.

Note: Because WebSphere Application Server for z/OS and OS/390, V4.0.1 will not be supported on z/OS V1.6 and its service will be discontinued on April 30, 2005 (as announced in Software Announcement 904-021, dated February 3, 2004), customers currently running WebSphere V4.0.1 are strongly encouraged to migrate to WebSphere V5 now in order to avoid the need to simultaneously upgrade the WebSphere and z/OS levels when z/OS 1.6 becomes available.

9.4 Performance settings and values

To enhance performance of HATS applications running on a z/Series server, IBM recommends that you use the WebSphere Administrative Console to adjust certain settings and values. In the following list, server1 represents the name of your server.

- ▶ Set the **Servers -> Application Servers -> server1 -> ORB Service -> Advanced Settings -> Workload Profile** value to LONGWAIT.
- ▶ For **Servers -> Application Servers -> server1 -> Server Instance -> Multiple Instances Enabled**, check the box.
- ▶ Set the **Servers -> Application Servers -> server1 -> Server Instance -> Minimum Number of Instances** to a valid number.
- ▶ Set the **Servers -> Application Servers -> server1 -> Server Instance -> Maximum Number of Instances** to a valid number.
- ▶ Set the **Servers -> Application Servers -> server1 -> Process Definitions -> JVM -> Initial Heap Size** value to an appropriate value for your installation.
- ▶ Set the **Servers -> Application Servers -> server1 -> Process Definitions -> JVM -> Max Heap Size** value to an appropriate value for your installation.

In addition, IBM recommends that you add the following variables under Manage WebSphere Variables and set the values as listed:

1. protocol_http_tcp_no_delay=1
2. protocol_http_max_persist request=1
3. protocol_http_timeout_input=0
4. protocol_http_timeout_output=0
5. protocol_http_timeout_persistentSession=0

Set wlm_stateful_session_placement_on=1 to enable HTTP requests to be distributed across servant regions without servant region affinity. See “Configuring an Application Server to use the WLM even distribution of HTTP requests function” for more information.

Each server region has 40 threads to handle concurrent HATS requests. This figure should be taken into account when sizing the number of server regions.

9.5 Enabling graphics support for HATS on z/Series

HATS enables you to transform host data such as tables into graphs on a Web page. Java APAR PQ78790 is required to support the complete graphing capabilities of HATS applications running on zSeries. These HATS applications require the Java Abstract Windowing Toolkit (AWT) and therefore need access to an Xwindows server. In the WebSphere Administrative Console, under Manage WebSphere Variables, you need to add the DISPLAY environment variable, and set the value to x.xx.xx.xx:0.0, where x.xx.xx.xx:0.0 is the XWindows environment variable that defines the IP address of the XWindows terminal.

9.6 z/OS version updates

The following sections present information on z/OS version updates.

9.6.1 z/OS 1.5 update

z/OS 1.5, which became available on March 26, 2004, is the first and only IBM operating system to provide Multilevel Security. This technology can help improve the way government agencies and other organizations share critical classified information. Other enhancements in z/OS 1.5 include extended self-optimization of WebSphere applications, improved backup and recovery of DB2 data, improved performance for DFSORT®, and expanded scope for Intrusion Detection Services.

z/OS 1.5 can also help simplify the management of z/OS with a new front-end for managing print across the enterprise, improved navigation of the z/OS library, and easier setup of DB2 Universal Database™ for z/OS V8, RMF™ and FTP on z/OS.

9.6.2 z/OS 1.6 update

z/OS 1.6, which became available on September 24, 2004, offers enhancements that include enabling a larger single image with support for up to 24 engines in a single z/OS image, 64-bit application development support for C/C++, and improvements in the scale of WebSphere address spaces.

It also offers capabilities to improve availability of TCP/IP networks across a sysplex, and extended scope of DFSMSrmm™ to manage tapes across platforms. z/OS 1.6 is the first release of z/OS that requires a zSeries server (z800, z890, z900, z990).

9.6.3 z/OS.e on zSeries z800 server

z/OS.e is a specially priced offering of z/OS for IBM zSeries 800 customers. z/OS.e Version 1 Release 4 is the second release of z/OS.e.

z/OS.e Version 1 Release 4 and follow-on releases use the same code base (customized with new system parameters) as z/OS Version 1 Release 4 and its follow-on releases. It invokes an operating environment that is comparable with z/OS in qualities of service, management, and reporting. Also, z/OS.e invokes the same z800 hardware functionality that you would get with full-function z/OS.

For more information, visit:

http://www.ibm.com/servers/eserver/zseries/zose/hw_sw.html

9.7 Configuring the Display Terminal for zSeries® when running WebSphere Application Server v5.0.x

With Java 1.3.1 installed on your zSeries system, you can display and use the Display Terminal check box. However, some additional configuration is required on the zSeries WebSphere Application Server and on your choice of a secondary network system (such as an AIX® system). This configuration is necessary because the zSeries server does not support graphical display devices as direct connect devices.

On a zSeries system, you can export the Display Terminal and graphics to an Xserver running on the remote system. The Xserver handles the graphical display of the terminal screen exported from the primary environment. The graphical display is also needed to view components and widgets in HATS Studio, such as visual tables and graphs.

To export the graphics to an Xserver, open the WebSphere administration console and select Manage WebSphere Variables. Add a new variable DISPLAY and supply the address of the Xserver. On the Xserver, you must grant permission to the host to export the graphics (such as xhost +mvs03).

Archived

Appendixes

Archived

Archived

What's new in HATS v5.0.4

This Refresh Pack is cumulative and supersedes the following Refresh Packs: IC39284, IC40475, IC41824, IC42725, IC43271.

For the most current release information, visit:

<http://www.ibm.com/software/webservers/hats/support.html>

Support alternate rendering sets for default rendering

New functions in v5.0.4

In previous versions and releases of HATS, default rendering has been restricted to the "main" rendering set in the application.hap file. With this Refresh Pack, you have the option of creating additional rendering sets, and to select which rendering set to apply to a particular transformation.

To define additional rendering sets, edit the application.hap file and locate the <defaultRendering> tag. Following the <defaultRendering> tag is the <renderingSet> tag with the name attribute specifying "main". Following the <renderingSet> tag are all of the <renderingItem> tags that define how default rendering is applied for the rendering set:

```
<defaultRendering>
  <renderingSet name="main">
    .
    .
    .
    <renderingItem associatedScreen=""
      description="Transform visual tables" enabled="false"
      endCol="-1" endRow="-1" startCol="1" startRow="1"
      type="com.ibm.hats.transform.components.VisualTableComponent"
      widget="com.ibm.hats.transform.widgets.TableWidget">
      <componentSettings>
        <setting name="minRows" value="5"/>
        <setting name="minColumns" value="4"/>
        <setting name="includeEmptyRows" value="false"/>
        <setting name="columnDelimiter" value=" "/>
      </componentSettings>
    </renderingItem>
  </renderingSet>
</defaultRendering>
```

```

        <setting name="excludeRows" value=""/>
        <setting name="excludeCols" value=""/>
    </componentSettings>
    <widgetSettings/>
</renderingItem>
<renderingItem associatedScreen=""
    description="Transform URLs" enabled="false" endCol="-1"
    endRow="-1" startCol="1" startRow="1"
    type="com.ibm.hats.transform.components.URLComponent"
widget="com.ibm.hats.transform.widgets.LinkWidget">
    <componentSettings/>
    <widgetSettings>
        <setting name="style" value=""/>
        <setting name="layout" value="SEPARATED"/>
        <setting name="captionType" value="TOKEN"/>
        <setting name="target" value="_blank"/>
        <setting name="separator" value=" | "/>
        <setting name="columnsPerRow" value="1"/>
        <setting name="linkStyleClass" value="HATSLINK"/>
    </widgetSettings>
</renderingItem>
.
.
.
</renderingSet>
</defaultRendering>

```

To define additional rendering sets, copy all of the code from the `<renderingSet>` tag to its `</renderingSet>` tag. Paste the copied code into the application.hap file immediately before the `</defaultRendering>` tag.

To differentiate the two rendering sets, change the name on the copied `<renderingSet>` tag to the name you want to use for the new rendering set. For this example, the name "alternate" is used, but you can assign a name of your choosing to the name attribute. You can also add a description attribute to the `<renderingSet>` tag to further define the different rendering sets:

```

<renderingSet description="alternate rendering set" name="alternate">
.
.
.
</renderingSet>

```

Because there is now more than one rendering set, one of the rendering sets must be designated as the default rendering set to use for default rendering in the project. The default attribute is added to the `<defaultRendering>` tag:

```

<defaultRendering default="main">
.
.
.
</defaultRendering>

```

Note: If you want to define all new rendering items for your new rendering set using the Rendering tab of the Project Settings in the HATS Studio interface, you can add an empty rendering set to the `<defaultRendering>` tag, name the empty rendering set, and specify the empty rendering set in the default attribute of the `<defaultRendering>` tag. For example:

```

<defaultRendering default="alternate">
.
.
.

```



```

</defaultRendering>
.
.
.
<renderingSet name="alternate">
</renderingSet>

```

Save the application.hap file and click the **Rendering** tab.

Now all that remains to complete the alternate rendering set is to add, delete, or modify the `<renderingItem>` tags and define the attribute and setting values you want to use. All of the tags and attributes for default rendering are described in Appendix A, “HATS Studio files” of the *HATS Programmer's Guide*, except the following:

renderingSet

The `<renderingSet>` tag is the enclosing tag for rendering items defined in the rendering set.

The attributes of the `<renderingSet>` tag are:

name

The name specified for the rendering set when it is created.

description

The description specified for the rendering set when it is created.

renderingItem

The `<renderingItem>` tag is the enclosing tag for rendering items defined in the rendering set.

A new attribute of the `<renderingItem>` tag is:

associatedScreen

The name of the captured screen used when a rendering item is added to or edited in the project. This item is informational purposes only.

componentSettings

The `<componentSettings>` tag is the enclosing tag for any settings modified for the component for this rendering item. This tag has no attributes.

widgetSettings

The `<widgetSettings>` tag is the enclosing tag for any settings modified for the widget for this rendering item. This tag has no attributes.

setting

The `<setting>` tag is the enclosing tag for any settings modified for the component or widget for this rendering item.

The attributes of the `<setting>` tag are:

name

Specifies the name of a customizable setting for the component or widget. The available settings depend on the component or widget.

Refer to Appendix A, “Component and widget descriptions and settings” of the *HATS User's and Administrator's Guide* for descriptions of component and widget settings.

value

Specifies the value of a customizable setting for the component or widget. The default values vary depending on the setting.

Notes:

- ▶ When you have finished making all of the changes for default rendering in the application.hap file, you must save, close, and reopen the file with the HATS Studio editor. When the application.hap file is reopened, the com.ibm.hats.transform.DefaultRendering class is added to the file, which contains `<setting name="applicationDefaultRenderingSetName" value="alternate"/>`. The value ("alternate") defines the rendering set that you added to the default attribute of the `<defaultRendering>` tag.

If you subsequently change the default attribute value of the `<defaultRendering>` tag, you must also change the value attribute of the `applicationDefaultRenderingSetName` to match. These two attributes must always be synchronized.

- ▶ If you want alternate rendering sets to be available to any new projects you create, go to the Navigator view of the HATS Studio and add the changes you made to the application.hap file located in the `WSxD_INSTALL_DIR\wstools\eclipse\plugins\com.ibm.hats\predefined\projects\new\Web Content\WEB-INF\profiles` directory, where `WSxD_INSTALL_DIR` is the directory where you installed WebSphere Studio.

Now that you have defined alternate rendering sets, you must specify their usage in the transformation.jsp files. Edit a transformation.jsp file for which you want to use an alternate rendering set and locate the `<HATS:DefaultRendering>` tag.

Until applying this Refresh Pack, the `<HATS:DefaultRendering>` tag had no attributes, nor were any additional tags associated with it. With this Refresh Pack applied, the `<HATS:DefaultRendering>` tag can be defined as follows:

```
<HATS:DefaultRendering row="1" col="1" erow="18" ecol="49"
    renderingSet="alternate" applyGlobalRules="true"
    applyTextReplacement="true" />
</HATS:DefaultRendering>
```

row

This defines the starting row of the text on the host screen for which the default rendering set is applied. The default value is 1.

col

This defines the starting column of the text on the host screen for which the default rendering set is applied. The default value is 1.

erow

This defines the ending row of the text on the host screen for which the default rendering set is applied. The default value is -1, which is a special designation for the host screen maximum number of rows.

ecol

This defines the ending column of the text on the host screen for which the default rendering set is applied. The default value is -1, which is a special designation for the host screen maximum number of columns.

renderingSet

This specifies the name of the rendering set to use for rendering this tag.

A rendering set with this name must be defined either in the application.hap file or within this <HATS:DefaultRendering> tag. If rendering sets with the same name are defined in both the application.hap file and the <HATS:DefaultRendering> tag, the rendering set on the <HATS:DefaultRendering> tag takes precedence.

The rendering set identified on the default attribute of the <defaultRendering> tag is used in the following cases:

- The renderingSet attribute is not specified on the <HATS:DefaultRendering> tag.
- There is no name defined for the renderingSet attribute of the <HATS:DefaultRendering> tag. For example:

```
<HATS:DefaultRendering renderingSet="" />
```

- The name defined on the renderingSet attribute does not match a renderingSet defined on the <HATS:DefaultRendering> tag or in the application.hap file.

applyGlobalRules

This specifies whether the default rendering should apply global rules to the data for this tag. The default is true.

applyTextReplacement

This specifies whether the default rendering should apply text replacements to the data for this tag. The default is true.

You can also define a rendering set directly on the <HATS:DefaultRendering> tag, specifying a name for the renderingSet attribute that only exists in a rendering set defined on the tag. For example:

```
<HATS:DefaultRendering row="1" col="1" erow="18" ecol="49"
    renderingSet="alternate" applyGlobalRules="true"
    applyTextReplacement="true" >
  <renderingSet name="alternate">
    <renderingItem associatedScreen=""
      description="Transform visual tables" enabled="false"
      endCol="-1" endRow="-1" startCol="1" startRow="1"
      type="com.ibm.hats.transform.components.VisualTableComponent"
      widget="com.ibm.hats.transform.widgets.TableWidget">
      <componentSettings>
        <setting name="minRows" value="5"/>
        <setting name="minColumns" value="4"/>
        <setting name="includeEmptyRows" value="false"/>
        <setting name="columnDelimiter" value=" " />
        <setting name="excludeRows" value="" />
        <setting name="excludeCols" value="" />
      </componentSettings>
    </renderingItem>
    .
    .
    .
  </renderingSet>
</HATS:DefaultRendering>
```

The descriptions for the tags and attributes for default rendering sets defined on the <HATS:DefaultRendering> tag are the same as the descriptions for the tags and attributes on the <defaultRendering> tag in the application.hap file.

New Fast5250 minimum delay setting in the HATS connection file for APAR IC43452

The Fast5250 screen-settling algorithm (default algorithm for 5250) had no minimum delay setting.

In some cases, the Fast5250 algorithm completed screen-settling before all of the related events are processed at the Telnet client. 5250 screens may settle too quickly, and text from the transient host screens might not be included in a transformation. This can happen especially when the screen is responding to a host application error with a locked-keyboard condition.

A new minimum delay setting, fast5250.minimumWait, can now be added to the HATS connection settings file (.hco). To enable this function, edit the source of the connection settings file in the HATS Studio. Locate the com.ibm.hats.common.NextScreenSettings class in the <classSettings> tag. Add the following setting:

```
<setting name="fast5250.minimumWait" value="0"/>
```

For the value attribute, specify the minimum time, in milliseconds, that HATS should wait during screen settling when using the Fast5250 algorithm. The initial default value is 0 milliseconds, which signifies that there is no minimum wait time. The Fast5250 algorithm is not affected. Increasing this value to a small minimum wait time enables more dependable screen settling.

New setting for the field widget for defect 30600

HATS Version 5.0.3 introduced a new setting, verticalSegmentAlignment, for the field widget to better handle complex host screens.

When you add the following as a setting for the field widget in your application.hap file, multiple fields are aligned word-by-word in the transformation:

```
<setting name="verticalSegmentAlignment" value="true"/>
```

A nowrap flag is automatically added to the HTML output when verticalSegmentAlignment is defined.

A new nowrap setting can now be added to the application.hap file for the field widget to turn off the automatic addition of the nowrap flag to the HTML output when verticalSegmentAlignment is defined. To enable this function, edit the source of the application.hap file in the HATS Studio. Locate the class name="com.ibm.hats.transform.widgets.FieldWidget" class in the <classSettings> tag. Add the following setting:

```
<setting name="nowrap" value="false"/>
```

This setting should only be added if the verticalSegmentAlignment setting is also specified for the field widget, and you continue to have alignment problems. The value of nowrap should always be "false". Setting the value to "true" could cause undesirable results.

New functions in v5.0.3

In the following sections, we describe new functions in v5.0.3.

Additional options for subfile recognition and rendering

In previous versions and releases of HATS, subfiles have been recognized using visual hints. With this Refresh Pack, you have the option of recognizing the table by visual hint (VisualTable) or by field information (FieldTable). In addition, you have the option to not render a subfile in a table at all, but to render the drop-down for detected actions and render each row line by line.

To use any of these options, you must edit the source of the project settings file (application.hap) or a transformation in the HATS Studio. Locate the `com.ibm.hats.transform.components.SubfileComponent` class in the file for transforming the subfile component, and specify one of the settings for the class, as follows:

Visual hint (the default setting):

```
<class name="com.ibm.hats.transform.components.SubfileComponent">
  <setting name="tableComponentClassName"
    value="com.ibm.hats.transform.components.VisualTableComponent"/>
</class>
```

Field information:

```
<class name="com.ibm.hats.transform.components.SubfileComponent">
  <setting name="tableComponentClassName"
    value="com.ibm.hats.transform.components.FieldTableComponent"/>
</class>
```

Row by row rendering:

```
<class name="com.ibm.hats.transform.components.SubfileComponent">
  <setting name="tableComponentClassName"
    value="none"/>
</class>
```

There are various ways in which you can set these settings; at the component level, the default rendering level, and at the transformation level. Following are examples of how to set the subfile to render each row line by line:

Component level in the project settings (application.hap)

```
<class name="com.ibm.hats.transform.components.SubfileComponent">
  <setting name="tableComponentClassName" value="none"/>
</class>
```

Default rendering level in the project settings

```
<renderingItem associatedScreen="WorkWithActiveJobs"
  description="Transform subfiles" enabled="true"
  endCol="-1" endRow="-1" startCol="1" startRow="1"
  type="com.ibm.hats.transform.components.SubfileComponent"
  widget="com.ibm.hats.transform.widgets.SubfileWidget">
  <componentSettings>
    <setting name="tableComponentClassName" value="none"/>
  </componentSettings>
  <widgetSettings/>
</renderingItem>
```

Transformation level

```
<HATS:Component type="com.ibm.hats.transform.components.SubfileComponent"
  widget="com.ibm.hats.transform.widgets.SubfileWidget"
  row="6" col="1" erow="19" ecol="80"

  componentSettings="tableComponentClassName:com.ibm.hats.transform.components.FieldTa
  bleComponent"
  widgetSettings="" textReplacement="" />
```

When row-by-row rendering is specified, you can specify that each subfile “row” should be visually separated with different colors on odd and even rows. To specify this, use the following settings in the `com.ibm.hats.transform.components.SubfileComponent` class in the source of the project settings file (`application.hap`) in the HATS Studio:

```
<class name="com.ibm.hats.transform.components.SubfileComponent">
  <setting name="subfileFixedNumberOfRows" value="true"/>
  <setting name="subfileNumberOfRowsPerRecord" value="1"/>
  <setting name="subfileDetectActionInputFieldPos" value="false"/>
</class>
```

Following are the descriptions of the settings:

subfileFixedNumberOfRows

This specifies whether there are a fixed number of rows for each subfile record. Valid values are true and false. The default is true.

subfileNumberOfRowsPerRecord

This specifies the number of rows in each subfile record. This is only valid when `subfileFixedNumberOfRows` is set to true. The default value is 1.

subfileDetectActionInputFieldPos

This specifies whether the number of rows per subfile record should be determined by detection of the action input field position of each record. Valid values are true and false. The default is false.

Only one of the `subfileFixedNumberOfRows` or `subfileDetectActionInputFieldPos` settings can be set to true at the same time. They are mutually exclusive.

In previous versions and releases of HATS, the input fields in subfile records are sometimes incorrectly recognized as action fields if the input field length is less than three, and rendered as drop-down lists. To render the text input fields correctly, you can now specify the action input field start column and action input field length.

Locate the `com.ibm.hats.transform.components.SubfileComponent` class for recognizing the subfile component, add the following settings, and modify their values:

```
<class name="com.ibm.hats.transform.components.SubfileComponent">
  <setting name="subfileOverrideActionFieldStartCol" value="false"/>
  <setting name="subfileActionFieldStartCol" value="1"/>
  <setting name="subfileOverrideActionFieldLength" value="false"/>
  <setting name="subfileActionFieldLength" value="1"/>
</class>
```

Following are the descriptions of the settings:

subfileOverrideActionFieldStartCol

This specifies whether to override the action input field start column. Valid values are true and false. The default is false.

subfileActionFieldStartCol

This specifies the start column of the action input field. This is only valid when subfileOverrideActionFieldStartCol is set to true. The default value is 1.

subfileOverrideActionFieldLength

This specifies whether to override the action input field length. Valid values are true and false. The default is false.

subfileActionFieldLength

This specifies the field length of the action input field. This is only valid when subfileOverrideActionFieldLength is set to true. The default value is 1.

The subfile widget now supports foreground colors and extended attributes detected in the subfile. Supporting foreground colors and extended attributes requires additional settings in the project settings file (application.hap) or a transformation in the HATS Studio. Locate the com.ibm.hats.transform.components.SubfileWidget class for transforming the subfile widget, add the following settings, and modify their values:

```
<class name="com.ibm.hats.transform.widgets.SubfileWidget">
  <setting name="preserveColors" value="true"/>
  <setting name="mapExtendedAttributes" value="true"/>
  <setting name="blinkStyle" value=" "/>
  <setting name="reverseVideoStyle" value=" "/>
  <setting name="underlineStyle" value=" "/>
  <setting name="columnSeparatorStyle" value=" "/>
</class>
```

For consistency, you might want to set these values to match the same settings for the field widget.

For the reverseVideoStyle setting, the default is blank, but you can create a cascading style sheet (CSS) class to control the appearance of the background colors. You can also create cascading style sheet classes to control the appearance of the extended attributes.

Style sheets control elements of output such as font color, size, and background color. Cascading stylesheet (CSS) is a simple style language that you can use to attach style to HTML elements. HATS provides stylesheets to modify color schemes and font size. The stylesheets that HATS provides are located in the Web Content/Common/Stylesheets node of the HATS Project View tab.

For more information on cascading style sheets, go to:

<http://www.w3.org/Style/CSS>

New selection list component setting (APAR IC42366)

One of the functions of the selection list component is to group related list items together when used in default rendering. In order for two list items to be grouped together, the target input field for the value of one of the list items is the same for both list items. The two list items must not have any other text between them.

HATS was not rendering some text following list items when default rendering was used with the selection list component. Text is lost when a long list item is grouped with a short list item that contains text or input fields immediately following them on the same line.

A new setting, `groupIndividually`, has been added to the selection list component to group list items individually, preventing text immediately following a list item from being lost.

Note: This setting will cause undesirable behavior if it is used with the drop-down (selection) or radio button (selection) widgets, because they require groups of one or more list items to render properly. When this setting is enabled, it is recommended that you use a widget that is not concerned with grouping. The button and link widgets are both examples of widgets not concerned with grouping.

To enable this function, edit the source of the project settings file (`application.hap`) in the HATS Studio. Locate either the `<renderingItem>` tag defining the default rendering of the selection list or the `<class name="com.ibm.hats.transform.components.SelectionListComponent">` tag for recognizing the selection list. Add the following setting:

```
<setting name="groupIndividually" value="true"/>
```

New setting for the field widget (APAR IC41065)

HATS was not rendering some complex host screens correctly when default rendering was used, resulting in the misalignment of input fields and text. A *complex screen* is one that contains multiple input fields at varying column positions across the screen, or contains fields of static text that are expected to align above or below a row of associated input fields.

Default rendering uses an HTML table to align a Web page. The number of columns in the table is equal to the number of columns on the host screen. Normally, each field on the host screen is rendered in its own table data (`<td>`) tag, ensuring that every field starts at the correct horizontal position. However, misalignment can occur on screens that contain input fields at multiple column positions, even at differing vertical row positions, because input fields take up more horizontal space than normal text.

A new setting, `verticalSegmentAlignment`, has been added to the field widget to better handle complex host screens. Instead of using a single `<td>` tag for a field of static text that is not in an input field, multiple `<td>` tags are used. Each word segment is now wrapped in its own `<td>` tag. A word segment is defined as a group of sequential letters followed by any trailing whitespace. Multiple `<td>` tags ensure that every word on the host screen rendered by the field widget begins at the correct horizontal position in the HTML table generated by default rendering.

To enable this function, edit the source of the project settings file (`application.hap`) in the HATS Studio, locate the `<class name="com.ibm.hats.transform.widgets.FieldWidget">` tag, and add the following setting:

```
<setting name="verticalSegmentAlignment" value="true"/>
```


HATS portlet support in a Web Service Remote Portlet (WSRP) configuration

This Refresh Pack provides support for HATS portlets running in a Web Service Remote Portlet (WSRP) configuration. However, there is a limitation when used with WebSphere Portal V5.0.2.1. The WebSphere Portal Credential Vault plug-in provided by Web Express Logon (WEL) does not function.

Archived

APARs fixed in HATS Service Pack 4

Problems fixed in Service Pack 4

Refer to the HATS support page for detailed descriptions of these APARs:

IC41857 - Logout from HATS Administrative Console reports FormLogout servlet not found.

IC42532 - Macro "Stop Recording" button greys out when using the Define the Workstation ID as "Use a specified value."

IC42567 - BIDI environment - Screen reverse not working.

IC42573 - Migrated HATS applications (V4 to V5) have corrupted transformation .JSPs if they contained Scandinavian characters.

IC42639 - Selected menu items change after pressing Enter.

IC42686 - Symbols render incorrectly after applying CSD3.

IC42725 - Partial screen rendered with FAST5250 enabled.

IC42767 - Numbers are left justified on subfile screen.

IC42799 - Pressing Field Exit appends trailing 0 or space appends 0 or space after typing n-1 characters in a field of length n.

IC42926 - HATS5 Studio is inconsistent in preview rendering of function keys between component and widget views.

IC42967 - Input of KANA(SBCS-KATAKANA) into W-FIELD(KANA) disappear.

IC42970 - DBCS keyboard input into DBCS-only fields of = - period and other characters & % \$ are OK.

IC43180 - Browser focus lost when the HATS application has the applet enabled and is connecting using Java 1.4.

IC43249 - Graph widget not rendering negative numbers correctly.

IC43335 - S-JIS character EBCDIC 0X43A1 not being correctly displayed.

IC43344 - Internal macro error - "Invalid values are used or actions exist after a Playmacro action in a same screen" message on Save.

IC43350 - Not all input fields from multiple drop-down boxes are updated to host.

IC43409 - Drop-down menus not working.

IC43431 - Resume application from View Print Job prompts for LU.

IC43452 - Extended help pop-up from host screen not rendered.

See New Fast5250 minimum delay setting in the HATS connection file for APAR IC43452 for more information.

IC43481 - 3270 numeric only field being padded with blanks.

Additional problems

The following problems were found internally, so there is no additional information about them on the HATS support page.

28355 - DBCS: Cursor position in browser status didn't jump two columns.

29565 - Input field rendering error.

29787 - Highlighted row not rendering to end of subfile.

29832 - Popup widget in transformation preview inconsistent with full screen widget preview.

30122 - SelectList: String before leading token not working.

30130 - Get error message on the Task view after adding the Hatsportlet war file to the module.

30187 - 3270 print doesn't work in Portal environment.

30194 - SUBFILE: Error message displayed when editing the subfile component in default rendering.

30214 - Dropdown does not auto submit when used with input component.

30232 - Unexpected data in preview window when codepage=1399.

30264 - Do not cache macros while running on server in Studio.

30294 - Maintenance does not get applied to portlet projects.

30305 - Subfile recognition with field table; second record of subfile not rendered.

30311 - BIDI text replaced in field widget is clipped.

30322 - <Condition> tag not working in macro screen descriptors.

30361 - Applet not working in vertically cloned runtime environment.

30385 - The visual table component is recognized as blank.

30405 - Hebrew text in Project Settings is saved incorrectly.

30455 - Show cursor position in status window enabled, but status window is not displayed in transformation.

30486 - Can't move actions up when editing disconnect event.

30500 - disconnectOnClose after a timeout causes a new connection to be opened.

30518 - Connection name conflict: Default name incorrect.

30598 - Don't apply global rules OR text replacement to default_transformation.

30600 - Add "nowrap" to field widget for word-by-word alignment.

See New setting for the field widget for defect 30600 for more information.

30691 -
 tag shown in widget preview for subfile.

30735 - HATS connection timeouts are not occurring soon enough.

If a connection definition has set the maximum busy time or the maximum idle time, HATS periodically scans all such connections to see if any should be terminated. Depending on the value of the times chosen, this periodic scan might not run as frequently as desired.

HATS has been modified to scan at least once per minute for connections that are eligible to be terminated.

30781 - Select Item popup only updating first input field on host using the Konqueror browser.

30782 - Macro: Unable to create IO with If/Else statement.

30819 - Timed-out clients should get a disconnected page.

HATS no longer creates a new legacy host connection and a new HATS application after the original application has timed out. Instead, the end user receives a page indicating the application has disconnected due to inactivity. This page offers a button to restart the application, if desired.

30834 - SBCS: The "Use project defaults" check box becomes unchecked in component settings for default rendering item.

30908 - XSLT Style Sheets not applied on server using Web Services.

30929 - Subfile: subfileActionFieldStartCol function not working.

30950 - Migrated a HATS V4 application through CVS and received an Error.

30984 - Mozilla sends unchanged input fields to the server.

Mozilla sends more fields to the server than were actually modified. This may cause some problems for legacy applications, such as TSO. This fix only allows modified fields to be sent to the server.

Problems fixed in Service Pack 3

See the HATS support page for a more detailed description of these APARs:

IC41065 - Menu has Arabic and English listings and the menu items on the right do not line up correctly.

See New setting for the field widget for APAR IC41065 for more information.

IC41371 - If there is an underscore in data entry field, HATS will not accept data being entered.

IC41415 - Admin only projects fail to start when using WebSphere Test Environment V5.1

IC41471 - No error is detected when a SBCS is entered into a DBCS only field.

IC41550 - PF15 Key displays PF51 on the status line in browser.

IC41577 - Pressing Enter multiple times causes a JavaScript error.

IC41631 - HATS global variable values not available in PORTAL mode.

IC41664 - After application of HATS Refresh Pack 2, tabbed folders are not handled properly in screens containing DBCS characters when the asynchronous update applet is enabled.

To fix existing projects that contain tabbed folders, edit the source of the JSP, and insert the <!--refresh--> tag into the JSP, following this line:

```
<!-- TabbedFolderportletxxxxxx TabbedFolder start -->
```

IC41696 - Subfile header not recognized by color.

IC41724 - Enter key not permitting exit from 5250 screen after error occurs.

IC41777 - Last AID key in row not recognized.

IC41829 - Applet enabled single checkbox becomes all on enter.

IC41990 - Fields with hints are incorrectly rendered when used in default rendering rule.

IC41996 - DBCS data can be entered into an A type field and no error is displayed.

IC42167 - DBCS characters adjacent to host URL component not rendered correctly to link widget.

IC42227 - Field minus/field plus keys not working properly.

IC42366 - Selection list rendering incorrect when there are input fields in the list items.

See New setting for the selection list component for APAR IC42366 for more information.

IC42372 - Input data length is over limit of host definition and host rejects it.

IC42516 - VIF loading from HATS Portlet problems: reversed data, visual input field not working correctly, no OIA.

Additional problems

The following problems were found internally, so there is no additional information about them on the HATS support page.

29285 - DBCS would display twice on subfiles after pressing the Enter key.

29373 - The options should not be shown in the header of a subfile.

28392 - Intermittently, the application server won't stop after running a HATS application.

29432 - A submit button is not needed for the dropdown of a HATS V4 migrated application.

29476 - Provide HATS portlet support in a Web Service Remote Portlet (WSRP) configuration.

29502 - HATS portlet can't be tested in Portal test environment.

29543 - The button widget is not working in a HATS project using the 424 Hebrew codepage.

29547 - After editing a global rule to modify the component and widget settings, the settings are not used when you run the application on the server.

If a user manually modifies the source of a JSP fragment associated with a global rule to change the component and widget settings, the global rule should not be edited using the Edit button on the Global Rules page of the project settings Rendering tab. If the global rule is edited with the button, the wrong tag is read and any manual modifications are erased.

29548 - With the applet enabled, host refresh is not working on Mozilla 1.6 and Internet Explorer.

29698 - With the applet enabled, the Show URL action for a screen customization does not display the specified Web page.

29755 - Subfile rendering doesn't consume the first column in host screen.

29829 - Visual table and field table components are displaying hidden fields.

29849 - Header row style class not being applied in macro handler.

29882 - Continuous Internet Explorer Script Errors when inserting tabbed folder.

30006 - WEL parameter default is incorrect.

30021 - Integration Object creation does not work with macro IF or ELSE statements.

30065 - When the Asynchronous Update applet is enabled in Mozilla, the host connection is intermittently disconnected.

Archived

Lab 1: Creating a HATS application

This lab lead the reader through the creation of a base Host Access Transformation Services (HATS) application. The exercise also illustrates some of the more common customizations that can be made to legacy applications using HATS.

Note: Updates can be found at:

<http://websphere.dfw.ibm.com/whidemo>

The lab assumes that WSSD v5.1.0 or later and HATS v5.0.4 are installed on a PC meeting the following listed in Table C-1.

Table C-1 Requirements

Operating system	Software	Hardware
Linux	Red Hat 7.2 or 8.0, or SUSE 7.2 or 8.1. Web browser: Netscape Navigator 4.6 or 6.0, or Mozilla v0.7 or higher. TCP/IP installed and configured.	Intel® Pentium® II processor minimum; Pentium III 500 MHZ or higher recommended. XVGA (1024 x 768) display minimum. 512 MB RAM minimum; 768 MB RAM recommended. Disk space requirements: 1.3 GB minimum for installing

Operating system	Software	Hardware
Windows® 2000, Windows XP	Windows 2000 Professional SP 2 or higher, or Windows XP Professional SP 1 or higher. Web browser: Microsoft® Internet Explorer® 5.5 SP 1 or higher, or Netscape Navigator 4.76 or higher. TCP/IP installed and configured.	Intel® Pentium® II processor minimum; Pentium III 500 MHZ or higher recommended. SVGA (800 x 600) display minimum (1024 x 768 recommended). 512 MB RAM minimum; 768 MB RAM recommended. Disk space requirements: 1.3 GB minimum for installing

This set of laboratory exercises will provide you with hands-on experience with HATS. The base exercises cover:

- ▶ Default project
- ▶ Create project
- ▶ Test project
- ▶ Work with the template and stylesheets
- ▶ Change project settings
- ▶ Test template and project settings changes
- ▶ Screen customizations
- ▶ Screen customizations summary
- ▶ Welcome screen
- ▶ Sign-on Complete and blank screens
- ▶ CICS MENU transaction Operator Instructions screen
- ▶ CICS MENU transaction File Browse screen

Extra effort laboratory exercises:

- ▶ CICS MENU transaction File Inquiry screen
- ▶ CICS MENU transaction File Inquiry Error screen
- ▶ CICS NACT transaction Accounts Menu screen
- ▶ CICS NACT transaction Details of Account screen
- ▶ CICS NACT transaction Accounts Menu Error screen

Default project

To create a project, complete the following steps:

1. Start WebSphere Studio and the HATS perspective. Click **Start -> Programs -> IBM WebSphere HATS -> HATS Studio**. For workspace name, enter `c:\myworkspaces\myhatslab` and click **OK**.
2. Launch the Create HATS Project Wizard. On the Welcome to HATS page, click **launch the Create HATS Project Wizard**, as shown in Figure C-1 on page 117.

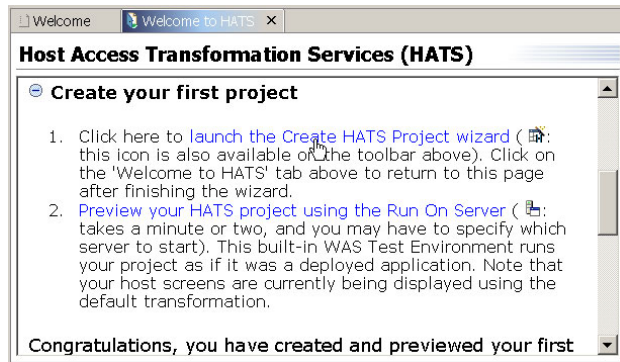


Figure C-1 Create HATS Project Wizard

3. Next, give your project a name. On the Create a Project panel, shown in Figure C-2, for Name:, enter myhats3270. Uncheck Use default Enterprise Application project. For Enterprise Application project name, enter myhats3270.ear. Click **Next**.

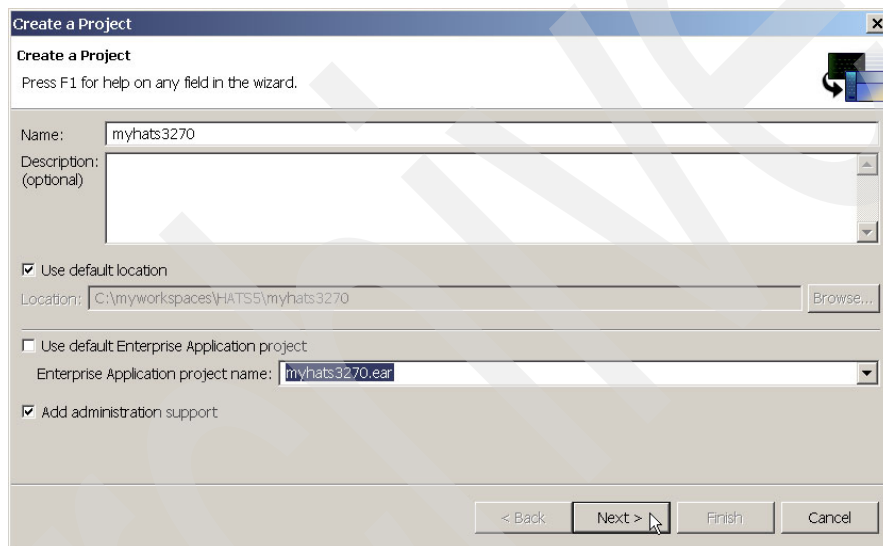


Figure C-2 Create project - name

4. Now set your host site. On the Connection Settings panel, shown in Figure C-3 on page 118, for Host name: enter zserveros.dfw.ibm.com. For Type: select **3270**. Click **Next**.

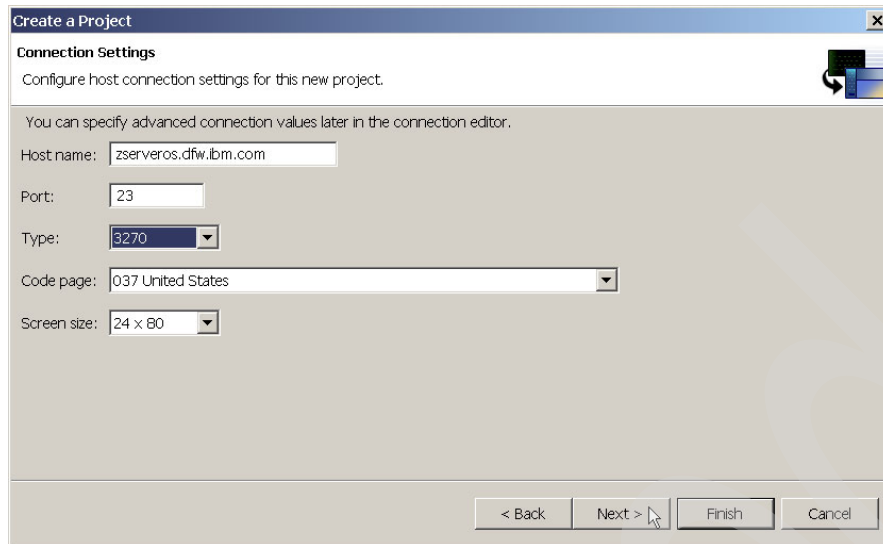


Figure C-3 Create project - settings

5. Select your default template. On the Select Default Template panel, from the Template pull-down, select whichever template you want to use. This example will use Sports.jsp, as shown in Figure C-4. Click **Finish**.

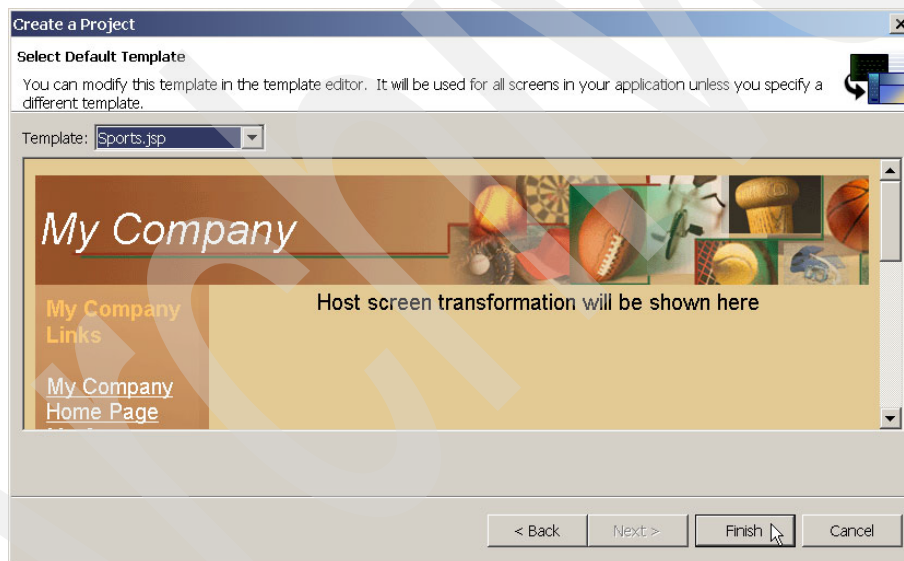


Figure C-4 Create project - default template

6. After finishing, the Project Settings Overview is displayed as shown in Figure C-5 on page 119. Click the **X** on the editor's settings tab to close this view for now. You will go through it later in the lab.

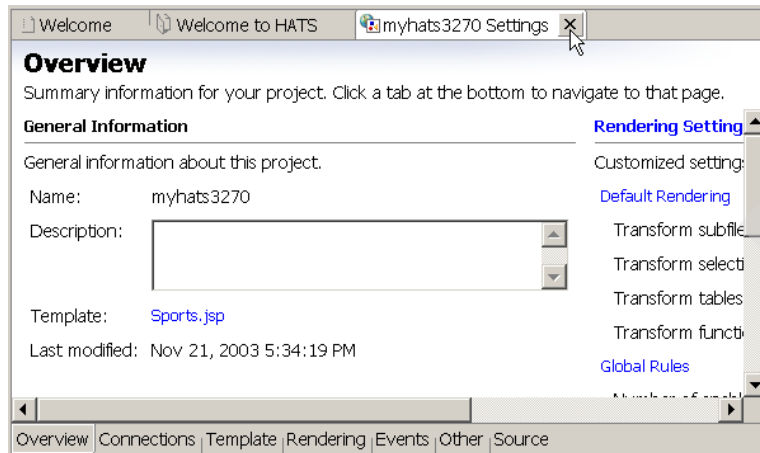


Figure C-5 Create project overview

In the HATS project view, notice your project, including all of its folders. You will be looking into these folders as you go through this lab.

Test project

1. In the HATS project view, right-click your project's folder and select **Run on Server**; see Figure C-6.

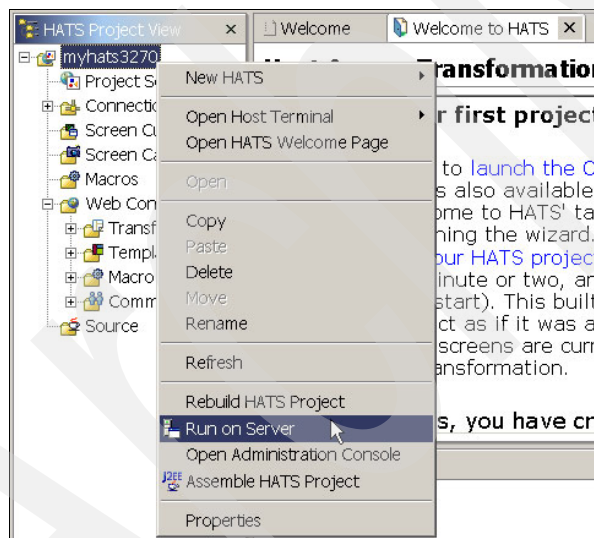


Figure C-6 Run on Server

2. On the Server Selection panel, as shown in Figure C-7 on page 120, create a new server, select **WebSphere version 5.0 Test Environment**. Check Set server as project default. Click **Finish**. In the Web Browser view, see your HATS project running live with your host system.

To maximize the Web Browser, double-click the **Web Browser** tab. Minimize it by double-clicking it again.



Figure C-7 Integrated Web browser

Click one of the links in the left panel of the template. Note that if you are doing the offline version of this lab, that is, if you are not connected to a real network, then this link will not work. Click the **Back** button to get back to the host screen. In the Modify the template section, you will see how to edit the template.

At the bottom left of the host screen you see five buttons, starting with the Reset button. This is the application keypad, as shown in Figure C-8. At the bottom of the host screen, notice that there are no buttons visible for host keys. This is because the host keypad is not displayed, by default. In the Change project settings section, you will see how to change these global project settings.

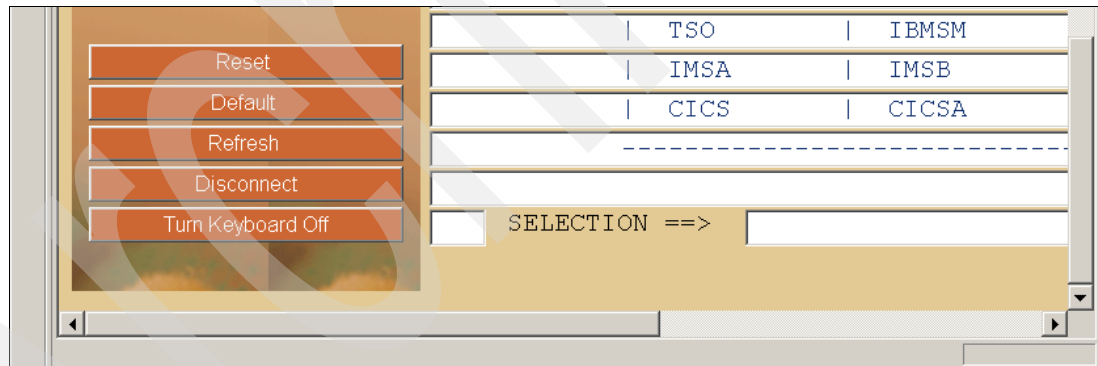


Figure C-8 Application key pad

Work with the template and stylesheets

If not already there, switch to the HATS Project View by clicking the **HATS Project View** tab at the bottom of the left frame. In the HATS Project View, expand the Web Content\Templates folder. Double-click the template you selected as your default; see Figure C-9 on page 121.

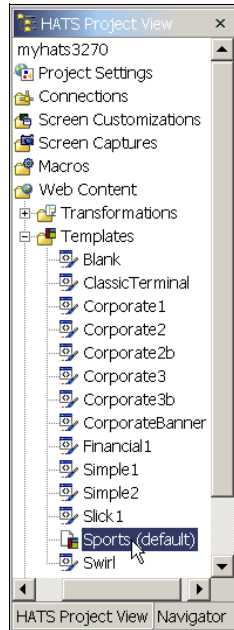


Figure C-9 Project view - select template

This brings up your template in an editor, as shown in Figure C-10. Notice, at the bottom of the editor view, the tabs Design, Source, and Preview. Click these tabs to see what happens. You can make your changes in either the Design view or in the Source view.

Using the Design view, change the company name by clicking the text My Company and then typing in whatever company name you wish. This example uses JK Enterprises.

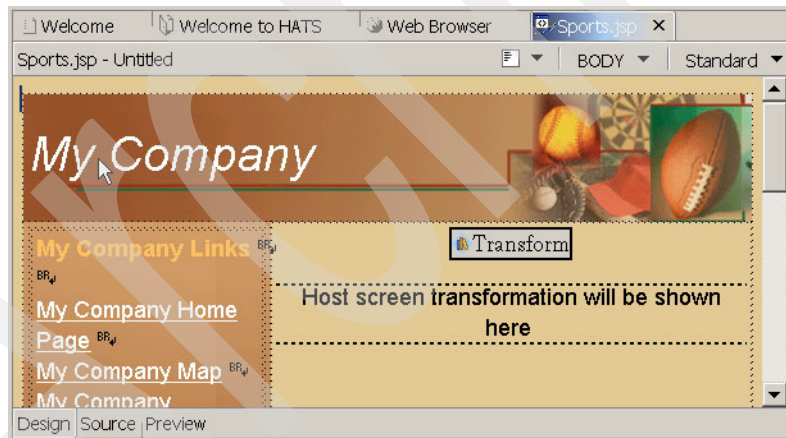


Figure C-10 Design view - edit template

In a similar manner, change any other text you wish; for example, change the other occurrences of My Company.

Next, make a change to one of the links on the template, as shown in Figure C-11 on page 122. With the right mouse, select a link (for example, the Home Page link) and then select **Attributes**. Notice the Attributes view that is displayed at the bottom of the workbench.

Scroll down and change the URL to some other site you know and for Target, select **New Window**. This will open the URL in a different window from your host system window. (If you

are using the offline version of this lab, that is, you are not connected to a network, then your link will not work.)

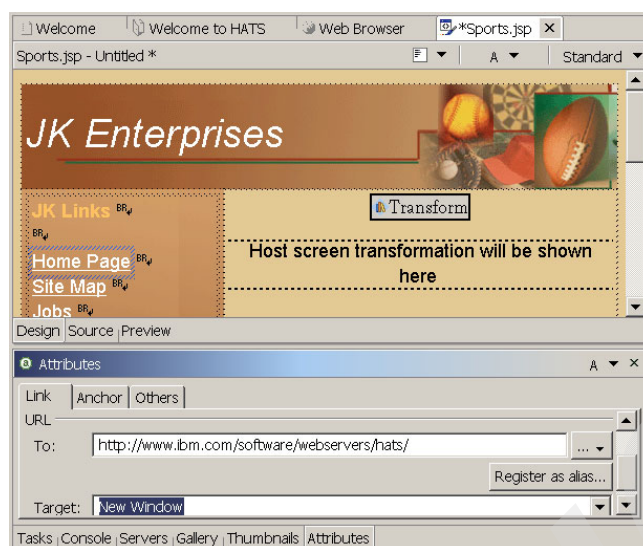


Figure C-11 URL link target settings

You can make any other changes to your template that you desire. You may find it easier to maximize the editor view by double-clicking the tab for your template's JSP. As an example, click the **Source** tab and change the stylesheet for the template to one that will use a white background. On the LINK statement, change `tantheme.css` to `whitetheme.css`, as shown in Figure C-12.



Figure C-12 Template source view

Also, you may find it convenient to create your own stylesheet with your own personal styles and import it into your project. For this lab, two personal stylesheets have been created for you. These stylesheets contain some overrides to the HATS default styles and some styles for text you add yourself. To import them, follow these steps (refer to Figure C-13 on page 123):

1. From the toolbar, select **File -> Import**.
2. On the Import-Select screen, select **File System** and click **Next**.
3. On the Import-File System screen:
 - a. For From directory, click **Browse** and browse to `c:\temp` (if you are using your own machine, browse to whatever folder you downloaded the stylesheets into).
 - b. In the left panel, highlight the temp folder (do not check the check box).
 - c. In the right panel, check the check boxes for each of the stylesheets.

- d. For Into folder, click **Browse** and browse to myhats3270\WebContent\common\stylesheets. Highlight the stylesheets folder and click **OK**.
- e. For Options, check only **Create selected folders only**.
- f. Click **Finish**.

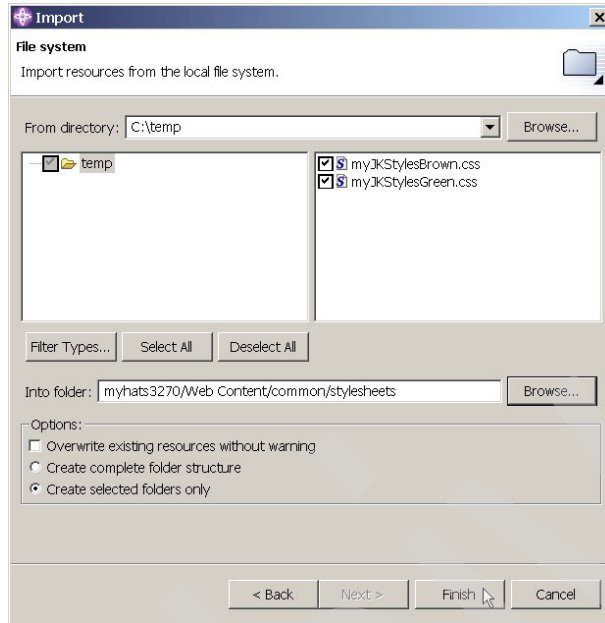


Figure C-13 Import style sheet wizard

After importing these stylesheets into your project, in order to use either one of them, you must link to it from your template. In the source for your template (in this case, Sports.jsp) as shown in Figure C-14, add the following LINK after the last </STYLE> tag:

```
<LINK rel="stylesheet" ref="../../common/stylesheets/myJKStylesBrown.css" type="text/css">
```



Figure C-14 Sports.jsp source

Look at the stylesheet you just added by bringing it up in an editor, as shown in Figure C-15 on page 124.

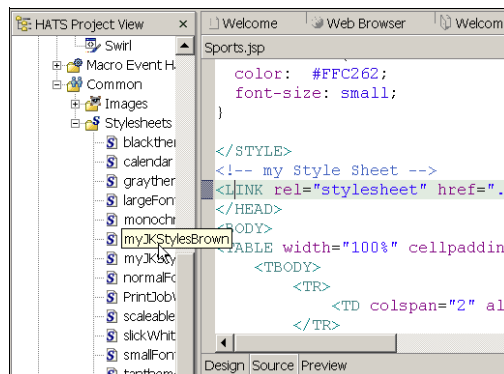


Figure C-15 myJKStylesBrown.css

Find the stylesheet in the Stylesheets folder and double-click it; see Figure C-16.



Figure C-16 myJKStylesBrown.css - close up

All of these style classes could have been modified or added directly into the template. However, in this lab we kept all the personal changes separate from HATS default styles. Close the editor for your new stylesheet.

When you have finished making changes to your template, close the editor view by clicking the **X** on the JSP tab and clicking **Yes** when asked if you want to Save changes; see Figure C-17.

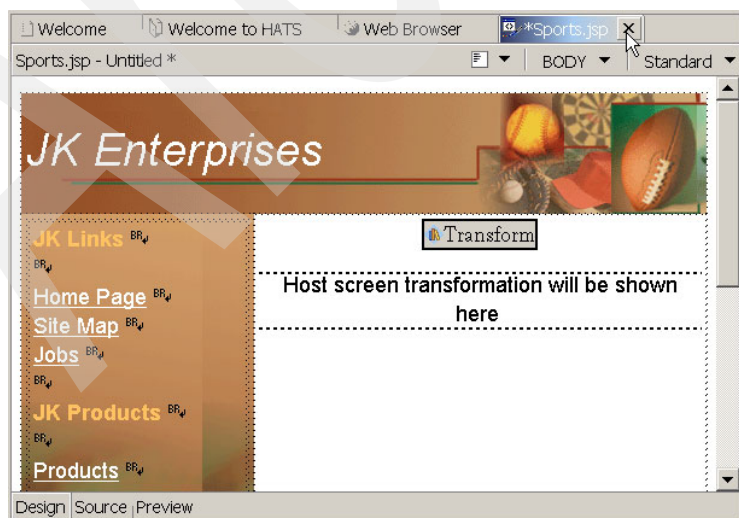


Figure C-17 Close JSP editor

Change project settings

1. In the HATS Project View, double-click Project Settings; see Figure C-18.

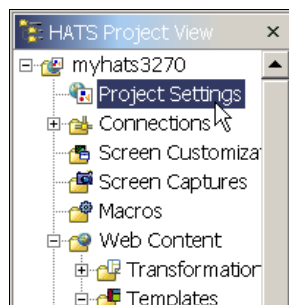


Figure C-18 Open project settings

2. Maximize the Project Settings view and click through all of the different tabs at the bottom to see the various items that can be set.
3. When you are finished, click **Rendering**.
4. From the Rendering view, you can see all of the default rules that HATS uses to recognize and transform screens. In the left panel, expand the list of Components, as shown in Figure C-19.

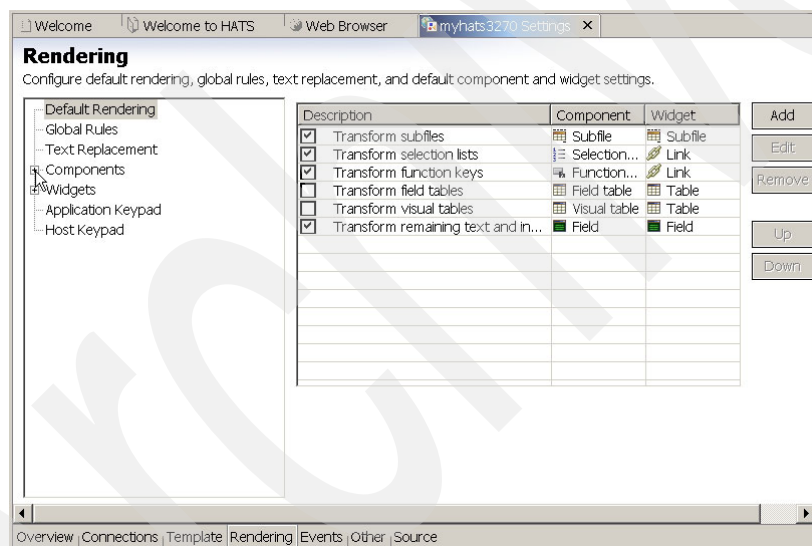


Figure C-19 Rendering panel

These are the components that HATS can recognize on a host screen and the default rules for recognizing them, as shown in Figure C-20 on page 126. For example, notice the default rules for how HATS will recognize an area of the screen to represent a function key.

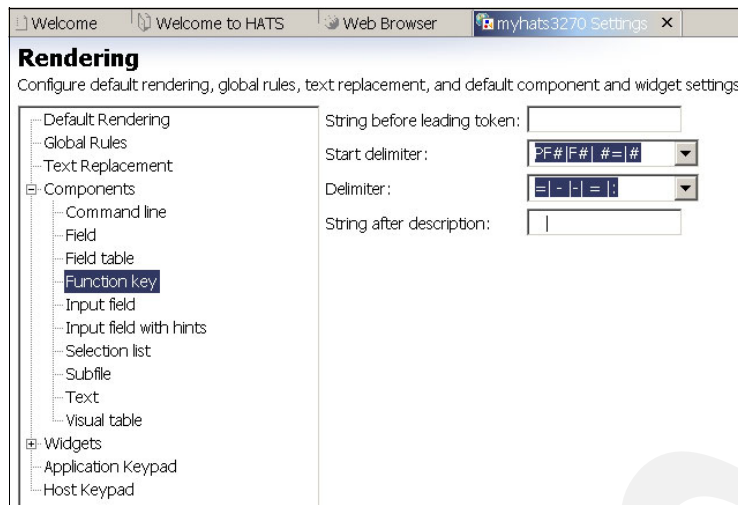


Figure C-20 Rendering panel - component detail

When you finish looking at the various Components, expand the list of widgets that can be used to render a graphical view of a host component, as shown in Figure C-21. Notice the default definition of how a drop-down (data-entry) widget can be displayed. You will use this widget later in this lab.

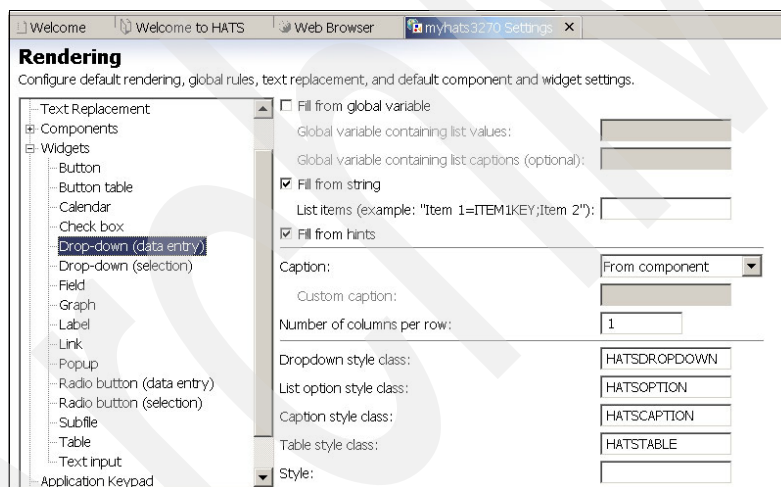


Figure C-21 Rendering panel - drop down widget

5. When you are finished looking at the various Widgets, click **Application Keypad**.

Earlier in this lab you saw the application keypad that is displayed by default. Here you can change what keys to display and whether to display keys as buttons or links, as shown in Figure C-22 on page 127. Uncheck Turn Keyboard On/Off so that this key will not appear on the application keypad.

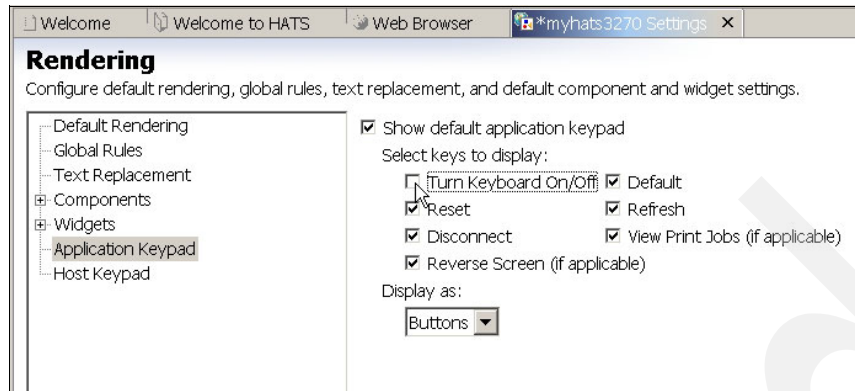


Figure C-22 Application keypad customization

6. Click **Host Keypad**. Notice that by default the host keypad is not displayed. Here is where you can tell HATS to display the host keypad and what keys to include. For this lab, use the default to *not* show the host keypad.
7. Close the Project Settings View by clicking the **Sand** click **OK** to save your changes; see Figure C-23.



Figure C-23 Save rendering customizations

Test template and project settings changes

1. Switch back to the Web Browser view, scroll down and click the **Refresh** button on the application keypad, as shown in Figure C-24. If you get the Disconnected screen, click Restart. Notice that the Turn Keyboard On/Off key is no longer displayed on the application keypad.



Figure C-24 Customized application keypad

Try the link you changed on the template, as shown in Figure C-25 on page 128.

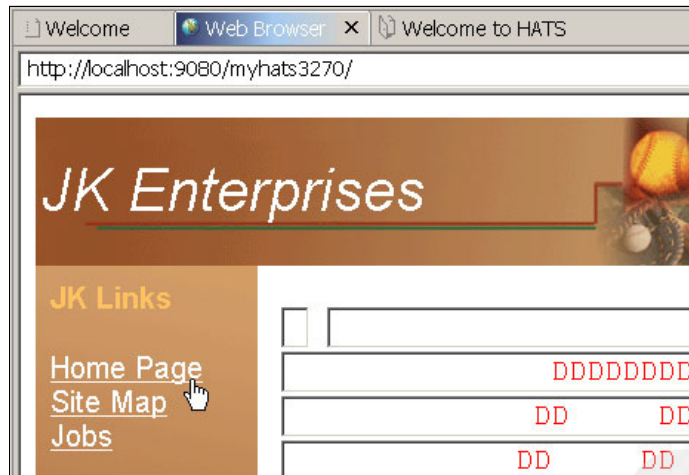


Figure C-25 Customized URL link

2. Notice that the new site opened in a new window, as shown in Figure C-26. Close the window to the new site.

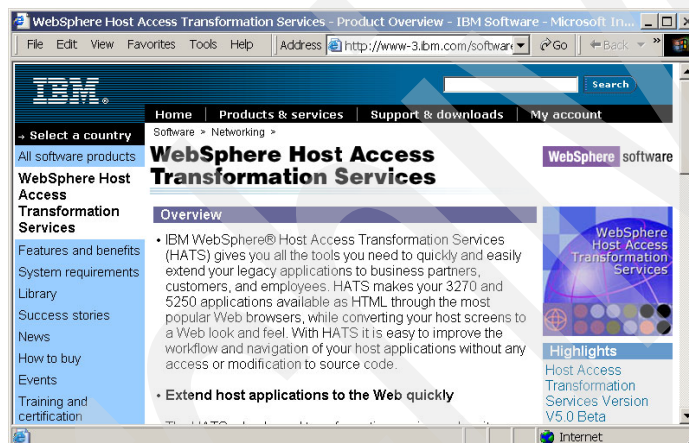


Figure C-26 Customized link - new window

3. Minimize the Web Browser view in the studio.

Screen customizations

In the following sections are exercises for customizing screens from our host application. HATS is very flexible regarding the method you use to customize screens. One method would be to capture all of the screens you want to customize and create all of the macros you want to use first, then go back and customize your screens.

Another method is to capture a screen, record any macros necessary for the screen, and then customize the screen before going on to the next screen. In this lab, you will use the second method so that you can have a more self-contained exercise for each screen customization.

Note that you do not need to do all of the exercises to get a basic understanding of HATS. However, the more you do, the more functions you will learn. Following is a summary of what

HATS functions are used for each screen customization. You can scan this summary and find the ones that cover the functions that are of most interest to you.

Screen customizations summary

The following is a summary of the screen customizations you will complete.

Welcome screen

In this exercise you will:

- ▶ Create a screen capture
- ▶ Record a macro
- ▶ Create a screen customization that applies a transformation
- ▶ Create a transformation using:
 - Text you supply
 - Macro button

Sign-on complete and blank screens

In this exercise you will:

- ▶ Create a screen capture
- ▶ Create a screen customization that:
 - Applies to two different screens
 - Applies a transformation
- ▶ Create a transformation using:
 - Text you supply
 - Input field component with drop-down list widget
 - Individual host key buttons

CICS MENU transaction operator Instructions screen

In this exercise you will:

- ▶ Create a screen capture
- ▶ Create a screen customization that applies a transformation
- ▶ Create a transformation using:
 - Text you supply
- ▶ HATS components within a table:
 - Input field component with drop-down list widget
 - Input field component with text input widget
- Individual host key buttons

CICS MENU transaction file browse screen

In this exercise you will:

- ▶ Create a screen capture
- ▶ Record a macro that will loop through multiple screens, extract data from each screen, and display the data in a table

- ▶ Create a screen customization that automatically plays the macro above
- ▶ Modify the default transformation used by the macro to display the data

Welcome screen

Create a screen capture

1. In the HATS Project View, right-mouse click your project and select **Open Host Terminal** -> **Main** as shown in Figure C-27.

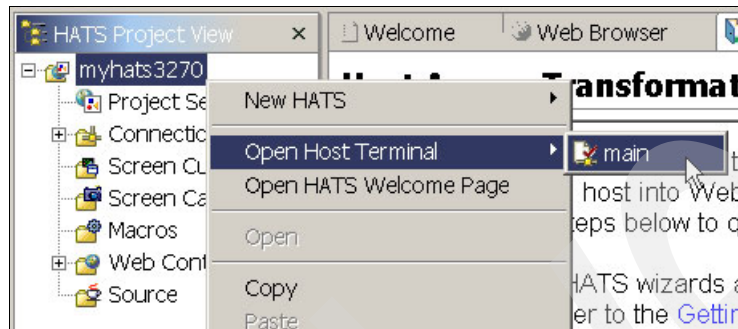


Figure C-27 Open Host Terminal

2. You will see the initial Welcome screen from the host system. Take a screen capture of this screen. On the toolbar, click **Create Screen Capture**. In the Capture a Screen panel, name this screen capture as **Welcome**, then click **Finish**; see Figure C-28.

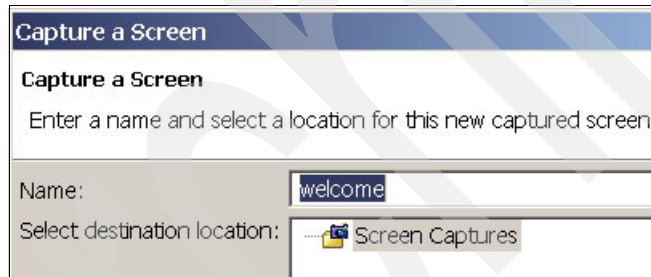


Figure C-28 Name captured screen

Record a macro

HATS macros can be played either by clicking a button or automatically when a screen from the host is recognized. In this case, you will record a macro that will select an application and sign on the user with a generic userid. Of course, if the application required that a user sign on with a unique userid, you would not create this signon macro. But in this example you are using a generic userid that has been set up to allow read only access to just a subset of the applications and transactions on the host.

1. To start recording the macro, click **Record Macro** on the toolbar.
2. On the Record Macro panel, name this macro, sign on, and click **Finish**; see Figure C-29 on page 131.

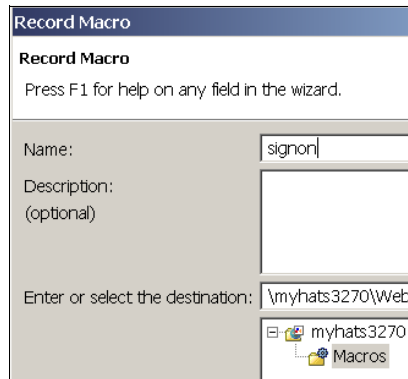


Figure C-29 Name new macro

3. On the Define the starting screen of the macro panel, for Screen Name, type: welcome, select **Within a rectangular region**, and click **Finish**.
4. On the Welcome screen for SELECTION, type: cicsa and press Enter (or click the **Enter** button).
5. On the Signon to CICS screen, click **Define Screen Recognition Criteria**.
6. On the Select Screen Recognition Criteria panel, for Screen Name, type: signonToCics and click **Finish**.
7. On the Signon to CICS screen, type: whidemo for Userid type and type: guest1 for Password type. Then press Enter (or click the **Enter** button).
8. On the Sign-on complete screen, click **Define Screen Recognition Criteria**.
9. On the Select Screen Recognition Criteria panel, for Screen Name, type: signonComplete. With your mouse, draw a box around the text Sign-on is complete at the bottom of the screen, select **At a specified position**, and click **Finish**.
10. On the Sign-on complete screen from the toolbar, click **Stop Macro**.
11. On the Define the exit screen of the macro panel, click **Finish**. On the Sign-on complete screen, click **Save Macro**.

In the Macro Navigator, notice that the macro that has been created. At this point you may want to test your macro. To test the macro, navigate back to the Welcome screen from the host. Do this by either typing `cesf logoff` on the host screen and pressing Enter, or disconnecting and reconnecting the host session from the icons on the host terminal toolbar. After returning to the welcome screen, from the Host Terminal toolbar select the **Play Macro** drop-down and select the **signon** macro.

You could continue to capture more screens and, after capturing all the screens you want, start customizing them. However, these exercises are written so that the work for each screen is a self-contained exercise. So after your macro is saved, click the **Disconnect** button on the toolbar to disconnect the terminal session and close the terminal window.

Create screen customization

You have now created a screen capture for the Welcome screen and a signon macro. Next, you will create a screen customization for this screen. A screen customization consists of how to recognize the screen and then what actions to perform once the screen is recognized.

1. Notice in the Screen Captures folder there is a screen capture for the Welcome screen. Right-mouse click the Welcome screen capture and select **New HATS -> Screen Customization** as shown in Figure C-30 on page 132.

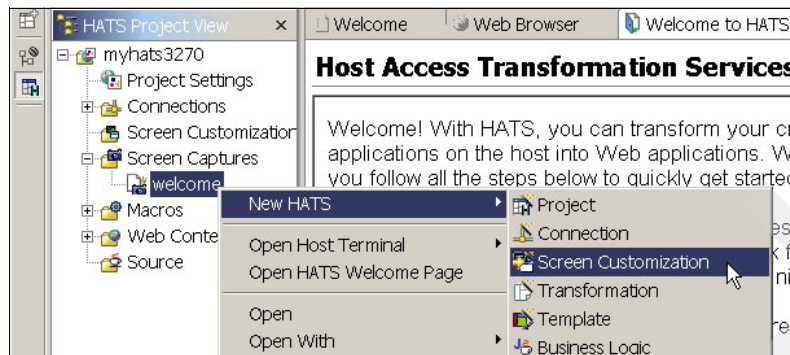


Figure C-30 Create screen customization

2. Accept the supplied screen customization name of welcome and click **Next**; see Figure C-31.

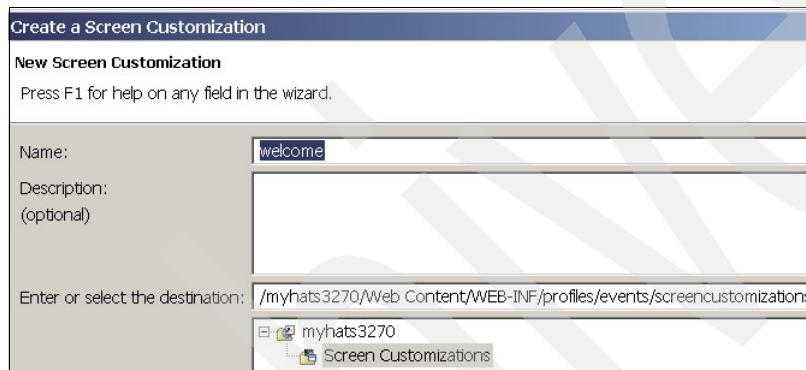


Figure C-31 Name screen customization

3. The first task in creating a screen customization is to tell HATS how to recognize the screen. On the Select Screen Recognition Criteria panel, accept the default recognition criteria and click **Next**.
4. After you tell HATS how to recognize the screen, then you tell HATS what to do once the screen is recognized. On the Select Actions panel, you see that you can Apply a transformation, which means to display the screen to the end user in a transformed way as shown in Figure C-32 on page 133.

You can tell also HATS to play a macro or to perform Advanced Functions when this screen is recognized. In the case of this example, however, simply check **Apply a transformation** and click **Finish**.

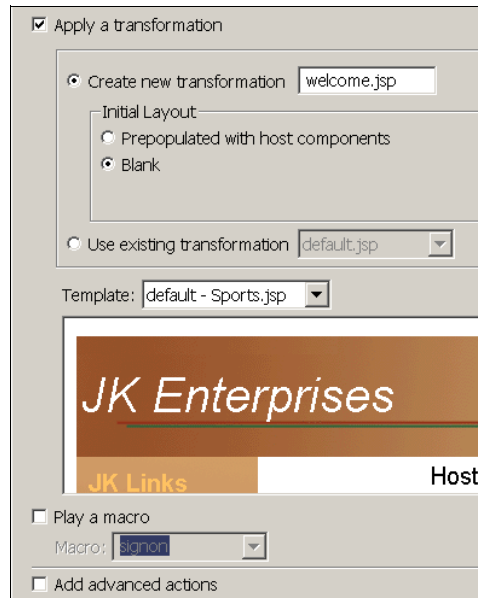


Figure C-32 Configure transformation

Create screen transformation

You are now in the Screen Transformation wizard. This first panel, **Select Screen Region**, is where you would select the first component from the host screen to display in a transformed way to the end user. However, in the case of this example, you will not display any of the components from the host screen. So, simply click **Cancel**.

You are back to the workbench view. Notice in the HATS Project View that there are now entries for the Welcome screen in both the Screen Customization and the Web Content\Transformations folders. Also notice the transformation for the Welcome screen, `welcome.jsp`, is displayed in the editor view.

1. Click the Design and Source tabs to see the different views, as shown in Figure C-33. Before you begin creating a transformed look, you may want to maximize the editor view. Do this by double-clicking on the tab for `welcome.jsp`.



Figure C-33 Open `welcome.jsp`

2. Now you will create a transformed look for the Welcome screen. Start by putting some text on the screen to identify the host system to the end user, as shown in Figure C-34 on page 134. Place your cursor just before the representation of the HATS Form tag. This is most easily done by selecting the **Form** tag with your mouse and then pressing the **Back** cursor key once.

3. After you have positioned the cursor, press the Enter key three or four times to insert some blank lines.
4. Type: JK Accounts (or any other title you prefer).

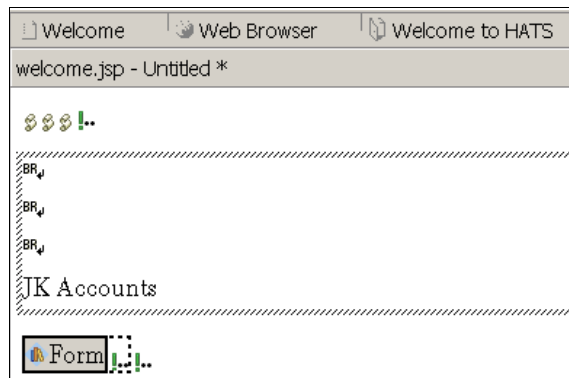


Figure C-34 Enter text into JSP

One method for changing the font of the text is to select the text and from the toolbar select **Format -> Font**. However, a more efficient method is by using stylesheets. Since you have previously imported and linked to your personal stylesheet, you can set the style of your text to match one of your style classes.

This is a better method because if you want to change the attributes of your text throughout the project, you can do so in one place by simply changing the class in your stylesheet, as described in the following step.

1. Right-mouse click your text and select **Style -> Edit Style Rule**; see Figure C-35.

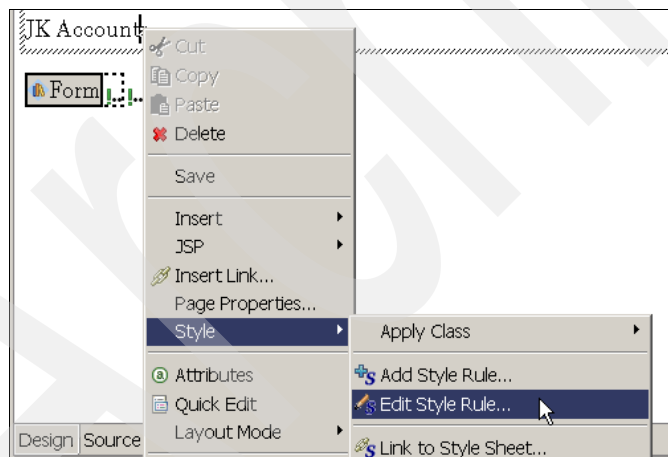


Figure C-35 Edit style rule

2. For Class name, type: myBigText as shown in Figure C-36 on page 135. This is a class name defined in the personal stylesheet you imported earlier. See how it is defined, then click **OK**.

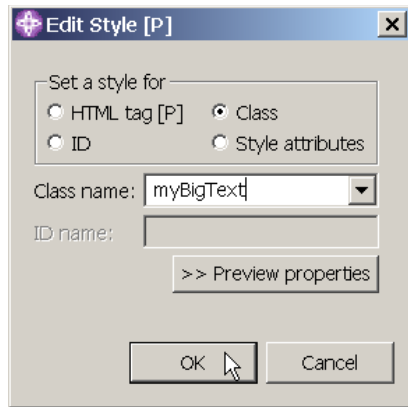


Figure C-36 Select style class name

3. On the Add Style panel, click **OK**.
4. Add a button that will play the signon macro by positioning the cursor at the end of the text you just added and pressing the Enter key to add another one or two blank lines.
5. Place your cursor following the comment after the representation of the HATS Form tag by selecting the **Form** tag with your mouse and then pressing the Forward cursor key twice.
6. From the toolbar, select **HATS Tools -> Insert Macro Key**; see Figure C-37.

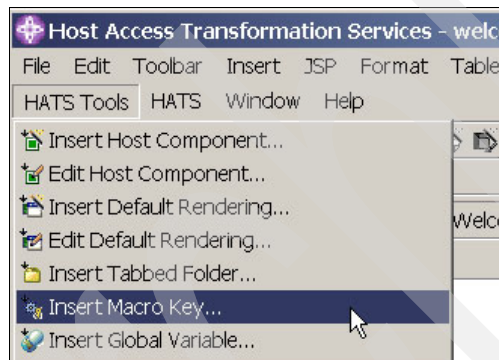


Figure C-37 Insert macro key

7. On the Insert Macro panel, check the **signon** macro and click **OK**; see Figure C-38.

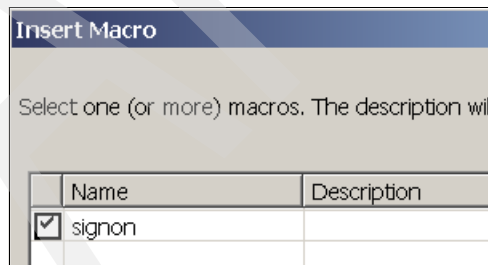


Figure C-38 Select macro

8. Change the style of the button to match the style we have planned for all of our screen buttons. Right-mouse click on the signon button and select **Style -> Edit Style Rule**, then change the Class name to HostPFKey and click **OK** as shown in Figure C-39 on page 136.

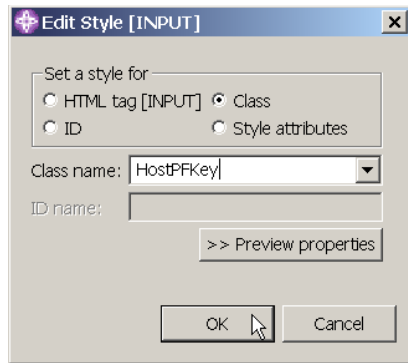


Figure C-39 Edit style for signon button

9. On the Add Style panel, click **OK**. If you want to change the label on the button, right-mouse click on the button, select **Attributes** and change the label in the Attributes view.
10. Next, center your work. From the toolbar select **Edit -> Select All**. Then, also from the toolbar, select **Format -> Align -> Horizontal Center**; see Figure C-40.

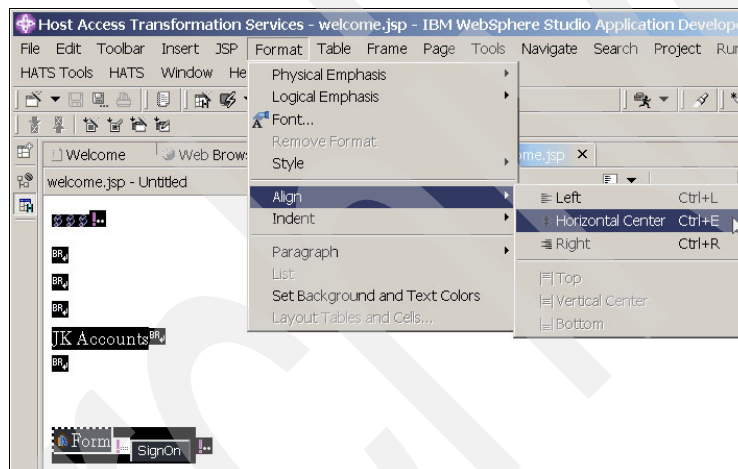


Figure C-40 Horizontally center page elements

11. You have now finished with the transformation for the Welcome screen. Close the editor by clicking the **X** and click **Yes** to save changes; see Figure C-41.



Figure C-41 Close editor and welcome.jsp

Test screen transformation

1. Switch back to the Web Browser view, scroll down and click the **Refresh** button on the application keypad. If you get the Disconnected screen, click **Restart**. Notice your transformed Welcome screen. Click the **SignOn** button to play the signon macro; see Figure C-42.

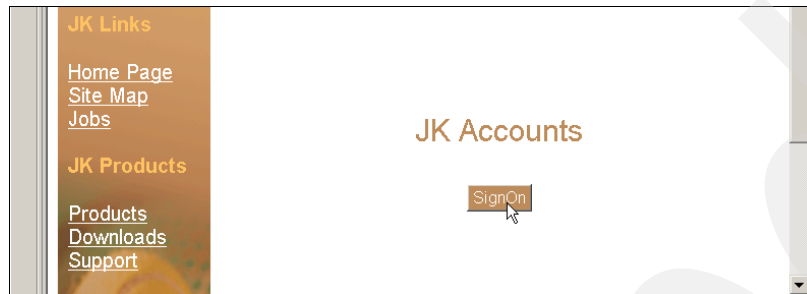


Figure C-42 New Welcome screen

The signon macro plays, which selects the CICS application on the Welcome screen and then signs on to CICS from the Signon to CICS screen. Next you see the Sign-on Complete screen, as shown in Figure C-43.

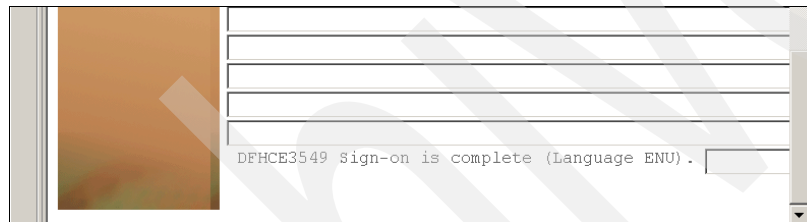


Figure C-43 Sign-on Complete

Sign-on Complete and blank screens

Create a screen capture

In the HATS Project View, right-mouse click your project and select **Open Host Terminal**; see Figure C-44.

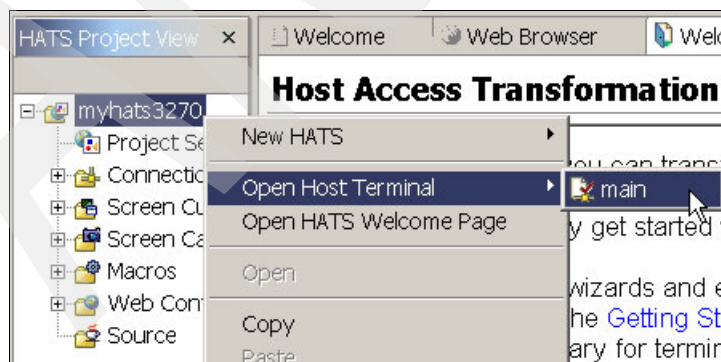


Figure C-44 Open Host Terminal

In this exercise you will create a screen customization for the Sign-on Complete screen and for the blank screen that is presented when you exit the CICS transactions. First you will use the terminal window to navigate to the Sign-on Complete screen.

On the Welcome screen, if you have already created a signon macro, then on the toolbar click the drop-down next to the Play Macro icon and select the **signon** macro. Otherwise, you can manually sign on as follows:

1. On the Welcome screen, type: `cicsa` and press Enter.
2. On the Signon to CICS screen, type: `whidemo` for userid type. Type: `guest1` for Password type, and press Enter. You are now at the Sign-on Complete screen. To create a screen capture, click the Create Screen Capture icon.
3. You will use this screen capture to represent both the Sign-on Complete and the blank screens. In the Capture a Screen panel, name this screen capture: `blank` and click **Finish**; see Figure C-45.

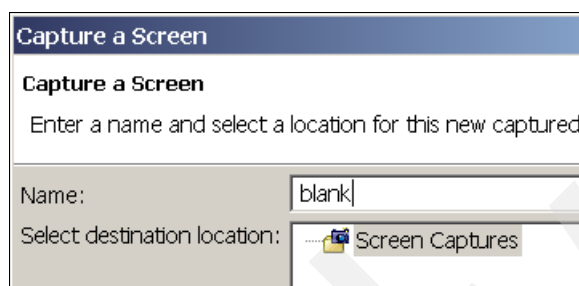


Figure C-45 Name screen capture

4. At this point you could continue to capture more screens and after capturing all the screens you want, then start customizing them. However, these exercises are written so that the work for each screen can be in a self-contained exercise. So after creating your screen capture, click the **Disconnect** button on the toolbar to disconnect the terminal session and close the terminal window.

Create Screen Customization

You have now created a screen capture for the Sign-on Complete and blank screens. Next you will create one screen customization for both screens. A screen customization consists of how to recognize the screen and then what actions to perform once the screen is recognized.

1. Notice in the Screen Captures folder you now have a screen capture for the blank screen. Right-mouse click the blank screen capture and select **New HATS -> Screen Customization**; see Figure C-46.

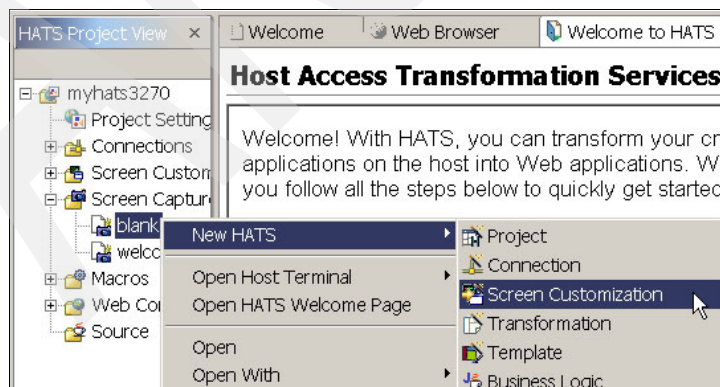


Figure C-46 Create screen customization

2. Accept the supplied screen customization name of `blank` and click **Next**; see Figure C-47 on page 139.

Create a Screen Customization

New Screen Customization
Press F1 for help on any field in the wizard.

Name:

Description: (optional)

Enter or select the destination:

myhats3270
Screen Customizations

Figure C-47 Name screen customization

- The first task in creating a screen customization is to tell HATS how to recognize the screen. You could use the Sign-on Complete text at the bottom of the screen—but using that method, this screen customization would work only for the Sign-on Complete screen and not for the blank screen.

So, on the Select Screen Recognition Criteria panel, with your cursor draw a box around as much of the blank screen at the top to recognize both the Sign-on Complete and blank screens uniquely from any other screens. Check **Within a rectangular region**, and click **Next**.

- After you tell HATS how to recognize the screen, then you tell HATS what to do once the screen is recognized. On the Select Actions panel, notice that you can Apply a transformation, which means to display the screen to the end user in a transformed way. Also, you can tell HATS to play a macro or perform Advanced Functions when this screen is recognized.

In the case of this example, simply check **Apply a transformation** and click **Finish**; see Figure C-48.

Create a Screen Customization

Select Actions
These actions will be performed when a host screen matches the criteria. You can also add or modify actions later using the screen customization editor.

☒ Apply a transformation

☒ Create new transformation

Initial Layout:

☐ Prepopulated with host components

☒ Blank

☐ Use existing transformation

Template:

JK Enterprises

JK Links Host

☐ Play a macro
Macro:

☐ Add advanced actions

Figure C-48 Configure transformation settings

Create Screen Transformation

You are now in the Transformation wizard. This first panel, Screen Region, is where you can select the first component from the host screen to display in a transformed way to the end user.

1. If not already checked, check the box **Highlight input fields**. After you do this you will notice that, except for the Sign-on complete message at the bottom, the whole screen is an input field. This is where the end user can enter a CICS transaction. In the case of this example, you will create a list of transactions from which your end users can select.

To start, select an area of the input field from the first row on the screen with your cursor. Make sure you select an area that is at least 11 characters in length so that it will contain the longest string we use in this lab. Click **Next**.

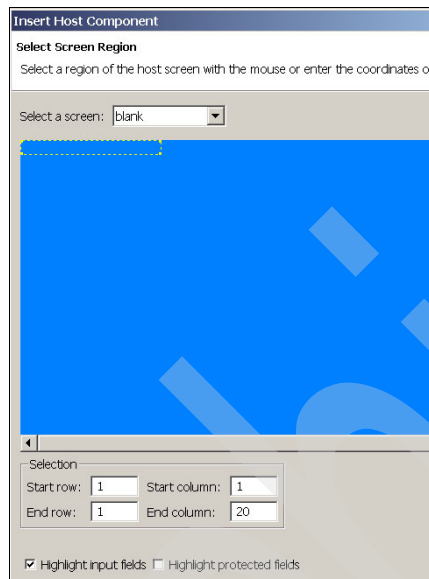


Figure C-49 Select text area from screen

2. Here you see the Rendering Options panel. This is where you tell HATS what host component to use for this area of the screen and the widget into which it will be transformed. Notice when you click each component that different widget options are displayed.

Select the **Input field component**. Notice the widgets displayed. Select the Drop-down (data entry) widget. Click the **Widget Settings** button as shown in Figure C-50 on page 141.

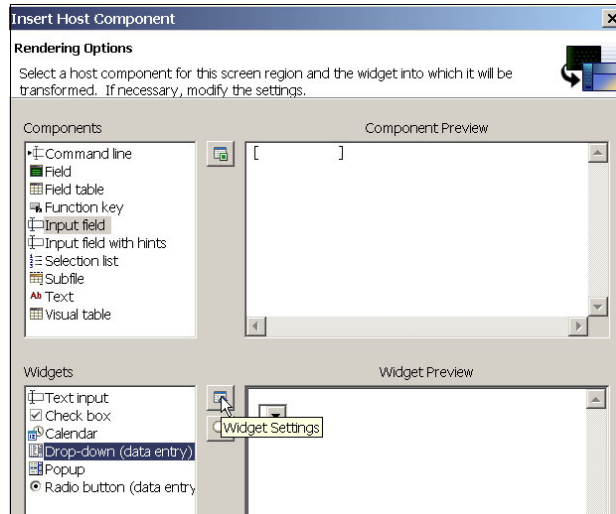


Figure C-50 Customize widget settings

3. Here is where you can change the settings of the Drop-down (data entry) widget. For this exercise do the following:
 - a. Uncheck Use project defaults.
 - b. Check **Fill from string**.
 - c. In the List items box, type: Customer Accounts=menu;Credit Card Accounts=nact;Sign Off=cesf logoff; where, for example if Customer Accounts is selected from the list, then the string menu is passed to the host. menu is the CICS transaction that displays our customer accounts. nact displays credit card accounts.
4. Notice the style classes. You can determine how these classes are defined in the template and stylesheets for the project. If you imported and linked to the personal stylesheet earlier in this lab, refer to it to see how these styles are defined. Click **OK**.

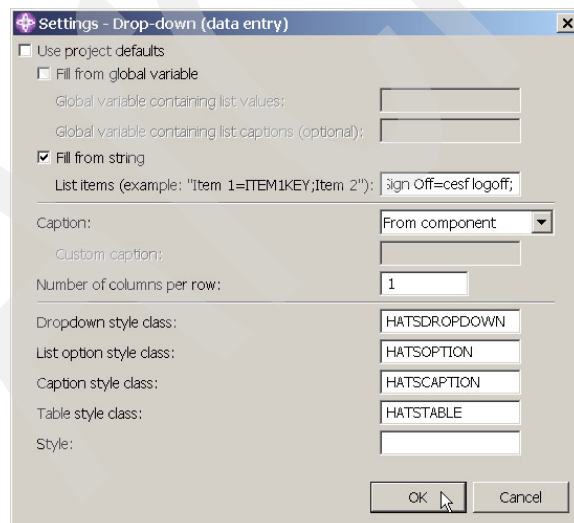


Figure C-51 Drop down widget settings

5. Back on the Rendering Options panel under Widget Preview, you can now see what your widget will look like. Click **Finish**.

6. You are back to the workbench view. Notice in the HATS Project View that there are now entries for the blank screen in both the Screen Customization and the Web Content/Transformations folders. Also notice that the transformation for the blank screen, blank.jsp, is displayed in the editor view.

Click the **Design** and **Source** tabs to see the different views; see Figure C-52.

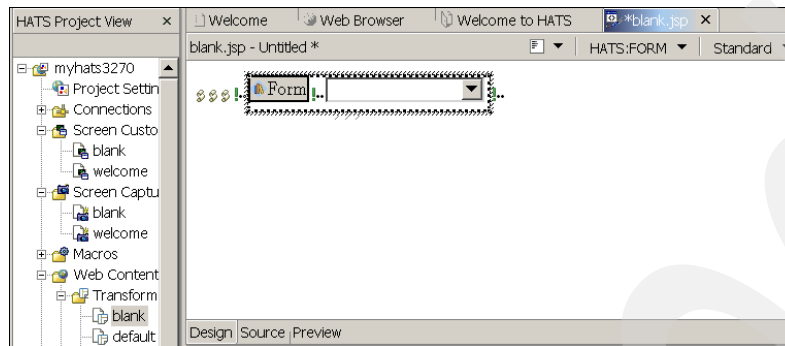


Figure C-52 Transformation design view

7. Now you will continue creating a transformed look for the blank screen. If you want to maximize the editor, double-click the blank.jsp tab. Start by putting some text on the screen to instruct your end users what to do.

Place your cursor just before the representation of the HATS Form tag. This is most easily done by selecting the **Form** tag with your mouse and then pressing the **Back** cursor key once.

8. After you have positioned the cursor, press the Enter key one or two times to insert some blank lines. Then type the following text: Select application and click Enter (or anything else you prefer). Press the Enter key once or twice to add blank lines after your text, as shown in Figure C-53.

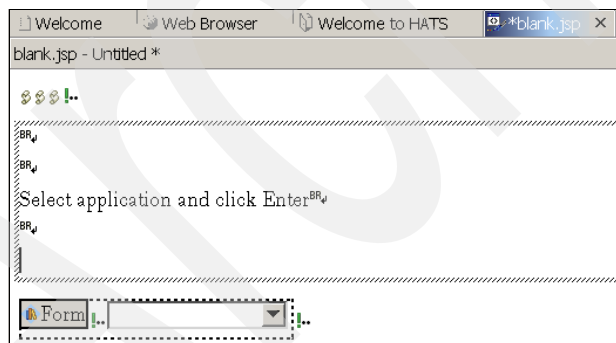


Figure C-53 Edit the blank.jsp transformation

9. One method for changing the font for your text is to select the text and then select **Format > Font** from the toolbar. However, a more efficient method is to use stylesheets. Since you have previously imported and linked to your personal stylesheet, you can set the style of your text to match one of your style classes. This is a better method because, if you want to change the attributes of your text throughout the project, you can do so in one place by changing the class in your stylesheet.

Right-click your text and select **Style -> Edit Style Rule** as shown in Figure C-54 on page 143.

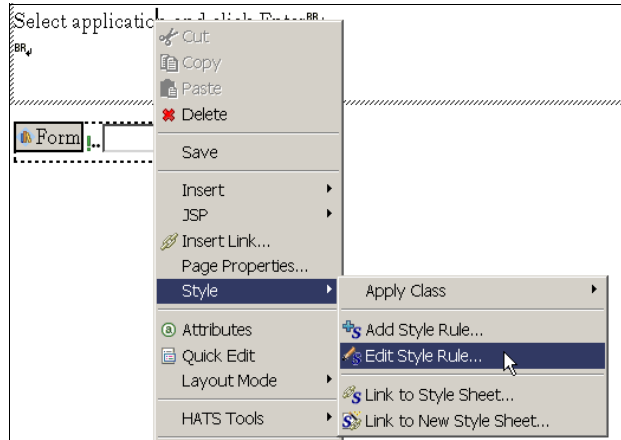


Figure C-54 Edit the style rule

10. For Class name, type: myBigText. This is a class name defined in the personal stylesheet you imported earlier. Look at how it is defined. Click **OK**.

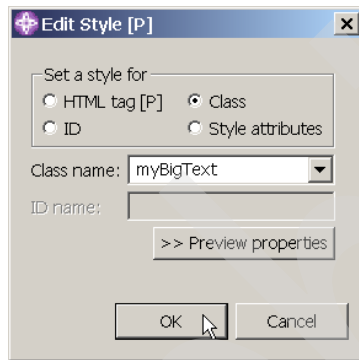


Figure C-55 Edit style class myBigText

11. On the Add Style panel, click **OK**. Add a button that, when clicked, will pass the Enter key back to the host. Position the cursor at the end of your drop-down list. This is most easily done by selecting the drop-down list and pressing the forward cursor key once.

With the cursor positioned after the drop-down list, from the toolbar select **HATS Tools** -> **Insert Individual Host key**; see Figure C-56.

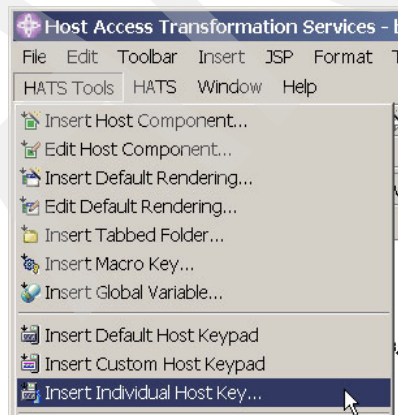


Figure C-56 Insert host key

12. On the Insert Host Key panel check Enter and click **OK**; see Figure C-57 on page 144.

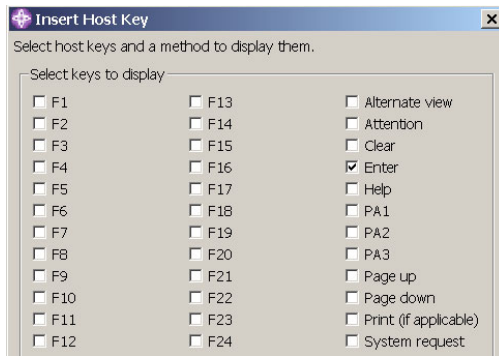


Figure C-57 Insert Host Key panel

If you would like to change the label on the Enter button, right-mouse click the button and select **Attributes**. In the attributes view, scroll down and change the label.

13. Next, center your work. From the toolbar select **Edit -> Select All**. Then also from the toolbar select **Format -> Align -> Horizontal Center** as shown in Figure C-58.

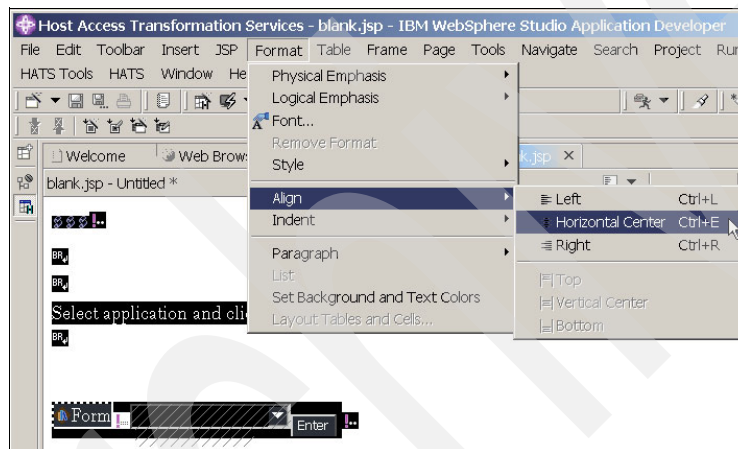


Figure C-58 Horizontally center screen elements

14. You have now finished with the transformation for the Sign-on Complete screen. Close the editor by clicking the **X** and click **Yes** to save changes.

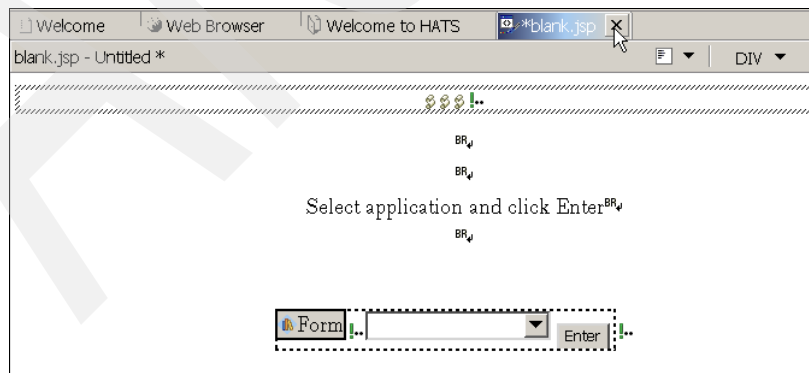


Figure C-59 Close JSP editor

Test screen transformation

Switch back to the Web Browser view, scroll down and click the **Refresh** button on the application keypad. If you get the Disconnected screen, click **Restart**.

1. Notice the transformed Sign-on Complete screen. Select **Customer Accounts** and press Enter to execute the CICS MENU transaction, as shown in Figure C-60.

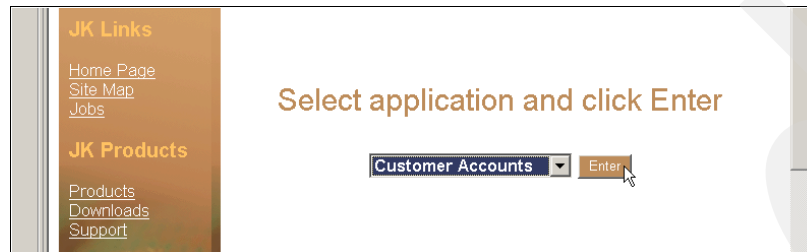


Figure C-60 New sign-on Complete screen

2. The CICS MENU transaction is executed. Next you see the MENU transaction's OPERATOR INSTRUCTIONS screen; see Figure C-61.

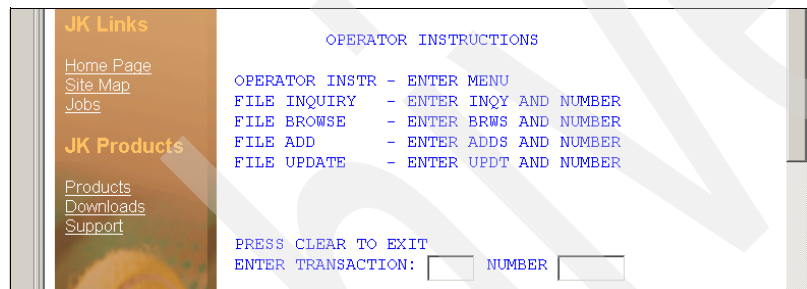


Figure C-61 CICS Menu screen

3. Now press **Clear** (the Esc key) to exit the transaction. You should see the same Select application transformation you saw for the Sign-on Complete screen, but this time for the blank screen. Click the **Default** button on the application keypad to verify that the underlying screen is the blank screen and not the Sign-on Complete screen.

CICS MENU transaction OPERATOR INSTRUCTIONS screen

Create a screen capture

In the HATS Project View, right-mouse click your project and select **Open Host Terminal -> main** as shown in Figure C-62 on page 146.

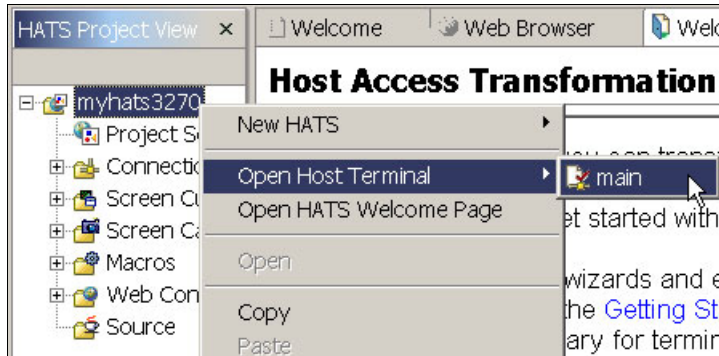


Figure C-62 Open Host Terminal

In this exercise you will create a screen customization for the CICS MENU transaction OPERATOR INSTRUCTIONS screen. First you must use the terminal window to navigate to that screen.

On the Welcome screen, if you have already created a signon macro, then on the toolbar click the drop-down next to the Play Macro icon and select the signon macro. Otherwise, you can manually sign on as follows.

1. On the Welcome screen, type: cicsa and press Enter.
2. On the Signon to CICS screen, type: whidemo for userid type. For Password, type: guest1 and then press Enter.
3. On the Sign-on Complete screen, type: menu and press Enter.
4. You are now at the Operator Instructions screen. To create a screen capture, click the **Create Screen Capture** icon.
5. In the Capture a Screen panel, type in a name for this screen capture: menuOperatorInstructions then click **Finish**; see Figure C-63.

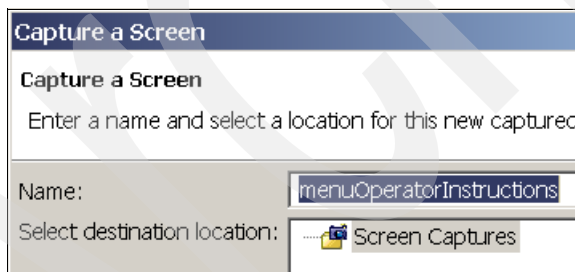


Figure C-63 Name screen capture

At this point you could continue to capture more screens and after capturing all the screens you want, then start customizing them. However, these exercises are written so that the work for each screen can be in a self-contained exercise.

After creating your screen capture, click the **Disconnect** button on the toolbar to disconnect the terminal session and close the terminal window.

Create Screen Customization

You have now created a screen capture for the Operator Instructions screen. Next you will create a screen customization for this screen. A screen customization consists of how to recognize the screen and then what actions to perform once the screen is recognized.

1. Notice in the Screen Captures folder you now have a screen capture for your menuOperatorInstructions screen. Right-mouse click the **menuOperatorInstructions** screen capture and select **New HATS -> Screen Customization** as shown in Figure C-64.

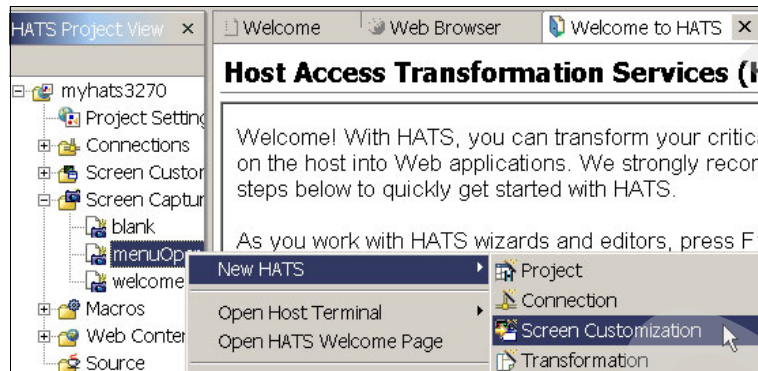


Figure C-64 Create screen customization

2. Accept the supplied screen customization name of menuOperatorInstructions and click **Next**.

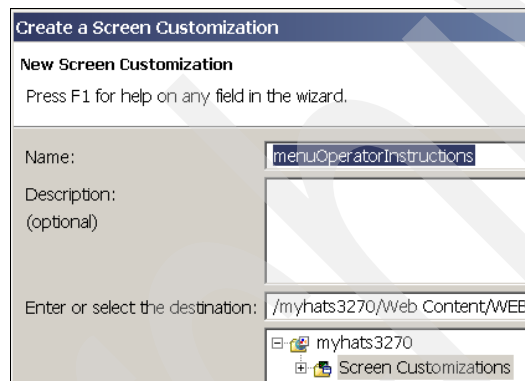


Figure C-65 Name screen customization

3. The first task in creating a screen customization is to tell HATS how to recognize the screen. On the Select Screen Recognition Criteria panel, with your cursor, draw a box around the Operator Instructions text at the top of the screen.

If you want to be sure to uniquely recognize this screen as the operator instructions for the menu transaction, you should also include at least the next line of text as well. Then check **Within a rectangular region** and click **Next**.
4. After you tell HATS how to recognize the screen, then you tell HATS what to do once the screen is recognized. On the Select Actions panel notice that you can Apply a transformation, which means to display the screen to the end user in a transformed way. Also, you can tell HATS to play a macro or perform Advanced Functions when this screen is recognized.

In our case, for this screen, just check **Apply a transformation** and click **Finish** as shown in Figure C-66 on page 148.

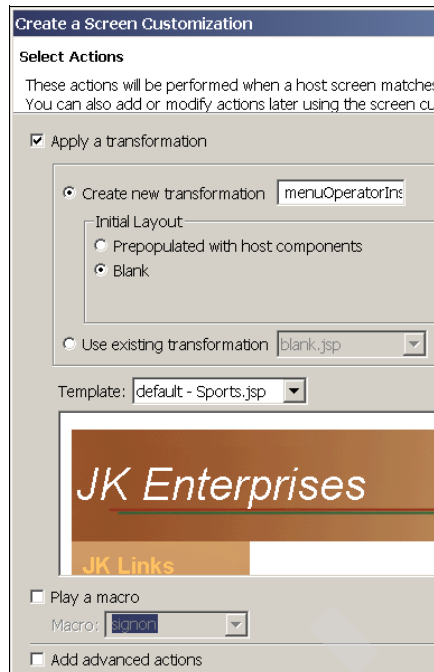


Figure C-66 Configure transformation options

Create Screen Transformation

You are now in the Transformation wizard. This first panel, Select Screen Region, is where you can select the first component from the host screen to display in a transformed way to the end user. For now, click **Cancel**. You will add your host components later.

You are back to the workbench view. Notice in the HATS Project View that there are now entries for the menuOperatorInstructions screen in both the Screen Customization and the Web Content/Transformations folders. Also notice the transformation for the menuOperatorInstructions screen, menuOperatorInstructions.jsp, is displayed in the editor view.

1. Click the **Design and Source** tabs to see the different views, as shown in Figure C-67. Before creating a transformed look, you may want to maximize the editor view. Do this by double-clicking the tab for **menuOperatorInstructions.jsp**.



Figure C-67 Maximize editor view

2. Now you will create a transformed look for the menuOperatorInstructions screen. Start by putting some text on the screen for instructions to the end user. Place your cursor just before the representation of the HATS Form tag. This is most easily done by selecting the **Form** tag with your mouse and then pressing the **Back** cursor key once.
3. After you have positioned the cursor, press the Enter key one or two times to insert some blank lines then type Customer Accounts as shown in Figure C-68 on page 149.

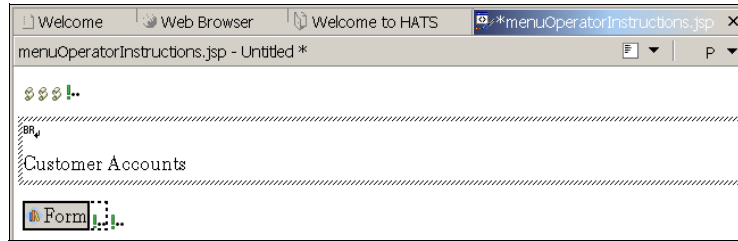


Figure C-68 Insert blank lines

- One method for changing the font for your text is to select the text and select **Format -> Font** from the toolbar. However, a more efficient method is to use stylesheets. Since you have previously imported and linked to your personal stylesheet, you can set the style of your text to match one of your style classes. This is a better method because if you want to change the attributes of your text throughout the project, you can do so in one place by changing the class in your stylesheet.

Right-mouse click your text and select **Style -> Edit Style Rule** as shown in Figure C-69.

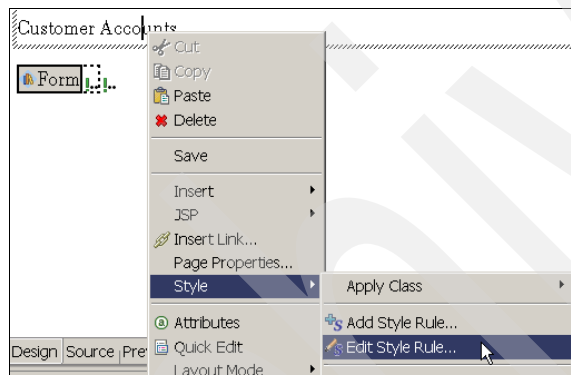


Figure C-69 Edit style rule

- For Class name, type: `myBigText`. This is a class name defined in the personal stylesheet you imported earlier. Look at how it is defined. Click **OK**.

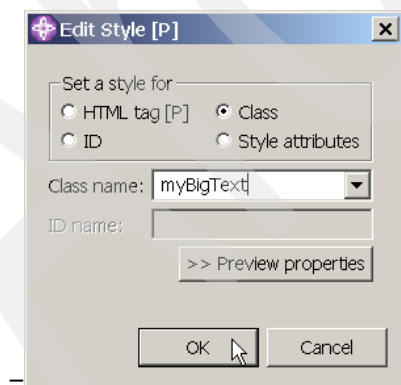


Figure C-70 Select style class name

- On the Edit Style panel, click **OK**.
- Now add another line of text that will have a different style class. With the cursor at the end of the text you just added, from the toolbar, select **Insert -> Paragraph -> Normal**.
- In the new paragraph, type: Make a selection, enter a number, and click Enter as shown in Figure C-71.

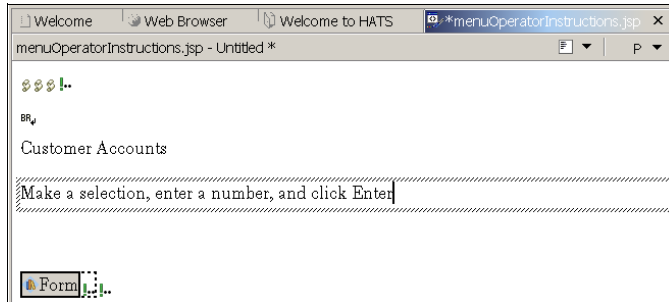


Figure C-71 Edit menuOperatorInstructions.jsp

As you did previously, change the style class for this new text, but this time set the class to myText.

9. Now add some HATS components from the host screen to a table. First insert a table on the JSP. Select the **Form** tag with your mouse and press the **Forward** cursor key twice. Then from the toolbar, select **Insert -> Table** as shown in Figure C-72.

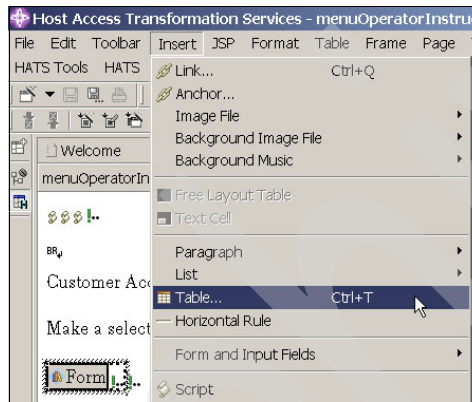


Figure C-72 Insert table

10. On the Insert table panel, select **Rows: 2 Columns: 2** and click **OK**.

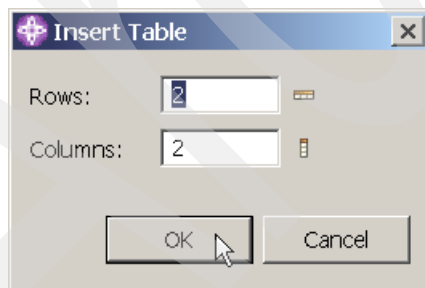


Figure C-73 Define table size

11. Now put some text into row 1. Position your cursor in row 1 column 1 and type: Selection. In row 1 column 2, type: Number. After entering the text, change the style class for the text in each column to myText, as you did for text you added previously.
12. Now you are ready to add HATS components to the second row of the table. Position the cursor in row 2 column 1, right mouse and select **HATS Tools -> Insert Host Component** as shown in Figure C-74 on page 151.

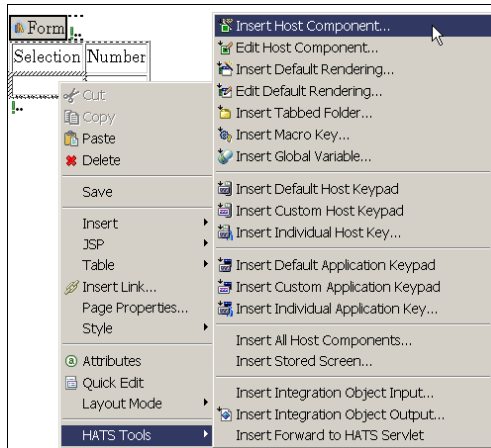


Figure C-74 Insert Host Component

13. On the Select Screen Region panel, if not already checked, check the box **Highlight input fields**. After you do this, you will notice that there are two input fields on this screen. The first is where the end user enters the transaction and the second the account number. In this case, you will create a subset list of transactions from which the end users can select. To start, drag your cursor around the input field for the transaction and click **Next**.

14. Here you see the Rendering Options panel. This is where you tell HATS what host component to use for this area of the screen and the widget into which it will be transformed. Notice, when you click each component, that different widget options are displayed.

For this case, select the **Input field** component. Notice that, by default, the component includes the caption for the input field, in this case, ENTER TRANSACTION. In this example, you do not want to include the caption.

To change this, click the **Component Settings** button; see Figure C-75.

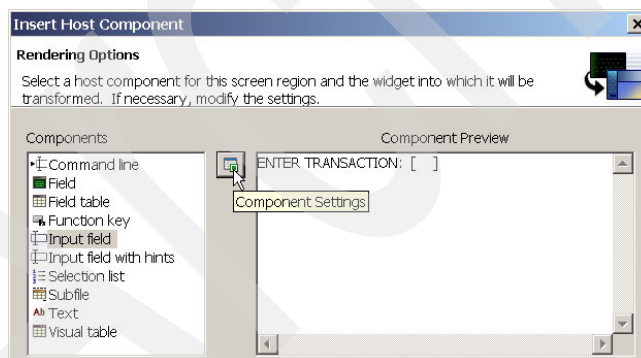


Figure C-75 Component settings

15. Uncheck Use project defaults, uncheck Extract field caption, then **OK** as shown in Figure C-76 on page 152.

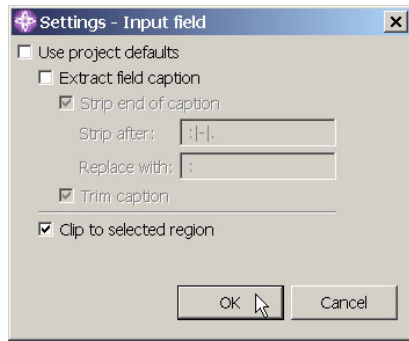


Figure C-76 Input field settings

16. Under Widgets select **Drop-down (data-entry)** and then click the **Widget settings** button as shown in Figure C-77.

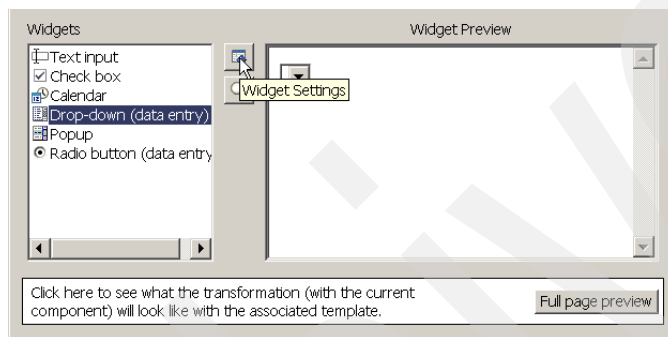


Figure C-77 Widget settings

17. Change the settings of the Drop-down widget as follows:

- a. Uncheck Use project defaults.
- b. Check **Fill from string**.
- c. In the List items box, type: Account number details=inqy;Browse accounts starting with=brws; where for example if "Browse accounts starting with" is selected from the list, then the string brws is passed to the host.

BRWS is the CICS transaction that displays all customer accounts starting with a particular number. Notice the class names for the drop-down list. These are defined in the stylesheets for the project.

- d. Click **OK**, as shown in Figure C-78 on page 153.

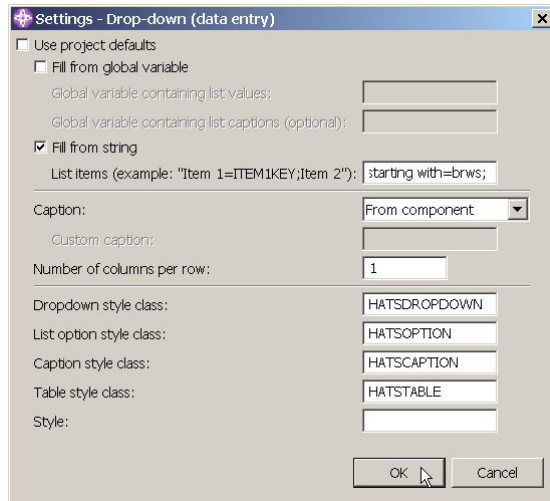


Figure C-78 Drop-down settings

18. Back on the Rendering Options panel under Widget Preview, you can now see what your widget will look like. Click **Finish**.
19. Now add another host component to the transformation to represent the account number input field. Position your cursor in row 2 column 2 of the table, right-mouse click and select **HATS Tools -> Insert Host Component**; see Figure C-79.

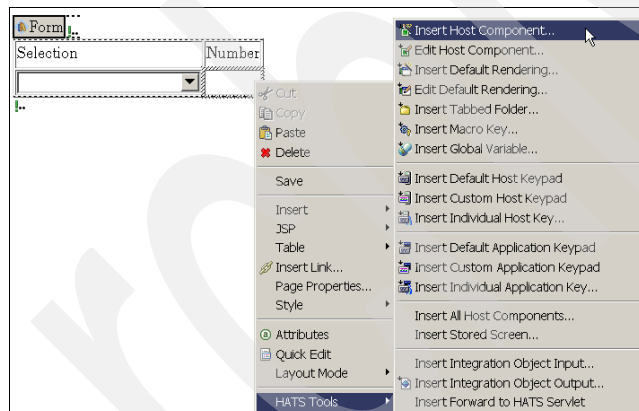


Figure C-79 Insert Host Component

20. On the Select Screen Region panel, with your mouse drag the cursor around the input field for the Number and click **Next**.
21. On the Rendering Options panel for the Component, select **Input field**. Then click the **Component Settings** button; see Figure C-80 on page 154.

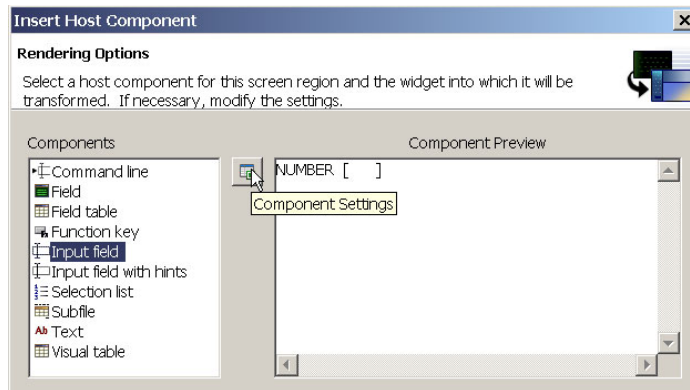


Figure C-80 Change Component settings

22. On the Settings panel, uncheck Use project defaults, uncheck Extract field caption, and then click **OK**.

23. Under Widgets, select the **Text input widget** and click **Finish**; see Figure C-81.

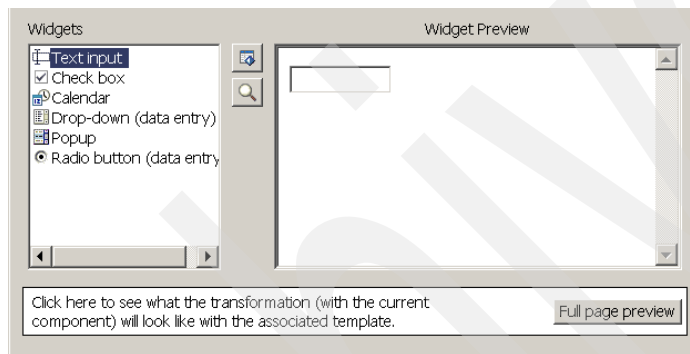


Figure C-81 Select text input widget

24. Next, add a button that, when clicked, will pass the Enter key back to the host. Press the **Forward** cursor button until the cursor moves out of and below the table. With the cursor positioned below the table, press the Enter key to add a blank line, then from the toolbar select **HATS Tools -> Insert Individual Host key**; see Figure C-82.

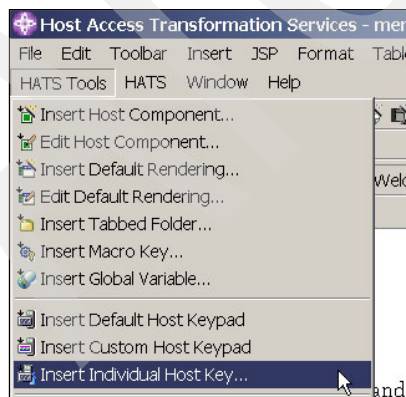


Figure C-82 Insert host key

Check **Enter** and click **OK** as shown in Figure C-83 on page 155.

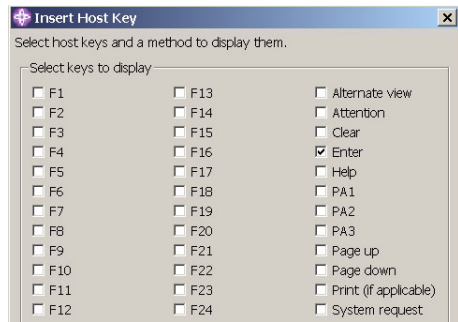


Figure C-83 Insert Host Key panel

If you would like to change the label on the Enter button, right-mouse click the **Enter** button and select **Attributes**. In the Attributes view, scroll down and change the label.

25. Now add another button that will send a Clear key to the host. Users will click this button to exit. Position the cursor following the Enter button and in the same manner add a button to send the Clear key.

After adding the Clear button, change its label as you did above for the Enter button. Label this button **Exit** as shown in Figure C-84.

26. Next, center your work. From the toolbar select **Edit -> Select All**. Then, also from the toolbar, select **Format -> Align -> Horizontal Center**.

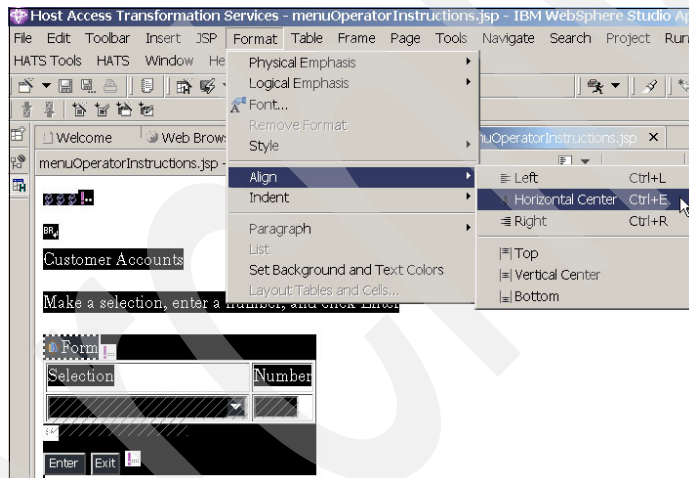


Figure C-84 Horizontally center screen elements

You have now finished the transformation for the CICS MENU transaction OPERATOR INSTRUCTIONS screen.

Close the editor by clicking the **X** and click **Yes** to save changes as shown in Figure C-85 on page 156.

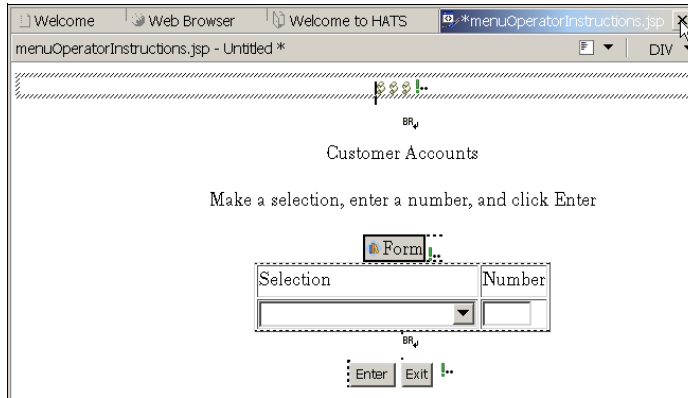


Figure C-85 Close JSP editor

Test Screen Transformation

Switch back to the Web Browser view, scroll down and click the **Refresh** button on the application keypad. If you get the Disconnected screen, click **Restart** and navigate back to the OPERATOR INSTRUCTIONS screen.

Notice your transformed OPERATOR INSTRUCTIONS screen. Select **Browse accounts starting with**, then type the number 10000 and click the **Enter** button to execute the CICS BRWS transaction; see Figure C-86.

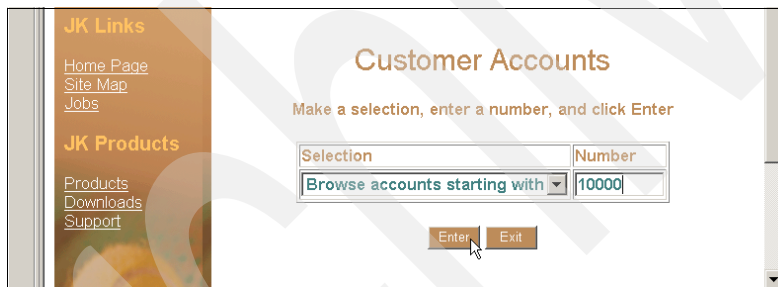


Figure C-86 New Operator Instructions screen

The CICS BRWS transaction is executed. Next you see the BRWS transaction FILE BROWSE screen, as shown in Figure C-87.

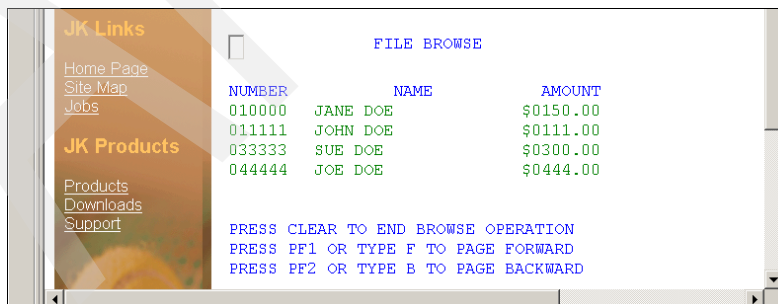


Figure C-87 Transaction browse screen

CICS MENU transaction FILE BROWSE screen

Create Screen Capture

In the HATS Project View, right-mouse click your project and select **Open Host Terminal -> main** as shown in Figure C-88.

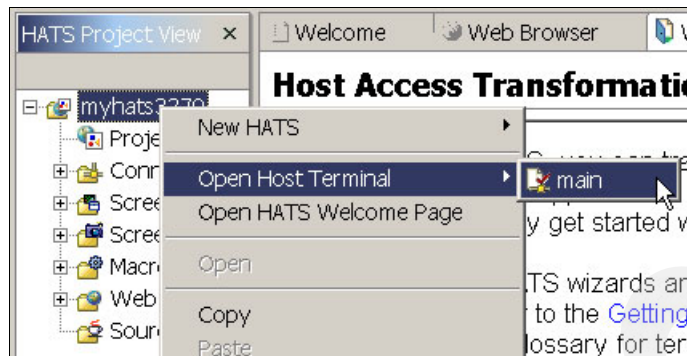


Figure C-88 Open Host Terminal

In this exercise you will create a screen customization for the MENU transaction's FILE BROWSE screen. First you must use the terminal window to navigate to that screen.

On the Welcome screen, if you have already created a signon macro, then on the toolbar click the drop-down next to the Play Macro icon and select the signon macro. Otherwise, you can manually sign on as follows:

1. On the Welcome screen, type: cicsa and press **Enter**.
2. On the Signon to CICS screen, type: whidemo for userid, type: guest1 for Password, and press **Enter**.
3. On the Sign-on Complete screen, type: menu and press **Enter**.
4. On the Operator Instructions screen for transaction, type: brws and press **Enter**.

You are now at the File Browse screen. To create a screen capture, click the **Create Screen Capture** icon.

5. In the Capture a Screen panel, type: menuFileBrowse to name this screen capture, then click **Finish**.

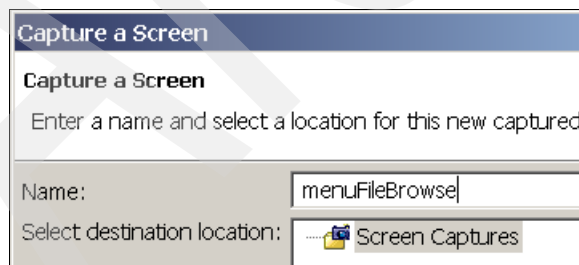


Figure C-89 Name screen capture

Record a macro

HATS macros can be played either by clicking a button, or automatically when a screen from the host is recognized. In this example you will record a macro that will loop through multiple screens, collect data from each screen, and display the results.

The results will be displayed using what is called a Macro Event Handler jsp. First you will make a copy of the default Macro Event Handler jsp to use for this exercise.

In the HATS Project View, navigate to the default jsp in the Web Content\Macro Event Handlers folder. Right-click the default jsp and select **Copy**.

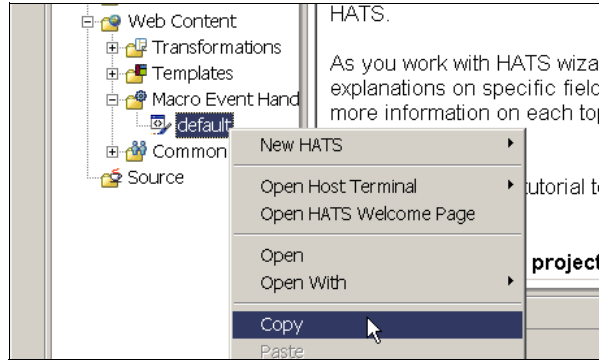


Figure C-90 Copy default.jsp

1. Right-click the **Macro Event Handlers** folder and click **Paste**.

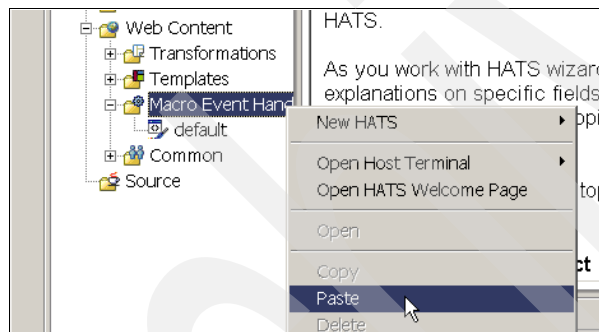


Figure C-91 Paste default.jsp

2. Type: menuFileBrowseMacro.jsp as the name of your jsp and click **OK**.

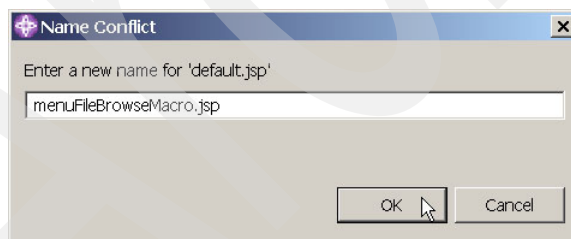


Figure C-92 Rename JSP

3. To start recording your macro, return to the open terminal window and click **Record Macro** on the toolbar.
4. If you get the message about recording a new macro or appending to the open macro, click **New Macro**. Type: menuFileBrowse to name your macro, then click **Finish** as shown in Figure C-93 on page 159.



Record Macro

Press F1 for help on any field in the wizard.

Name: menuFileBrowse

Description:
(optional)

Enter or select the destination: \myhats3270\Web Content\WEB

myhats3270
Macros

Figure C-93 Name macro

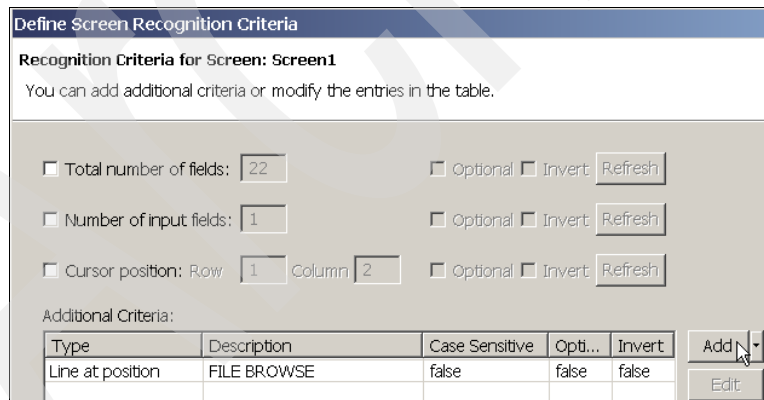
To define the starting screen of the macro, complete the following steps:

1. For Screen name, type: menuFileBrowse.
2. Drag your mouse around the text File Browse at the top of the screen.
3. Select **At a specified position**, then click **Finish**.
4. On the toolbar, click **Record a loop**.

While recording your loop, pay very close attention to the instructions in the panel below the host screen. You are already at the screen where the loop will start, so click **Next**.

Define the starting screen of the loop as follows:

1. For Screen Name, type: menuFileBrowsePressPF1.
2. Drag your mouse around the text File Browse at the top of the screen.
3. Select **At a specified position**, then click **Next**.
4. On the Recognition Criteria for Screen panel, click **Add** as shown in Figure C-94.



Define Screen Recognition Criteria

Recognition Criteria for Screen: Screen1

You can add additional criteria or modify the entries in the table.

☐ Total number of fields: 22 ☐ Optional ☐ Invert Refresh

☐ Number of input fields: 1 ☐ Optional ☐ Invert Refresh

☐ Cursor position: Row 1 Column 2 ☐ Optional ☐ Invert Refresh

Additional Criteria:

Type	Description	Case Sensitive	Opti...	Invert
Line at position	FILE BROWSE	false	false	false

Add Edit

Figure C-94 Macro - Add screen recognition criteria

On the String Criterion panel, complete the following:

1. Drag your mouse around the text Press PF1 in the center of the screen.
2. Select **At a specified position**, then click **OK**.
3. On the Recognition Criteria for Screen panel, click **Finish** as shown in Figure C-95 on page 160.

Define Screen Recognition Criteria

Recognition Criteria for Screen: Screen1

You can add additional criteria or modify the entries in the table.

☐ Total number of fields: ☐ Optional ☐ Invert

☐ Number of input fields: ☐ Optional ☐ Invert

☐ Cursor position: Row Column ☐ Optional ☐ Invert

Additional Criteria:

Type	Description	Case Sensitive	Opti...	Invert
Line at position	FILE BROWSE	false	false	false
Line at position	PRESS PF1	false	false	false

Figure C-95 Define Screen Recognition panel

In the terminal window, drag your mouse around the data you want to extract from the screen and click **Add Extract Action**.

On the Add Extract Action panel, complete the following:

1. For Name, type: menuFileBrowseAccts.
2. Select **Extract this region as a table**.
3. For Handle Macro Extract, select **Show handler**. For Handler, select **menuFileBrowseMacro.jsp**. This is the copy you made previously.
4. Click **Next** as shown in Figure C-96.

Add Extract Action

Add Extract Action

Select the extraction format and a method to handle the extract data.

Name:

Position

Start row: End row:

Start column: End column:

Extraction Format:

☐ Extract this region as one string

☐ Extract this region as a list of strings

☒ Extract this region as a table

Handle Macro Extract:

☒ Show handler

Handler:

☐ Save as global variable

Name:

Figure C-96 Add extract action

5. On the Table Extract Configuration panel, highlight Column1 and click **Divide** as shown in Figure C-97 on page 161.

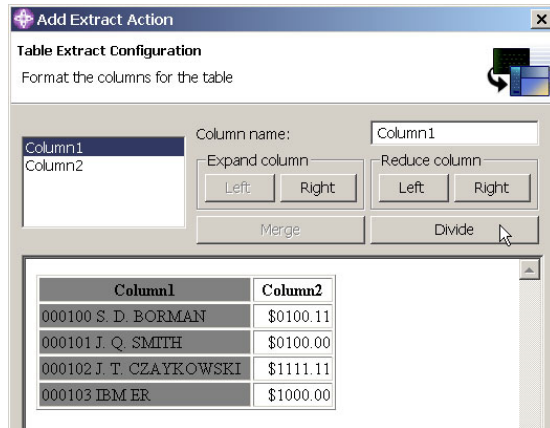


Figure C-97 Configure extraction table - divide

6. Rename the column names to match the names on the host screen. Highlight each column name and change the name in the Column name field. Rename Column1 to Number, Column2 to Name, and Column3 to Amount.
7. Highlight the Number column and click **Reduce column Right** until the data lines up correctly as shown in Figure C-98.

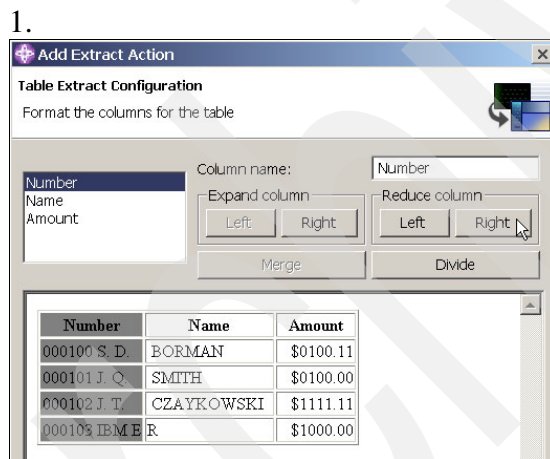


Figure C-98 Configure extraction table - reduce

8. Highlight the Name column and click **Expand column Right** to allow for the longest name in the list as shown in Figure C-99 on page 162. Remember, the longest name may not actually show up on this particular screen capture.

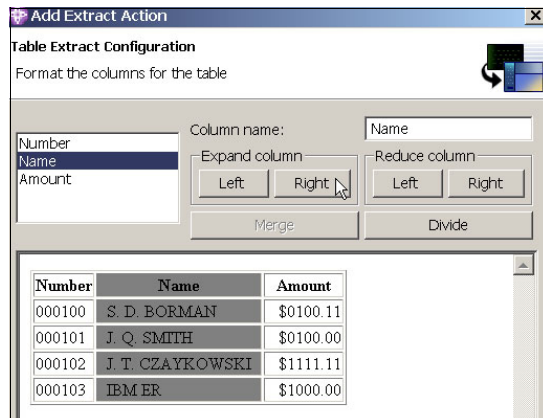


Figure C-99 Configure extraction table - expand

9. When finished, your panel should look similar to Figure C-100. Click **Finish**.

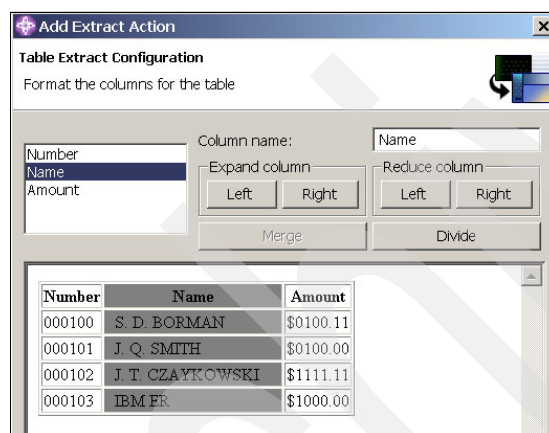


Figure C-100 Configure extraction table - finish

Back on the terminal window, you are instructed to perform the actions that will be executed during each cycle of the loop. In this case, press the **PF1** key or click the **PF1** button just once. Then click **Next** in the instructions panel.

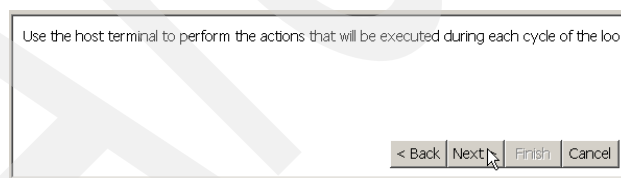


Figure C-101 Macro instruction panel

10. To end the loop, select **End when a unique screen is recognized** and click **Next**.

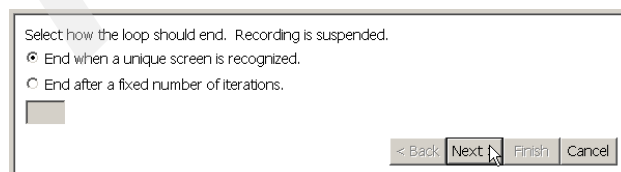


Figure C-102 Macro instruction panel - end

11. Check the box **Extract data from the last screen** as shown in Figure C-103.

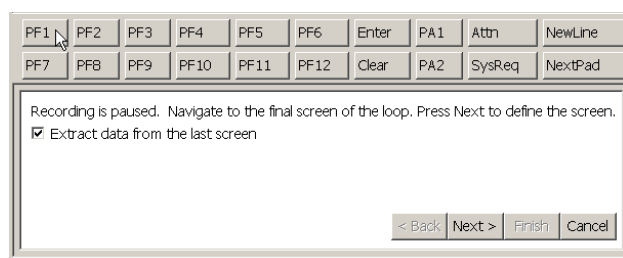


Figure C-103 Macro instruction panel - extract

Next you must navigate to the last screen. Press the **PF1** key or click the **PF1** button until you get to the last screen. In this example the last screen has the text HI-END OF FILE near the center of the screen. When you get to the last screen, if you did not earlier, check **Extract data from the last screen** and click **Next**; see Figure C-104.

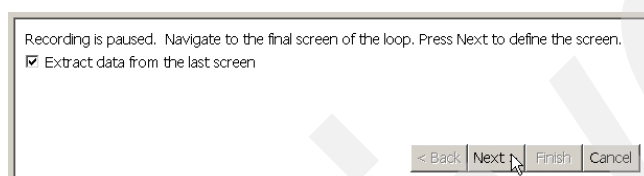


Figure C-104 Macro instruction panel

To define the final screen of the loop, complete the following steps:

1. For Screen Name, type: menuFileBrowseHIEnd.
2. Drag your mouse around the text File Browse at the top of the screen.
3. Select **At a specified position**, then click **Next**.
4. On the Recognition Criteria for Screen panel, click **Add**.
5. On String Criterion, drag your mouse around the text HI-END OF FILE, select **At a specified position**, then click OK.
6. On the Recognition Criteria for Screen panel, click **Finish**.
7. In the terminal window instruction panel, click **Finish** to complete the loop.

Now finish recording your macro by either pressing the **Clear** key or clicking the **Clear** button. To stop recording your macro, click **Stop Macro**.

For the Define the exit screen of the macro panel, complete the following:

1. For Screen Name, type: menuOperatorInstructions.
2. Drag your mouse around the block of text, including the top three lines of the screen.
3. Select **Within a rectangular region**, then click **Finish**.
4. Save your macro by clicking **Save Macro**.

In the Macro Navigator, notice the logic of the macro as shown in Figure C-105 on page 164.

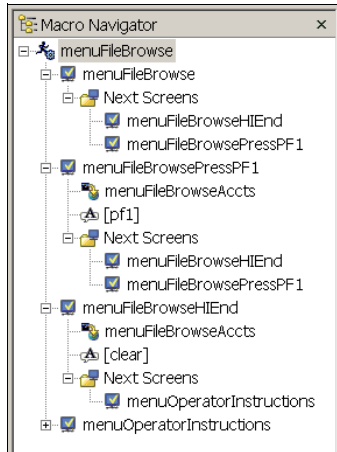


Figure C-105 Macro tree view

To test your macro, while on the Operator Instructions screen, type: brws in the Transaction field to navigate to the first FILE BROWSE results screen. Then, from the Play Macro drop-down, select the **menuFileBrowse** macro.

At this point you could continue to capture more screens and after capturing all the screens you want, then start customizing them. However, these exercises are written so that the work for each screen can be in a self-contained exercise.

After creating your screen capture, and recording your macro, click the **Disconnect** button on the toolbar to disconnect the terminal session and close the terminal window.

Notice the entry for the menuFileBrowse macro in the Macros folder.

Create Screen Customization

You have now created a screen capture for the FILE BROWSE screen and recorded a macro that will loop through all of the FILE BROWSE screens, collect data from each screen, and display the results. Next you will create a screen customization for this screen. A screen customization consists of how to recognize the screen, and then what actions to perform once the screen is recognized.

1. Notice in the Screen Captures folder that you now have a screen capture for the menuFileBrowse screen. Right-mouse click the menuFileBrowse screen capture and select New HATS -> Screen Customization as shown in Figure C-106.

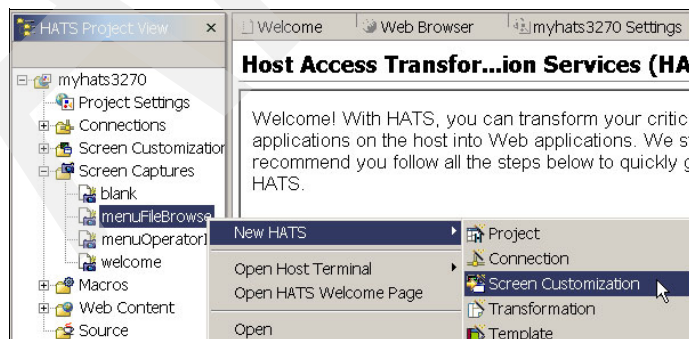


Figure C-106 Create screen customization

2. Accept the supplied screen customization name of menuFileBrowse and click **Next** as shown in Figure C-107 on page 165.

Create a Screen Customization

New Screen Customization

Press F1 for help on any field in the wizard.

Name:

Description: (optional)

Enter or select the destination:

File Explorer: myhats3270, Screen Customizations

Figure C-107 Name screen customization

3. The first task in creating a screen customization is to tell HATS how to recognize the screen. On the Select Screen Recognition Criteria panel, with your cursor draw a box around the File Browse text at the top of the screen. Check **At a specified position** and click **Next**.
4. After you tell HATS how to recognize the screen, then you tell HATS what to do once the screen is recognized. On the Select Actions panel, notice that you can Apply a transformation, which means to display the screen to the end user in a transformed way. Also, you can tell HATS to play a macro or perform Advanced Functions when this screen is recognized. In this case, you do not want to display this screen to the end user. You only want to collect data from it.

Uncheck Apply a transformation. However, you *do* want to play a macro when HATS recognizes this screen. So, check **Play a macro** and select the menuFileBrowse macro you recording earlier, then click **Finish**.

Create a Screen Customization

Select Actions

These actions will be performed when a host screen matches the criteria. You can also add or modify actions later using the screen editor.

☐ Apply a transformation

☒ Create new transformation:

Initial Layout:

☐ Prepopulated with host components

☒ Blank

☐ Use existing transformation:

Template:

☒ Play a macro

Macro:

☐ Add advanced actions

Preview: JK Enterprises, JK Links

Figure C-108 Configure transformation

Notice there is now a screen customization file for menuFileBrowse in the Screen Customization folder. But, there is no file for this screen in the Transformations folder. This is because you did not create a screen transformation for this screen. The data that is collected by the menuFileBrowse macro will be displayed by menuFileBrowseMacro.jsp that your

created previously in the Web Content\Macro Event Handlers folder and designated in your macro extract.

Test Screen Customization

Switch back to the Web Browser view, and if you are not already there, navigate to the transformation for the Operator Instruction's screen, select **Browse accounts starting with** and click the **Enter** button as shown in Figure C-109.

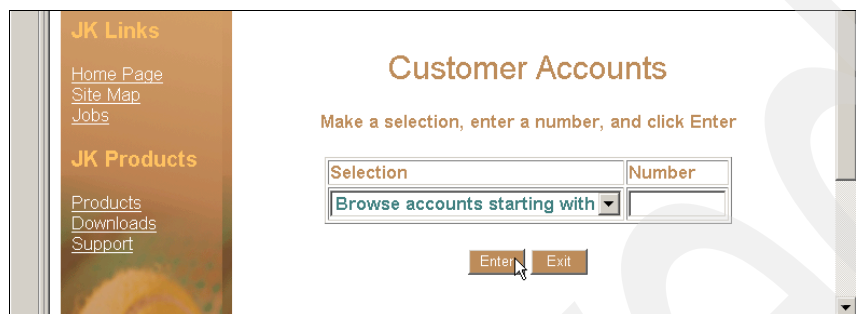


Figure C-109 Operator instructions screen

This may seem to take longer than your other screens took. The reason is that HATS is now navigating through 14 to 15 screens from the host system and collecting the data from them into your macro extract.

In the future, if during your testing you put an account number, for example, 10000, in the Number field, then the transaction will start with all account numbers equal to or greater than 10000. This will cut down on the number of host screens to navigate during your testing.

In the next exercise you will see how to modify the menuFileBrowseMacro.jsp.

Modify Macro Event Handler JSP

If you want to change the look and feel of the results shown in Figure C-110, you can edit the macro event handler jsp, which in this example is menuFileBrowseMacro.jsp in the Web Content\Macro Event Handlers folder. Start by double-clicking the jsp, as shown in Figure C-110.

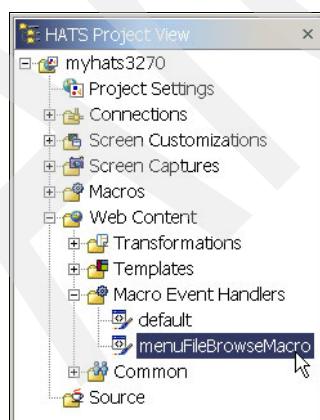


Figure C-110 menuFileBrowseMacro.jsp

To add your own text, complete the following steps as shown in Figure C-111:

1. Click the representation of the HATS Form tag and press **Enter** to create a new line.

- From the toolbar select **Insert -> Paragraph -> Normal**, then type your text, for example **Browse accounts starting with**

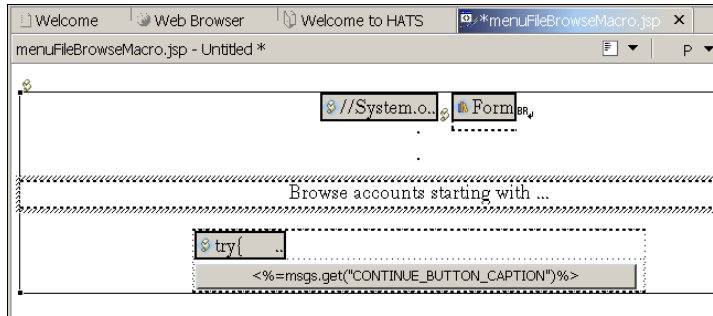


Figure C-111 Design view - menuFileBrowseMacro.jsp

- As you have done previously, change the style class for your text. Right-mouse click the text, select **Style -> Edit Style Rule**. Then set the class name to myText and click **OK**. On the add style panel, click **OK**.
- To change the style of the Continue button to match your other buttons, right-mouse click the **Continue** button, select **Style -> Edit Style Rule**. On the Edit Style panel, select **Class**, enter Class name of HostPFKey, and click **OK**. On the Add Style panel, click **OK**.
- You can skip this step if you like. However, to display the results in their own scrollable window, in the source for menuFileBrowseMacro.jsp, you *add or modify* the lines shown here marked with an asterisk (*).

Note: Do not actually put * in the source.

```

if (meEv != null){
    String name=meEv.getExtractName();
    String[] rows = meEv.getData();
    if ( !isTableFormat ) {

*        // reh added <div style=\"height: 200px; overflow: auto;\"> to following line
for scroll box
*        out.println("<tr><td><div style=\"height: 200px; overflow: auto;\"><table
class='HATSTABLE'>");
        //out.println("<tr><td>" + name + "</td></tr>");
        for (int i = 0; i < rows.length; i++) {
            //System.out.println("jsp extract line "+i+": "+rows[i]);
            out.println("<tr class='"+
                (i%2==0 ? "HATSTABLEEVENROW" : "HATSTABLEODDROW")
                +"'><td class='HATSTABLECELL'><tt>"
+Widget.htmlEscape(rows[i])+"</tt></td></tr>");
        }
*        // reh added </div> to following line for scroll box
*        out.println("</table></div></td></tr>");
    } else {
        int col,row;
        ColumnExtractInfo colInfo;

        if (displayColumnHeadings) {
*            // reh added <div style=\"height: 200px; overflow: auto;\"> to following
line for scroll box
*            out.println(" <tr><td><div style=\"height: 200px; overflow:
auto;\"><table class='HATSTABLE'>");
            out.println(" <tr>");
            for ( col = 0; col < columns.size(); col++ ) {
                colInfo = (ColumnExtractInfo)columns.elementAt(col);

```

```

        out.println("        <th class='.HATSTABLEHEADER'>" +
Widget.htmlEscape(colInfo.name) + "</th>");
    }
    out.println("        </tr>");
}
for ( row = 0; row < rows.length; row++ ) {
// reh added following 7 lines for trim
String[] wordsToOmit = {};
boolean skip = false;
for (int t=0; t<wordsToOmit.length; t++)
if (rows[row].indexOf(wordsToOmit[t]) >= 0) {
skip = true;
}
    if (! rows[row].trim().equals("") && a != meEvHt.size() && ! skip) {
// reh end of added lines for trim

        out.println("        <tr class='"+(totalTableRows%2==0 ?
"HATSTABLEEVENROW" : "HATSTABLEODDROW")+"'>");
        totalTableRows++;
        for ( col = 0; col < columns.size(); col++ ) {
            colInfo = (ColumnExtractInfo) columns.elementAt(col);
            out.println("            <td class='HATSTABLECELL'>" +
Widget.htmlEscape(
Util.getDBCSShrunkString(Util.getDBCSDoubledString(rows[row]).substring(colInfo.x,
colInfo.x + colInfo.dx)) ) + "</td>");
        }
        out.println("        </tr>");
// reh added following line for trim
    }
}
if (endTable)
* // reh added </div> to following line for scroll box
* out.println(" </table></div></td></tr>");
}
}

```

6. You can skip this step if you like. However, to trim blank lines from the end of the results, in the source for menuFileBrowseMacro.jsp, you add or modify the lines here marked with an asterisk (*).

Note: Do not actually put * in the source.

```

if (meEv != null){
    String name=meEv.getExtractName();
    String[] rows = meEv.getData();
    if ( !isTableFormat ) {

        // reh added <div style=\"height: 200px; overflow: auto;\> to following line
for scroll box
        out.println("<tr><td><div style=\"height: 200px; overflow: auto;\><table
class='HATSTABLE'>");
        //out.println("<tr><td>" + name + "</td></tr>");
        for (int i = 0; i < rows.length; i++) {
            //System.out.println("jsp extract line "+i+":."+rows[i]);
            out.println("<tr class='"+
                (i%2==0 ? "HATSTABLEEVENROW" : "HATSTABLEODDROW")
                +"'><td class='HATSTABLECELL'><tt>"
+Widget.htmlEscape(rows[i])+"</tt></td></tr>");
        }
        // reh added </div> to following line for scroll box
        out.println("</table></div></td></tr>");
    }
}

```

```

    } else {
        int col,row;
        ColumnExtractInfo colInfo;

        if (displayColumnHeadings) {
            // reh added <div style=\"height: 200px; overflow: auto;\"> to following
line for scroll box
            out.println(" <tr><td><div style=\"height: 200px; overflow:
auto;\"><table class='HATSTABLE'>");
            out.println(" <tr>");
            for ( col = 0; col < columns.size(); col++ ) {
                colInfo = (ColumnExtractInfo)columns.elementAt(col);
                out.println(" <th class='.HATSTABLEHEADER'>" +
Widget.htmlEscape(colInfo.name) + "</th>");
            }
            out.println(" </tr>");
        }
        for ( row = 0; row < rows.length; row++ ) {
            // reh added following 7 lines for trim
            * String[] wordsToOmit = {};
            * boolean skip = false;
            * for (int t=0; t<wordsToOmit.length; t++)
            * if (rows[row].indexOf(wordsToOmit[t]) >= 0) {
            * skip = true;
            * }
            * if (! rows[row].trim().equals("") && a != meEvHt.size() && ! skip) {
            * // reh end of added lines for trim
            out.println(" <tr class='"+(totalTableRows%2==0 ?
"HATSTABLEEVENROW" : "HATSTABLEODDROW")+"'>");
            totalTableRows++;
            for ( col = 0; col < columns.size(); col++ ) {
                colInfo = (ColumnExtractInfo) columns.elementAt(col);
                out.println(" <td class='HATSTABLECELL'>" +
Widget.htmlEscape(
Util.getDBCSShrunkString(Util.getDBCSDoubledString(rows[row]).substring(colInfo.x,
colInfo.x + colInfo.dx)) ) + "</td>");
            }
            out.println(" </tr>");
            * // reh added following line for trim
            * }
        }
        if (endTable)
            // reh added </div> to following line for scroll box
            out.println(" </table></div></td></tr>");
    }
}

```

7. Close the editor for your macro event handler jsp and click **Yes** to save changes.

Extra credit

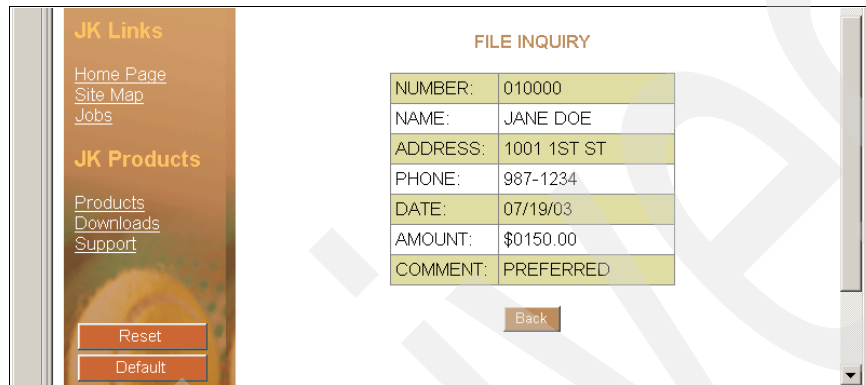
You have now finished the complete scenario for end users to list all of the accounts in the customer account file. This scenario used the CICS MENU transaction, followed by the CICS BRWS transaction. In the following sections describes other exercises you can try.

CICS MENU transaction FILE INQUIRY screen

On the MENU Operator Instructions screen (transformation with the Customer Accounts text on it), for Selection: select **Account number details**; for Number, type: 10000, then click the **Enter** button.

Notice the default transformation of the data, as shown in Figure C-112 on page 170.

Experiment with other ways to transform this data using different HATS components and widgets. For example in the following transformation, the text FILE INQUIRY on the host screen is rendered using the Text component and the Label widget, while the data is rendered using the Visual table component and the Table widget.



The screenshot shows a web application interface. On the left is a sidebar with 'JK Links' (Home Page, Site Map, Jobs) and 'JK Products' (Products, Downloads, Support). At the bottom of the sidebar are 'Reset' and 'Default' buttons. The main content area is titled 'FILE INQUIRY' and contains a table with the following data:

NUMBER:	010000
NAME:	JANE DOE
ADDRESS:	1001 1ST ST
PHONE:	987-1234
DATE:	07/19/03
AMOUNT:	\$0150.00
COMMENT:	PREFERRED

Below the table is a 'Back' button.

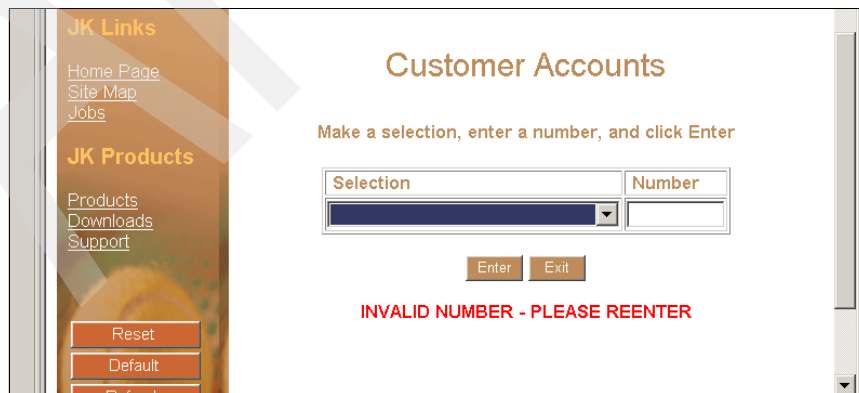
Figure C-112 FILE INQUIRY screen

CICS MENU transaction FILE INQUIRY error screen

On the MENU Operator Instructions screen (transformation with the Customer Accounts text on it), for Selection: select **Account Number**; for Number: type an invalid number (for example: 12121), then click the **Enter** button.

Notice that nothing seems to happen, as shown in Figure C-113. Click the **Default** button to see which screen is being displayed. Notice the error message that is displayed.

Create a screen customization that will recognize this screen and display the error message to the end user. In the example shown in Figure C-113, the error message text from the host screen is rendered using the Text component and the Label widget.



The screenshot shows a web application interface. On the left is a sidebar with 'JK Links' (Home Page, Site Map, Jobs) and 'JK Products' (Products, Downloads, Support). At the bottom of the sidebar are 'Reset', 'Default', and 'Default' buttons. The main content area is titled 'Customer Accounts' and contains the text 'Make a selection, enter a number, and click Enter'. Below this text are two input fields: 'Selection' (a dropdown menu) and 'Number' (a text box). Below the input fields are 'Enter' and 'Exit' buttons. At the bottom of the main content area, the error message 'INVALID NUMBER - PLEASE REENTER' is displayed in red text.

Figure C-113 Invalid account number screen

CICS NACT transaction accounts menu screen

On the Sign-On Complete (or Blank) screen (transformation displaying the text Select application ...), select **Credit Card Accounts**, then click the **Enter** button as shown in Figure C-114 on page 171.

Notice the default rendering of the NACT transaction's Accounts Menu screen.

Figure C-114 Account Menu screen

Now, for Request Type: type d (for display); for Account: type 10000, then press the **Enter** key.

Notice the default rendering of the Details of Account Number screen.

Create a screen customization for the Accounts Menu screen that allows an easier way to display account information, as shown in Figure C-115. In this example, a Screen Customization action is performed to automatically insert a “d” into the REQUEST TYPE field. Then a transformation renders the Account input field in a table cell using an Input field component and a Text input widget.

Figure C-115 Customized Account Menu screen

CICS NACT transaction details of account screen

On the Sign-On Complete (or Blank) screen (transformation with the text Select application ...), select **Credit Card Accounts**, then click the **Enter** button.

Notice the default rendering of the NACT transaction's Accounts Menu screen.

Now, for Request Type: type d (for display); for Account: type 10000, then press the **Enter** key.

Notice the default rendering of the Details of Account Number screen, as shown in Figure C-116 on page 172.

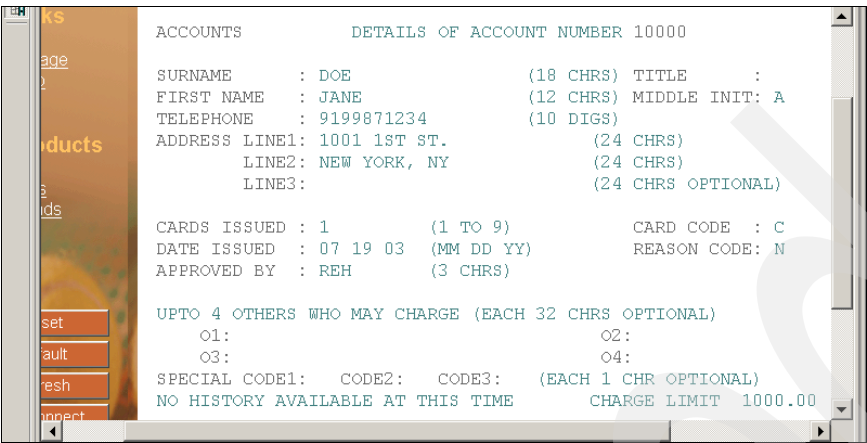


Figure C-116 Details of Account Number screen

Create a screen customization for the Details of Account Number screen that displays various fields on the screen in a table as shown below. In this example, the text from the host screen, DETAILS OF ACCOUNT NUMBER is rendered using a Text component and a Label widget. The transformation then displays, in a table, data that has been collected into global variables from different areas of the screen as shown in Figure C-117.

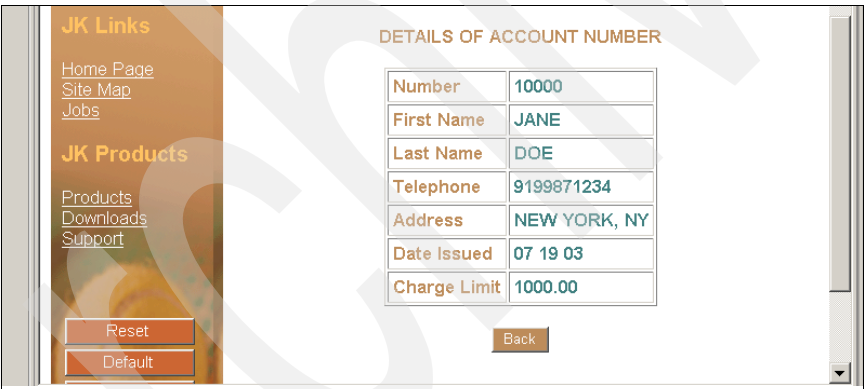


Figure C-117 Customized Details of Account Number screen

CICS NACT transaction accounts menu error screen

On the Sign-On Complete (or Blank) screen (transformation with the text Select application ...), select **Credit Card Accounts**, then click the **Enter** button.

Notice the default rendering of the NACT transaction's Accounts Menu screen.

Now, for Request Type: type d (for display); for Account: type 12121, then press the Enter key.

Notice the default rendering of the Details of Account Number Error screen as shown in Figure C-118 on page 173. The error message in this case reads NO RECORD OF THIS ACCOUNT NUMBER.



Figure C-118 Details of Account Error screen

Do the same thing as above but this time enter account number 1. Notice the error message in this case reads ACCOUNT NO. MUST BE NUMERIC AND FROM 10000 TO 79999, as shown in Figure C-119.

Create a single screen customization and transformation for the Details of Account Number Error screen that displays the various error messages as shown in Figure C-119. In this example, the error message text from the host screen is transformed using the Text component and the Label widget.

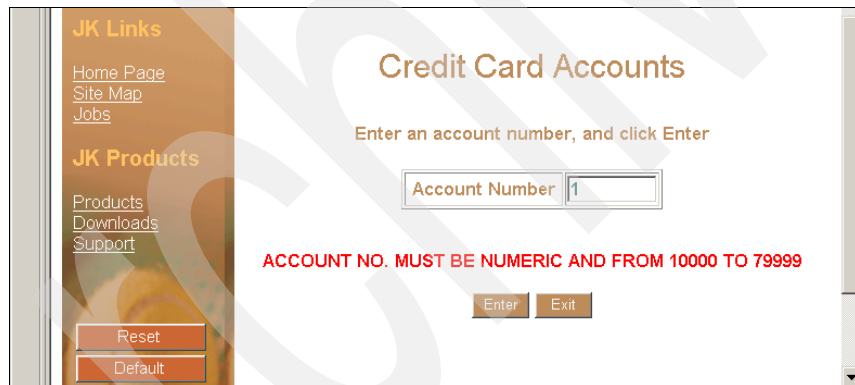


Figure C-119 Customized Details of Account Error screen

Archived

Lab 2: Exposing legacy applications as Web services

This lab exercise guides the reader through the process of enabling a 3270 application as a Web service.

Note: Updates can be found at:

<http://websphere.dfw.ibm.com/whidemo>

The lab assumes that WSSD v5.1.0 or later and Host Access Transformation Services (HATS) v5.0.4 are installed on a PC meeting the following requirements:

Table D-1 Installation requirements

Operating system	Software	Hardware
Linux	Red Hat 7.2 or 8.0, or SUSE 7.2 or 8.1. Web browser: Netscape Navigator 4.6 or 6.0, or Mozilla v0.7 or higher. TCP/IP installed and configured.	Intel® Pentium® II processor minimum; Pentium III 500 MHZ or higher recommended. XVGA (1024 x 768) display minimum. 512 MB RAM minimum; 768 MB RAM recommended. Disk space requirements: 1.3 GB minimum for installing.

Operating system	Software	Hardware
Windows 2000, Windows XP	<p>Windows 2000 Professional SP 2 or higher, or Windows XP Professional SP 1 or higher. Web browser: Microsoft Internet Explorer® 5.5 SP1 or higher, or Netscape Navigator 4.76 or higher.</p> <p>TCP/IP installed and configured.</p>	<p>Intel® Pentium® II processor minimum; Pentium III 500 MHZ or higher recommended.</p> <p>SVGA (800 x 600) display minimum (1024 x 768 recommended).</p> <p>512 MB RAM minimum; 768 MB RAM recommended.</p> <p>Disk space requirements: 1.3 GB minimum for installing.</p>

These instructions will show you how to use the HATS plug-in to WebSphere Studio to create a Web service that will access data from a zSeries 3270 application. First you will create a HATS project. Next you will create HATS macros that will be used to connect to and navigate through the host application. From one of the macros you will create a HATS Integration Object, and from the Integration Object you will create a Web service. You will test the Web service using the Web Services Explorer. Then you will create a sample client to use your Web service and test it, as well.

Naming conventions

When using HATS to create Web services, these naming conventions should be followed:

- ▶ Macro names can begin with either an upper case letter or a lower case letter.
- ▶ Integration Object names become class names and must begin with an upper case letter. Integration Object names are derived from macro names. So if the macro starts with a lower case letter, then HATS will convert it to upper case when creating the Integration Object name.
- ▶ Macro prompt and extract names become method names and must begin with a lower case letter.
- ▶ For any of the names mentioned here, a letter following an underscore or a number must be upper case.

Create a HATS project

Start WebSphere Studio and the HATS perspective, as follows:

1. Click **Start -> Programs -> IBM WebSphere Studio -> Application Developer 5.1.2.**
2. For workspace name, type: c:\myworkspaces\myhatslab and click **OK.**
3. Open the HATS perspective.
 - a. If the HATS perspective is not already open, click the **Open a Perspective** icon and select Host Access Transformation Services, as shown in Figure D-1 on page 177.

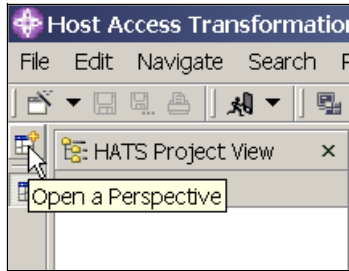


Figure D-1 Open WebSphere Studio perspective

4. Launch the Create HATS Project Wizard:

- a. On the Welcome to HATS page, click **launch the Create HATS Project Wizard** as shown in Figure D-2.

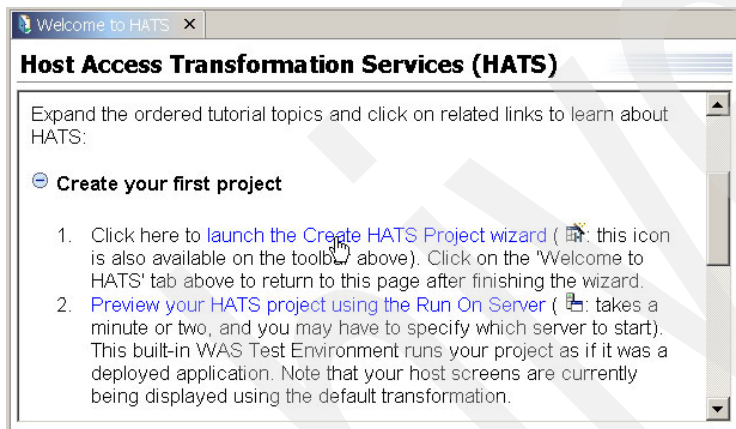


Figure D-2 HATS welcome panel

5. Name your project. On the Create a Project panel:

- a. For Name: enter MyAccounts.
- b. Uncheck Use default Enterprise Application project.
- c. For Enterprise Application project name: enter MyAccounts.ear.
- d. Click **Next** as shown in Figure D-3 on page 178.

Create a Project

Create a Project
Press F1 for help on any field in the wizard.

Name: MyAccounts

Description: (optional)

☒ Use default location
Location: C:\myworkspaces\myhatslab\MyAccounts

☐ Use default Enterprise Application project
Enterprise Application project name: MyAccounts.ear

☒ Add administration support

Figure D-3 Create a project panel

6. Set up your host connection. Your Web service will use this connection to communicate with the host application. On the Connection Settings panel:
 - a. For Host name: enter `zserveros.dfw.ibm.com`.
 - b. For Type: select 3270.
 - c. Click **Next** as shown in Figure D-4.

Create a Project

Connection Settings
Configure host connection settings for this new project.

You can specify advanced connection values later in the connection editor.

Host name: zserveros.dfw.ibm.com

Port: 23

Type: 3270

Code page: 037 United States

Screen size: 24 x 80

Figure D-4 Connection settings panel

7. Select your default template.
 - a. On the Select Default Template panel, from the Template pull-down select whichever template you want (for example, Swirl.jsp). The template will not be used by your Web service.
 - b. Click **Finish** as shown in Figure D-5 on page 179.

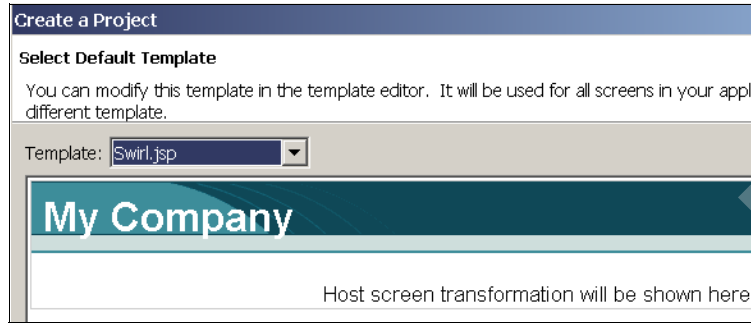


Figure D-5 Select default template panel

After finishing, the Project Settings Overview is displayed. Click the **X** on the editor's settings tab to close this view, as shown in Figure D-6.

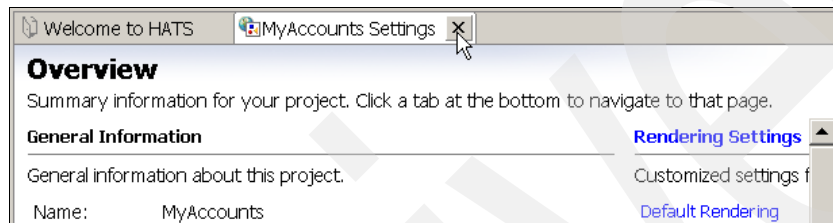


Figure D-6 Project overview panel

In the HATS project view, notice your project including all of its folders. You will be looking into these folders as you work through this lab.

Create macros

Next, you will create three macros. We will call them the Connect, Data, and Disconnect macros. The Connect macro will be used to sign on to the system and navigate to a point at which the Data macro will begin. The Data macro will be used to prompt for input, navigate through the application to extract data based on the requested input, and then navigate back to where it can prompt for the next input. Your Web service will ultimately be derived from the Data macro. Finally, the Disconnect macro will be used to sign off of the system.

Connect macro

First create the record connect macro, as follows.

Record connect macro

1. In the HATS project view, right-click your project's folder, select **Open Host Terminal** then select the **main host connection** as shown in Figure D-7 on page 180.

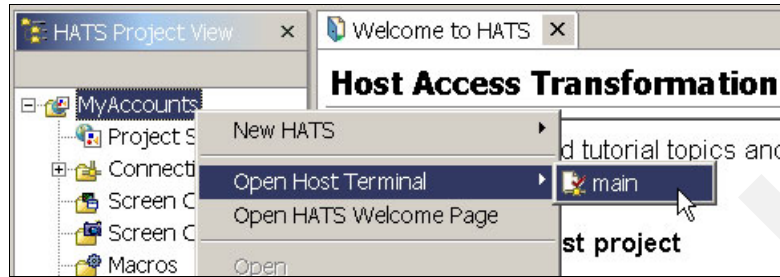


Figure D-7 Open Host Terminal

2. Here you see the Welcome screen from the host system. You will now record the Connect macro.

To start recording your Connect macro, click **Record Macro** on the toolbar.

3. On the Record Macro panel for Name, enter AcctConn. Click **Finish** as shown in Figure D-8.



Figure D-8 Name new macro

4. On the Define the starting screen of the macro panel, for Screen Name type: Welcome. Drag your mouse around the text **Welcome to IBM** in the middle of the screen. Select **At a specified position** and click Finish.
5. On the Welcome screen for Selection, type: cicsa, and press Enter (or click the **Enter** button).
6. Next you see the Signon to CICS screen. Click **Define Screen Recognition Criteria**.
7. On the Select Screen Recognition Criteria panel, for screen name type: SignonCICS. Drag your mouse around the screen title **Signon to CICS** at the top of the screen. Select **At a specified position** and click **Finish**.
8. On the Signon to CICS screen, for Userid type: whidemo. Tab to the Password field and type: guest1, then press Enter (or click the **Enter** button).
9. On the Sign-on is complete screen, click Define Screen Recognition Criteria.
10. On the Select Screen Recognition Criteria panel, for Screen Name type: SignonComp. With your mouse, draw a box around the text Sign-on is complete at the bottom of the screen. Select **At a specified position** and click **Finish**.

11. On the Signon Complete screen, type the CICS transaction menu and press Enter key (or click the **Enter** button).
12. On the Operator Instructions screen, click **Stop Macro**.
13. On the Define the exit screen of the macro panel, for Screen Name type: MenuInst. With your mouse, draw a box around the text Operator Instructions at the top of the screen. Select **At a specified position** and click **Finish**.
14. In the Macro Navigator notice the macro logic, then click **Save Macro**.

Test connect macro

Now you will test the macro you have just created. Navigate the terminal window back to the Welcome screen by pressing the Clear (Esc) key to exit.

1. On the blank screen that appears next, type: ces f logoff and press Enter. (Another way to get back to the SignOn screen is to disconnect and reconnect the host session from the icons on the host terminal toolbar.)
2. At the Welcome screen, from the Host Terminal toolbar, select the **Play Macro** drop-down and select the **AcctConn** macro.
3. Notice the macro playing and navigating to the Menu Operator Instructions. This is the screen where your Data macro will begin.

Data macro

Next we will develop a record data macro.

Record data macro

At this point you should have the Host Terminal window open at the Operator Instructions screen. If you are not at this screen, follow the preceding instructions to open the Host Terminal window and play your Connect macro to position the Host Terminal window at the Operator Instructions screen.

1. To start recording your Data macro, click **Record Macro** on the toolbar.
2. When asked if you want to record a new macro or append to the open macro, click **New Macro**.
3. On the Record Macro panel for name, type: AcctData. Remember that the naming conventions state that macro names can start with either an upper case letter or a lower case letter. However, Integration Object names must start with an upper case letter.

Later in the lab you will create an Integration Object from this Data macro. If you name your macro starting with a lower case letter, HATS will convert it to upper case when creating the Integration Object.

Click **Finish** as shown in Figure D-9 on page 182.



Figure D-9 Name new macro

4. On the Define the starting screen of the macro panel, for Screen Name type: MenuInst. With your mouse, draw a box around the text OPERATOR INSTRUCTIONS at the top of the screen. Select **At a specified position** and click **Finish**.

Your Web service will access data from the inqy transaction based on an account number. Therefore, create a prompt action that will allow your Web service clients to enter the account number on which to search.

1. On the Menu Instructions screen, for Transaction type: inqy. With the cursor positioned at the Number field, click **Add Prompt Action** from the toolbar.
2. On the Add Prompt Action panel, for Name type: acctSearchValue. (Remember the naming conventions; this is a macro prompt name that will become a method name, so it must start with a lower case letter.) Click **OK** as shown in Figure D-10.

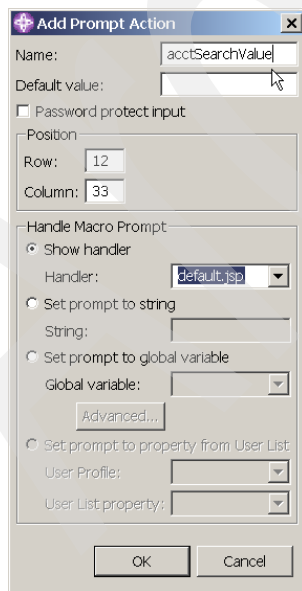


Figure D-10 Configure prompt action

3. On the Prompt panel, enter an account number value of 11111 and click **OK**; see Figure D-11 on page 183.

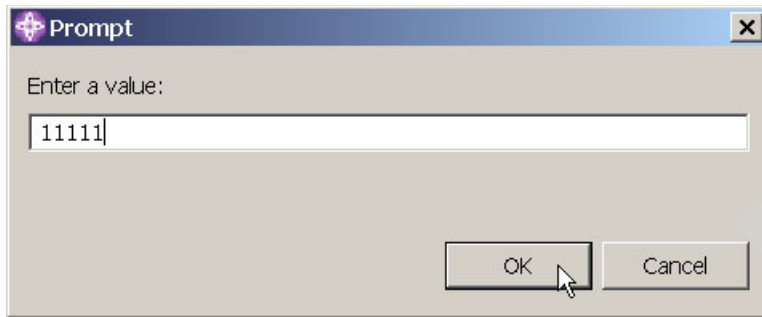


Figure D-11 Entry prompt value

4. On the Operator Instruction screen, press Enter (or click the **Enter** button).
5. On the File Inquiry screen, click **Define Screen Recognition Criteria**.
6. On the Select Screen Recognition Criteria panel, for Screen Name type: MenuInqy. With your mouse, draw a box around the text FILE INQUIRY at the top of the screen. Select **At a specified position** and click **Finish**.

Next you will create the outputs to be provided by your Web service.

1. On the File Inquiry screen, drag your mouse around the NUMBER field and click the icon **Add Extract Action**.
2. On the Add Extract Action panel, give this extract the name acctNumber. (Remember the naming conventions; this extract name will become a method name so it must start with a lower case letter.) Click **Finish** as shown in Figure D-12.

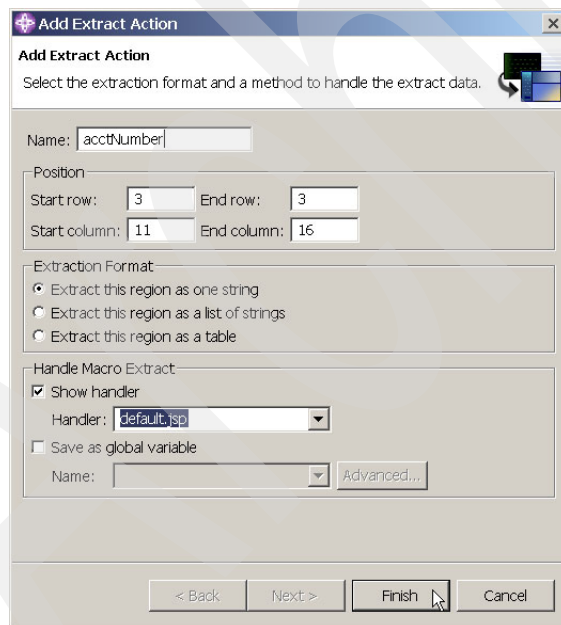


Figure D-12 Macro - configure extract action

3. Drag your mouse around the NAME field and click the icon **Add Extract Action**.
4. Give this extract the name acctName. (Remember, it must start with a lower case letter.) Change the value for the End column field to 30, to allow for the longest name. Click **Finish**; see Figure D-13 on page 184.

Add Extract Action

Select the extraction format and a method to handle the extract data.

Name:

Position

Start row: End row:

Start column: End column:

Extraction Format:

☒ Extract this region as one string

☐ Extract this region as a list of strings

☐ Extract this region as a table

Handle Macro Extract:

☒ Show handler

Handler:

☐ Save as global variable

Name:

Figure D-13 Macro - configure extract action

5. Drag your mouse around the ADDRESS field and click the icon **Add Extract Action**.
6. Give this extract the name acctAddress. (Remember, it must start with a lower case letter.) Change the value for the End column field to 30, to allow for the longest address. Click **Finish**; see Figure D-14.

Add Extract Action

Select the extraction format and a method to handle the extract data.

Name:

Position

Start row: End row:

Start column: End column:

Extraction Format:

☒ Extract this region as one string

☐ Extract this region as a list of strings

☐ Extract this region as a table

Handle Macro Extract:

☒ Show handler

Handler:

☐ Save as global variable

Name:

Figure D-14 Macro - configure extract action

7. Drag your mouse around the PHONE field and click the icon **Add Extract Action**.
8. Give this extract the name acctPhone. (Remember, it must start with a lower case letter.) Click **Finish**; see Figure D-15 on page 185.

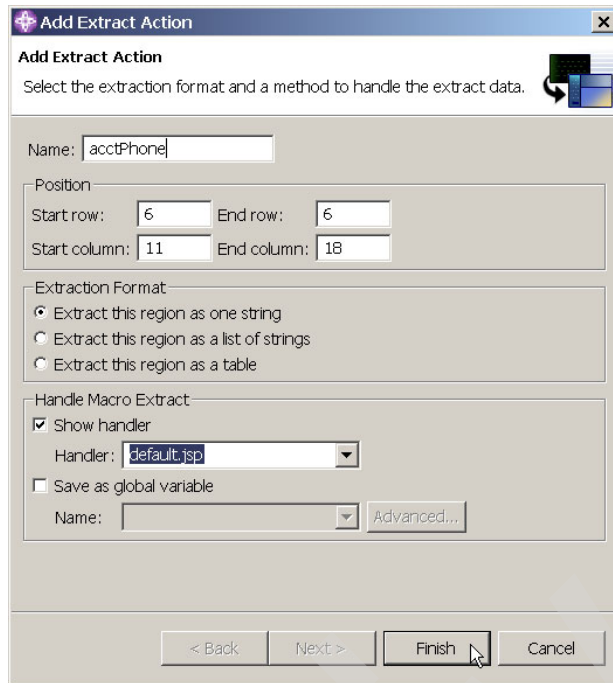


Figure D-15 Macro - configure extract action

Continue as above creating extracts for the DATE, AMOUNT, and COMMENT fields, naming them acctDate, acctAmount, and acctComment, respectively.

Now that you have extracted the results you want, you must navigate the Data macro back to the screen on which it begins so that the host session will be at the correct screen to run the Data macro again for another request. To navigate back to the screen on which the Data macro begins, click the **Enter** button at the bottom of the Host Terminal window.

You are now back at the Operator Instructions screen.

1. You have finished recording your Data macro, so click the **Stop Macro** button on the toolbar.
2. On the Define the starting screen of the macro panel, for Screen Name type: MenuInstEnd. Even though this is the same screen that the macro starts with, you must give it a different name. All screen names within a macro must be different.

With your mouse, draw a box around the text OPERATOR INSTRUCTIONS at the top of the screen. Select **At a specified position** and click **Finish**.

3. Notice the logic in the Macro Navigator, then click **Save Macro** on the toolbar.

Test data macro

Now you will test the Data macro you have just created. If the Host Terminal is not already at the Operator Instructions screen, then navigate to it. You can do this manually in the Host Terminal window, or by disconnecting and reconnecting the host session and using the Connect macro to play to the Operator Instructions screen.

4. After you are back at the Operator Instructions screen, from the Host Terminal toolbar select the **Play Macro** drop-down and select the **AcctData** macro.
5. In the Supply Prompt Values panel, enter a value of 11111 and click **OK**; see Figure D-16 on page 186.

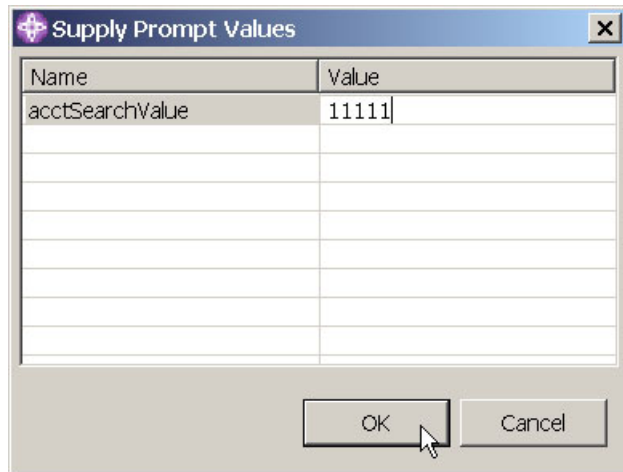


Figure D-16 Supply prompt values panel

Watch the macro play. In the Extract Results panel, select each of the Extract names from the drop-down as shown in Figure D-17.

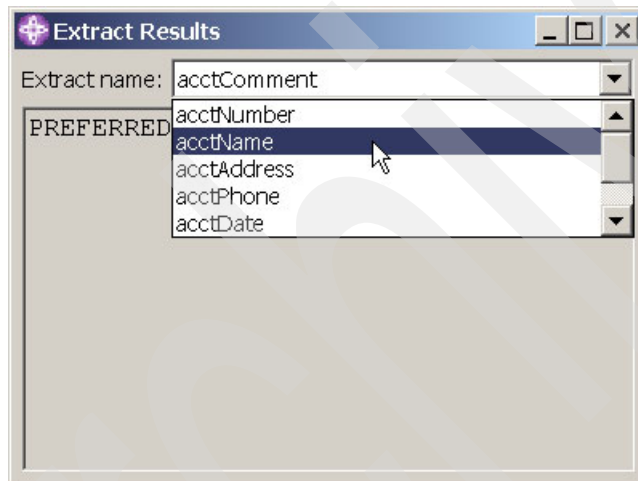


Figure D-17 Extract results view

Notice the results for each Extract name; see Figure D-18.

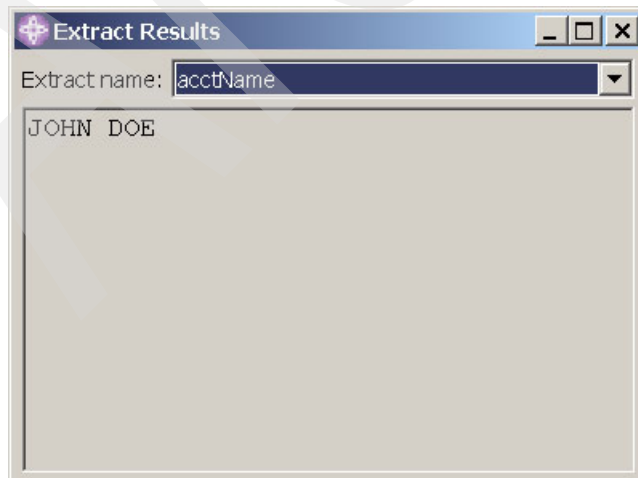


Figure D-18 Extract results view - account name

Disconnect macro

Next, you will develop the disconnect macro.

Record disconnect macro

At this point you should have the Host Terminal window open at the Operator Instructions. If you are not at this screen, follow the instructions given previously to open the Host Terminal window and play your Connect macro to position the Host Terminal window at the Operator Instructions screen.

1. To start recording your Disconnect macro, click **Record Macro** on the toolbar.
2. When asked if you want to record a new macro or append to the open macro, click **New Macro**.
3. On the Record Macro panel, for name type: AcctDisc, then click **Finish**; see Figure D-19.



Figure D-19 Name new macro

4. On the Define the starting screen of the macro panel, for Screen Name type MenuInst. With your mouse draw a box around the OPERATOR INSTRUCTIONS text at the top of the screen. Select **At a specified position** and click **Finish**.
5. On the Operator Instructions screen, press the Clear (Esc) key (or click the **Clear** button on the host key pad) to exit.
6. On the Blank screen, click **Define Screen Recognition Criteria**.
7. On the Select Screen Recognition Criteria panel, for Screen Name type: Blank. Drag your mouse around enough of the blank screen to uniquely identify it from any other screen. Select **Within a rectangular region** and click **Finish**.
8. On the Blank screen, type: cesf logoff, then press Enter (or click the **Enter** button).
9. On the Welcome screen, click **Stop Macro**.
10. On the Define the exit screen of the macro panel, for Screen Name type: Welcome. With your mouse, draw a box around the text WELCOME TO IBM in the middle of the screen. Select **At a specified position** and click **Finish**.
11. In the Macro Navigator, notice the logic of your Disconnect macro, then click **Save Macro**.

Test disconnect macro

Now you will test the macro you have just created.

1. Navigate the terminal window back to the Operator Instructions screen by playing your Connect macro.
2. After you are back to the Operator Instructions screen, from the Host Terminal toolbar select the **Play Macro** drop-down and select the **AcctDisc** macro.

Notice the macro playing and navigating back to the Welcome screen. Close the Host Terminal window. You have now created Connect, Data, and Disconnect macros.

In the HATS Project view, notice the macros in the Macros folder. If you ever need to go back to edit any of these macros, do so by double-clicking it as shown in Figure D-20.

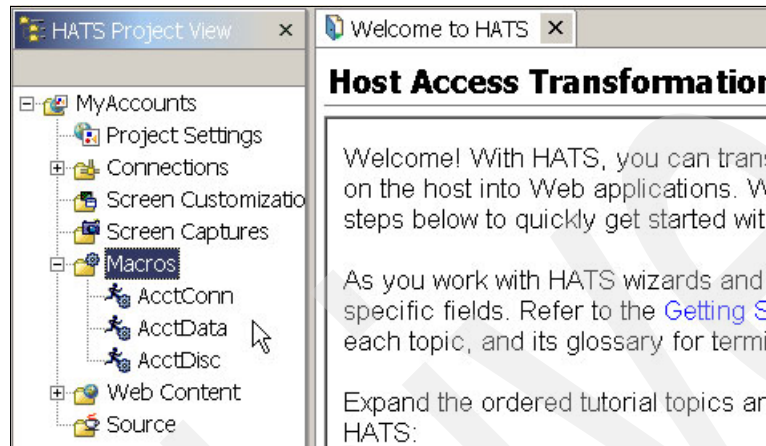


Figure D-20 HATS Project View

To summarize, the Connect macro will be used to sign onto the system and navigate to a point at which the Data macro will begin. The Data macro will be used to prompt for input, navigate through the application to extract data based on the requested input, and then navigate back to where it can prompt for the next input. Your Web service will ultimately be derived from the Data macro. Finally, the Disconnect macro will be used to sign off from the system.

Next you will set up connection parameters for your host connection.

Set connection parameters

When you deploy your Web service, it will use a connection (a 3270 session) to access your zSeries system. You set up the basic configuration parameters for your connection when you first built your project. Now you will define connection pooling parameters and link the macros you just built to the connection. Also you will enable the Display Terminal trace so that when you test your Web service, you will be able to see the host connection that your Web service is using.

Enable pooling

In order to improve the performance for your Web service, you will want to implement connection pooling for the connection to be used by the Web service. Connection pooling allows you to specify a number of connections (3270 sessions) that HATS will maintain in a pool that are already connected and ready to be used by your Web service. In other words, HATS will maintain a number of connections in the pool for which the Connect macro has already been run.

Therefore, each of the connections will be connected and immediately ready to run the Data macro. This avoids constant connecting and disconnecting from the host system in order to service multiple Web service requests.

The connections defined in your project are found in the Connections folder, so open this folder now. HATS can support multiple connections for the purpose of collecting and combining data from multiple back-end host sites. However, in this project you have defined just one connection, which by default is named “main”.

1. Double-click the main connection to set its connection parameters, as shown in Figure D-21.



Figure D-21 Main connection

2. Look through each of the tabs to see the different configuration settings. Then click the Pooling tab; see Figure D-22.

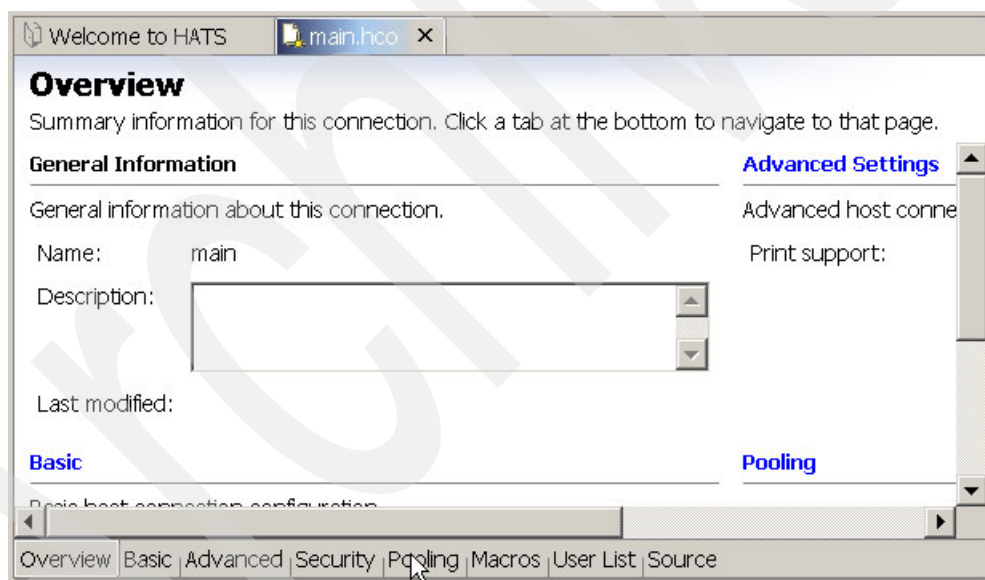


Figure D-22 Connection settings overview

3. On the Pooling tab, click **Enable Pooling**; see Figure D-23 on page 190.

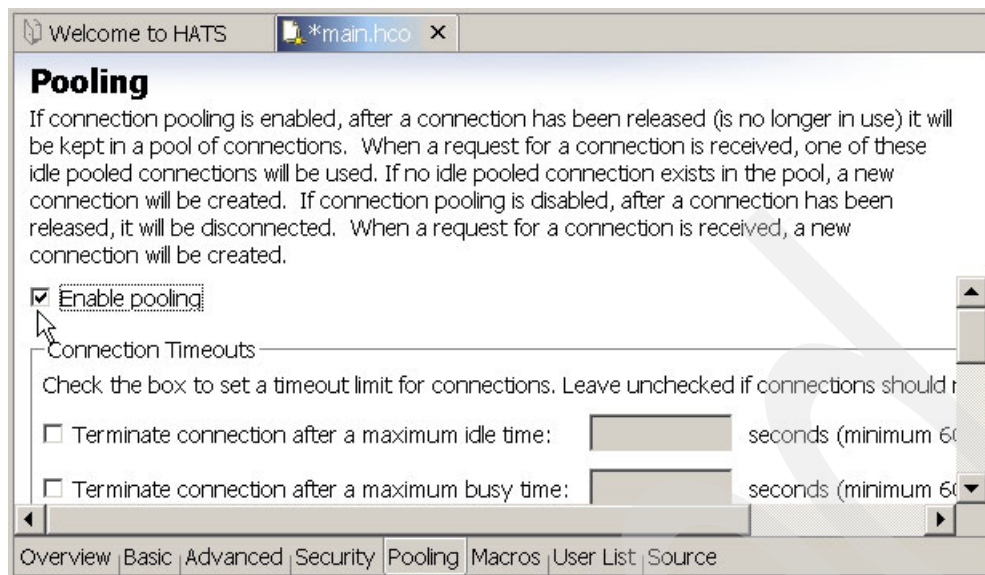


Figure D-23 Connection pooling configuration

Scroll down and set the Connection Limits. For this lab, set the limits to a Minimum of 1 and a Maximum of 2.

Then click the **Macros** tab; see Figure D-24.

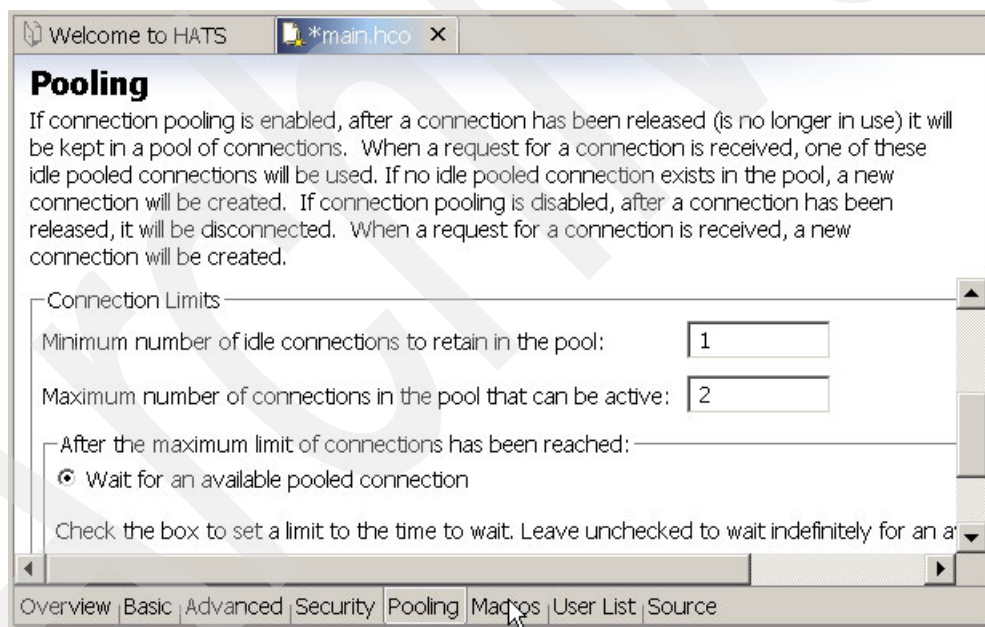


Figure D-24 Connection pooling configuration (continued)

Set up connect and disconnect macros

1. On the Macros tab for the Connect macro, use the drop-down menu and select your Connect macro AcctConn; see Figure D-25 on page 191.

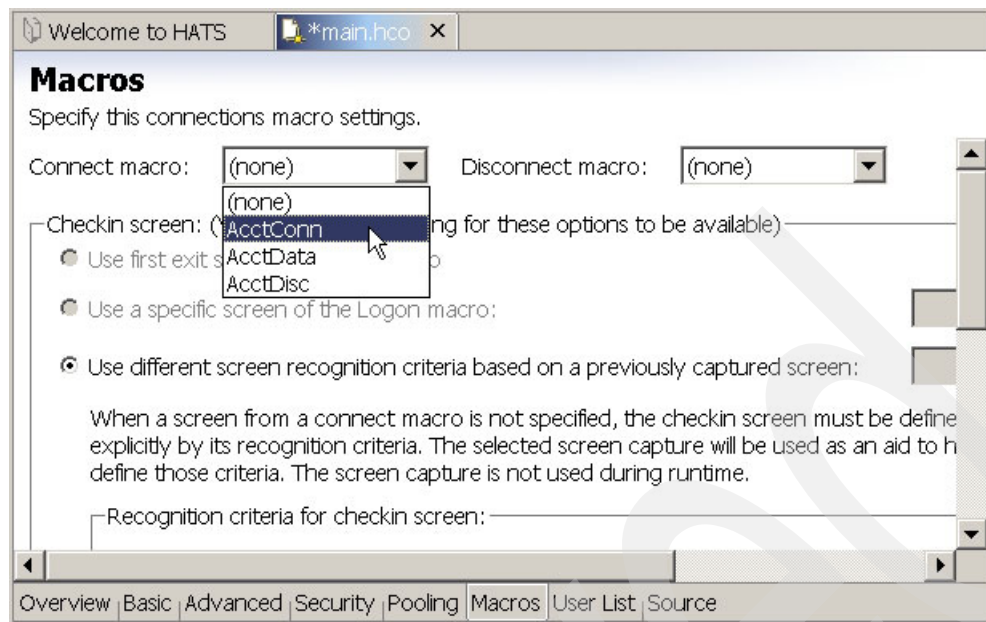


Figure D-25 Connection macro settings

2. For the Disconnect macro, select your Disconnect macro AcctDisc. Close the editor for your main connection by clicking the **X** on the main.hco editor tab; see Figure D-26.

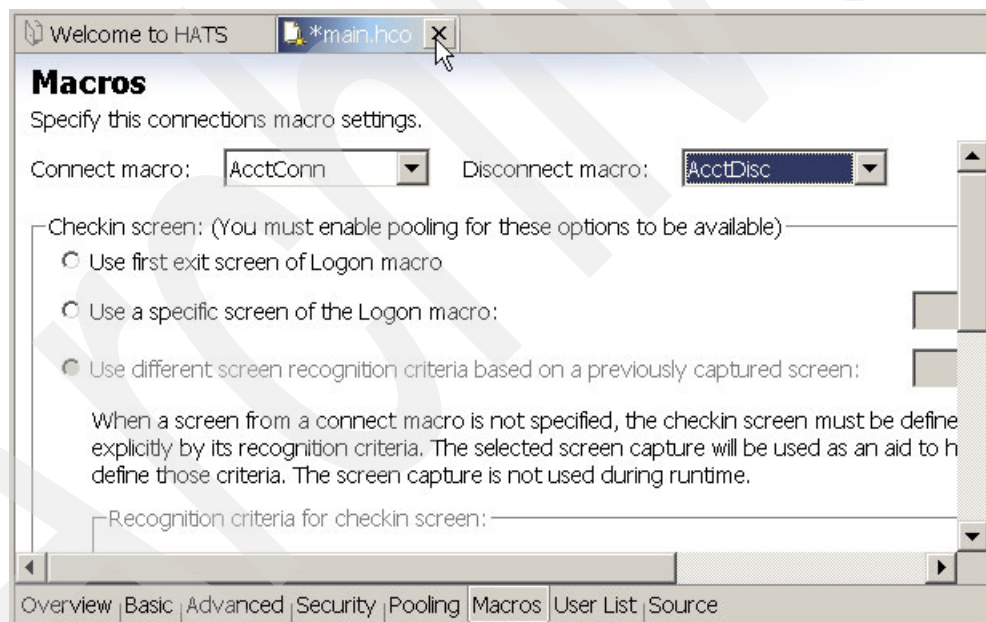


Figure D-26 Disconnection macro settings

3. When asked if you want to save the changes, click **Yes**.

Enable display terminal function

HATS provides a function called Display Terminal that will show a display screen for active host connections. This function is intended for use while developing and debugging HATS applications. Later in this lab you will test your Web service. By enabling the Display Terminal function, you will be able to see the host connection your Web service is using.

1. To enable the Display Terminal function, click the **Navigator** tab to switch to the Navigator view; see Figure D-27.

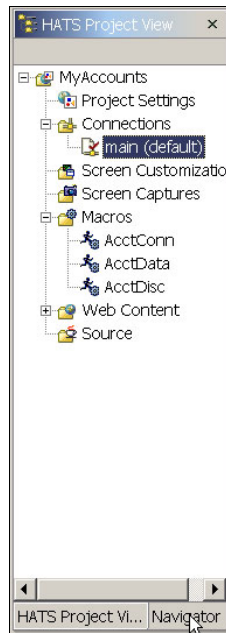


Figure D-27 Navigator view tab

2. In the Navigator view, open the folder for the ear file and double-click the runtime.properties file; see Figure D-28.

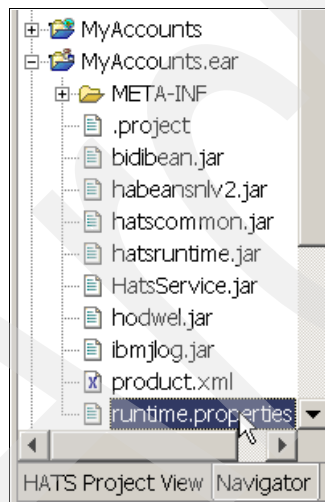


Figure D-28 Navigator - runtime.properties

3. In the editor for the runtime.properties file, add the line: `trace .HOD.DISPLAYTERMINAL=1` as shown in Figure D-29 on page 193, then close the editor for runtime.properties by clicking the **X**.

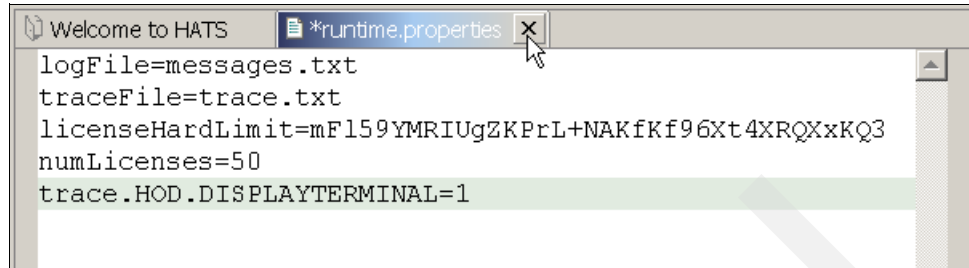


Figure D-29 runtime.properties - source

4. When asked if you want to save changes, click **Yes**.

Now switch from the Navigator view back to the HATS project view.

Create integration object

A HATS Integration Object is a JavaBean that encapsulates a programmed interaction with a host application. Integration Objects can be used in multiple ways to integrate interaction with a host application into new Java or Web-based programs. One use of an Integration Object is to provide the interaction with a host application for a Web service. HATS macros also provide a programmed interaction with a host application. In fact, creating a HATS macro is the first step in creating an Integration Object.

The Web service you are building in this project is intended to gather account information from the host system based on an account number submitted by the Web service client. You have just finished creating a HATS Data macro that does just that. So having created your Data macro, you are finished with the “hard” part of developing your Web service. Now all you need to do is tell HATS to create an Integration Object from your Data macro and then create a Web service from the Integration Object.

To create an Integration Object from your Data macro, in the HATS Project View open the Macros folder, right-click your Data macro AcctData and select **Create Integration Object**; see Figure D-30 on page 194.

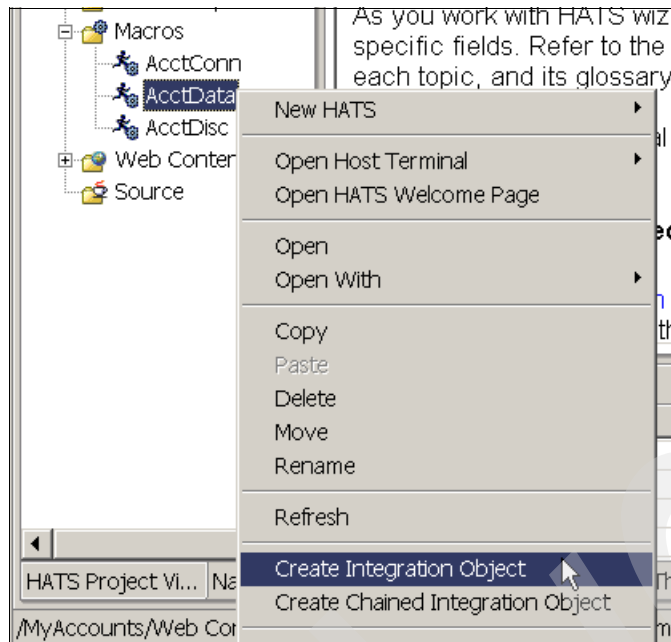


Figure D-30 Create Integration Object

After HATS finishes creating the Integration Object, you will find it in the HATS Project View in the Source/Integration Object folder. Look for your Integration Object now; see Figure D-31. Notice it has the same name as the macro used to create it; your Integration Object name is AcctData.

Because of the required naming conventions, if your macro had started with a lower case letter, HATS would have converted it to upper case in the Integration Object name.

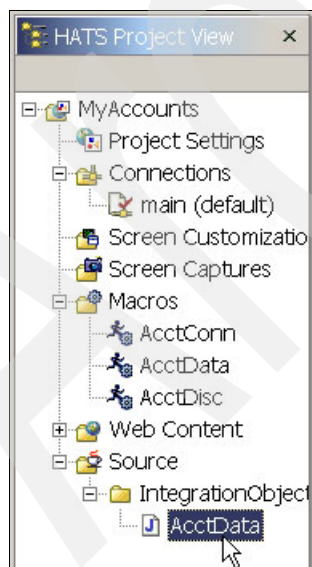


Figure D-31 AcctData Integration Object

Next, you will begin the process of creating a Web service.

Create Web service support files

Before you actually create your Web service, you must first create HATS Web service support files.

1. To do this, in the HATS Project View, right-click your Integration Object and select **Create Web Service Support Files**; see Figure D-32.

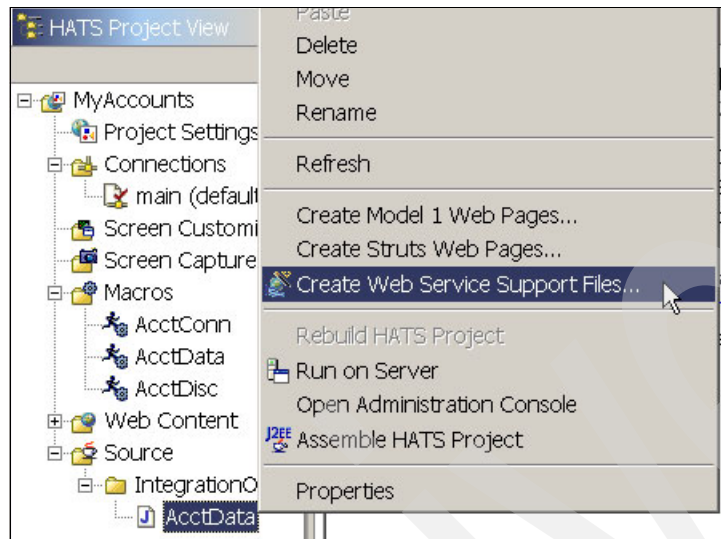


Figure D-32 Create Web Services Support Files

2. For Project, use your project name MyAccounts. For Class name, enter: AcctWS (since this is a class name, it must start with an upper case letter). Click **Next**; see Figure D-33.

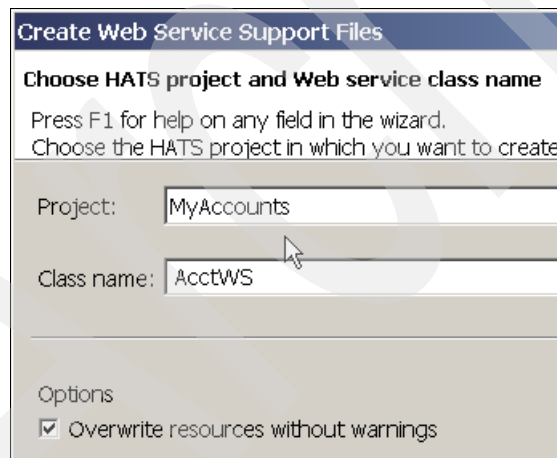


Figure D-33 Choose HATS project panel

3. Check your Integration Object **AcctData** to include it in the Web service class. You can include more than one Integration Object in a Web service class; in this case, however, you only have the one. Click **Finish**; see Figure D-34 on page 196.

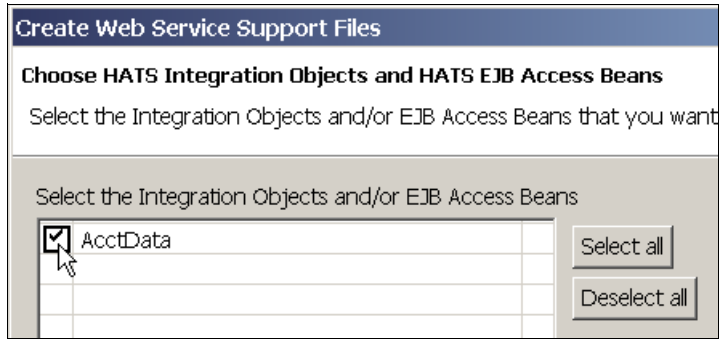


Figure D-34 Choose HATS Integration Object panel

4. In the HATS Project View, notice the webServiceClasses folder and files that have been created, as shown in Figure D-35.

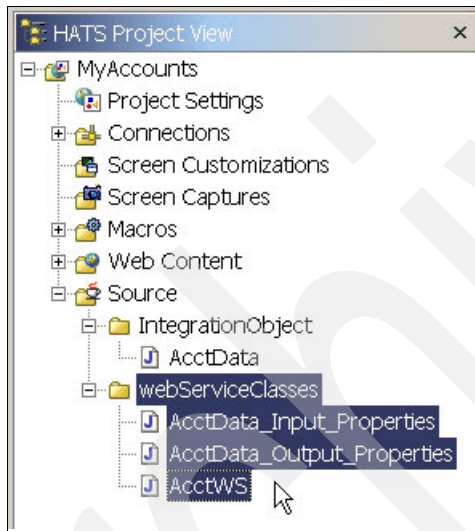


Figure D-35 AcctData web services classes

5. Switch to the Navigator view and notice the same files and additional files added to the IntegrationObject folder, as shown in Figure D-36 on page 197.

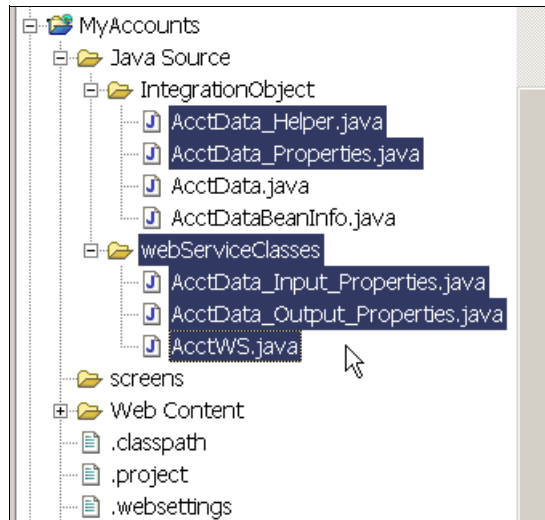


Figure D-36 Integration Object / webServices Classes folders

You can now deploy the Web service, as shown in Figure D-37.

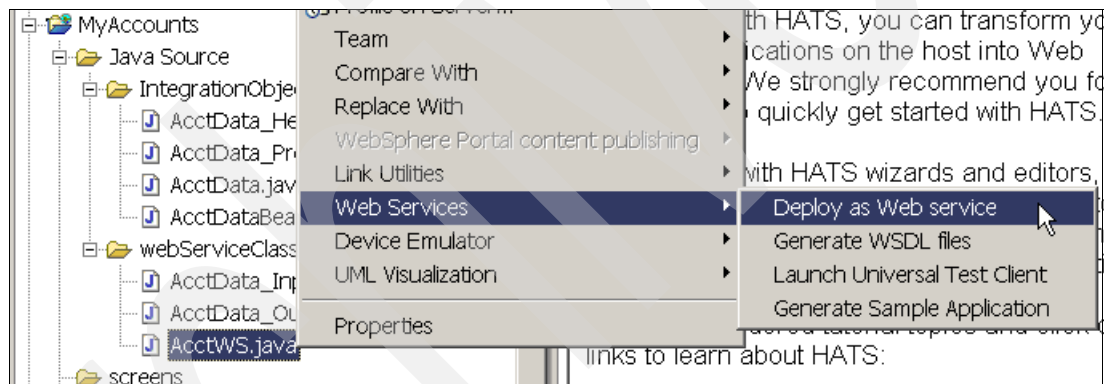


Figure D-37 Deploy Web Service

Now that you have created your Web service support files, you can create your Web service and deploy it to the WebSphere Test Environment. To do this, in the Navigator view right-click on your Web service class file **AcctWS.java** in the Java Source\webServiceClasses folder and select **Web Services -> Deploy as Web service**.

1. On the Service Deployment Configuration panel, notice that by default the server to which your Web service will be deployed is the WebSphere v5.1 Test Environment, as shown in Figure D-38 on page 198.

HATS 5.0.1 is required to run with WebSphere Application Server (WAS) 5.1. These instructions are written at the HATS 5.0 level. Click the **Edit** button to change the server to use the WebSphere v5.0 Test Environment.

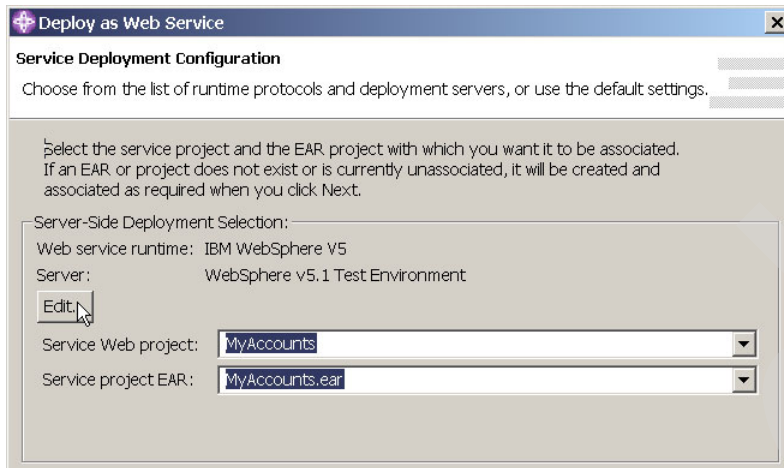


Figure D-38 Service Deployment Configuration panel

2. Scroll down and select **WebSphere v5.0 Test Environment**, then click **OK**; see Figure D-39.

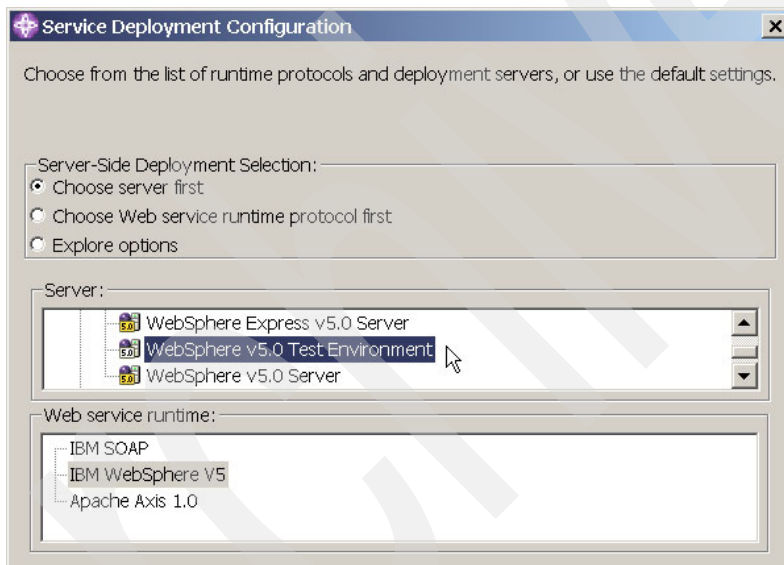


Figure D-39 Configure web services runtime environment

3. Accept the suggested Service Web project and Service project EAR names. At this point you could click **Finish** and take all of the defaults. But to see other possible settings, click **Next**, as shown in Figure D-40 on page 199.

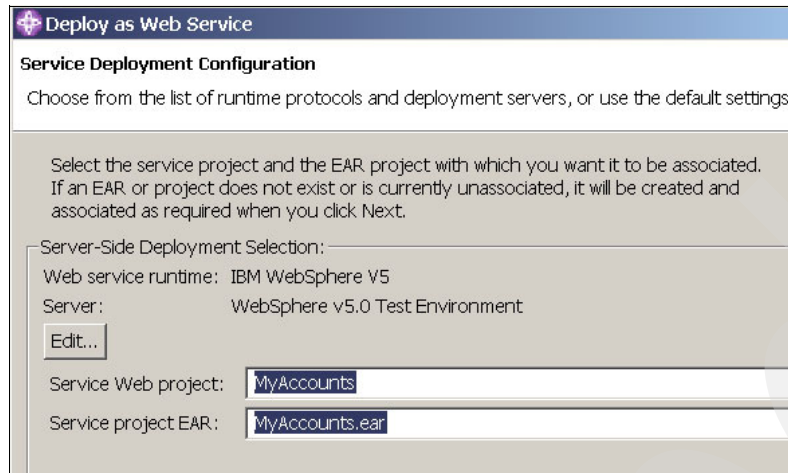


Figure D-40 Service Deployment Configuration panel

4. Notice the Bean name; do not change it. Click **Next**, see Figure D-41.

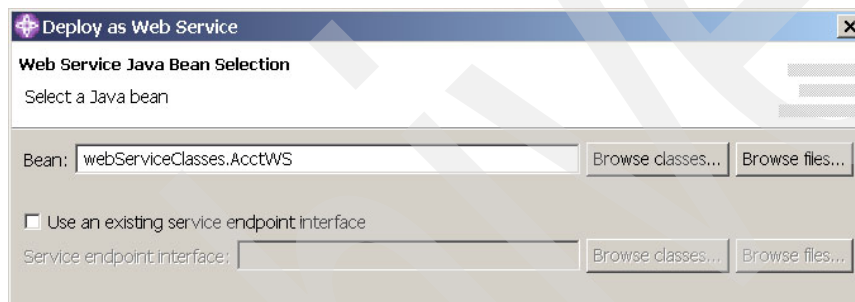


Figure D-41 Java Bean selection panel

5. Notice all of the Web service settings. The methods correspond to the Integration Objects you included in the Web service class file. Do not change anything. Click **Finish**; see Figure D-42 on page 200.

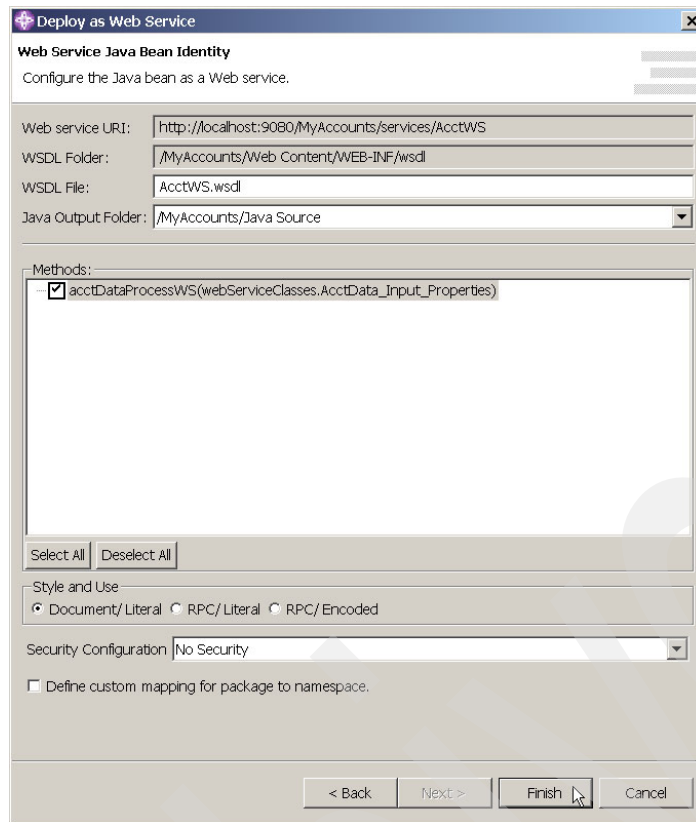


Figure D-42 Java Bean Identity panel

6. At this point your HATS application is published to the WebSphere Application Server Test Environment and the WAS server is started as shown in Figure D-43. An actual copy of WAS is brought up within the studio and your HATS application and started. This may take a few moments. Notice the activity in the Console and look for the message: Server server1 open for e-business.

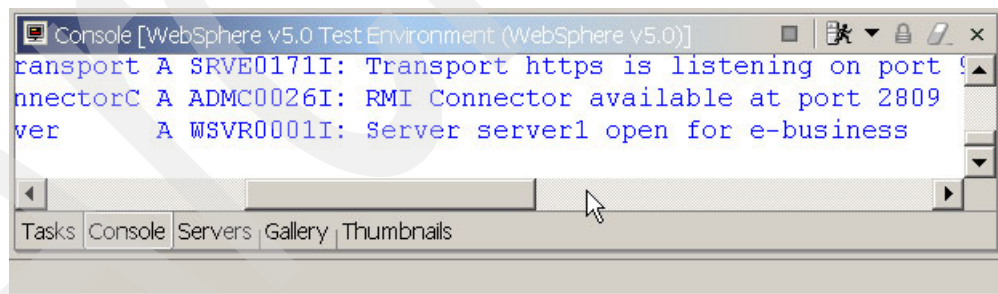


Figure D-43 WebStudio Console

When your HATS application is started, the connection to the host system is started and the Connect macro is run, leaving the connection at the Operator Instructions screen. You can actually see this by using the Display Terminal function.

Earlier in the lab you enabled the Display Terminal function in the runtime.properties file. Notice that a host terminal screen has become active. (Sometimes this task starts minimized on the Windows task bar, so you may need to look for it there and restore it.)

Notice that it is displaying the Operator Instructions screen. The end user does not see this screen; it is only displayed at the server for debugging purposes. Now, when you invoke your

Web service, you can see how HATS is using the host connection to communicate with the host application.

Your Web service is now deployed. Next, you will test it.

Test Web service

One output of creating your Web service is a Web Service Description Language (WSDL) file. You can find this file in the Navigator view in the Web Content\wsdl\webServiceClasses folder. This WSDL file can be used to test your Web service.

1. To do so, right-click on the WSDL file, AcctWS.wsdl, then select **Web Services -> Test** with Web Services Explorer, as shown in Figure D-44.

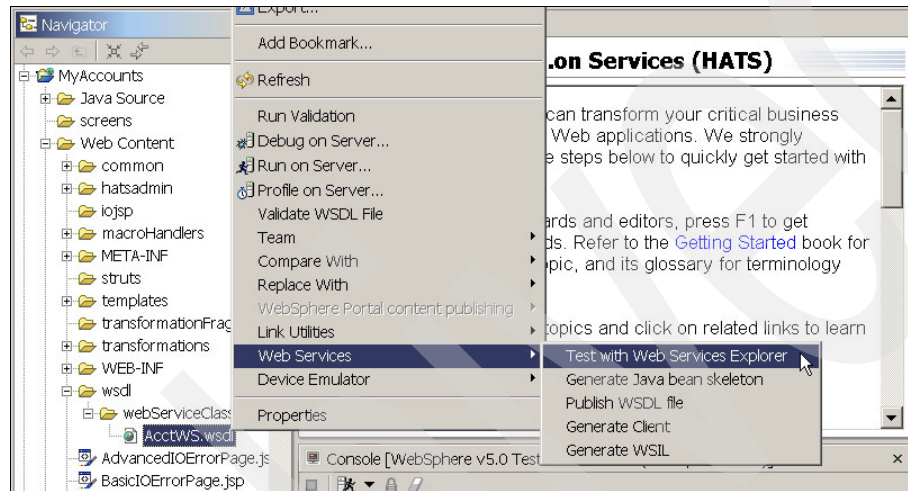


Figure D-44 Start Web Services Explorer

2. The Web Services Explorer comes up in the studio's Web Browser. Double-click the **Web Browser** tab to maximize it.
3. In the Actions panel under Operations, click the **acctDataProcessWS** operation, as shown in Figure D-45.

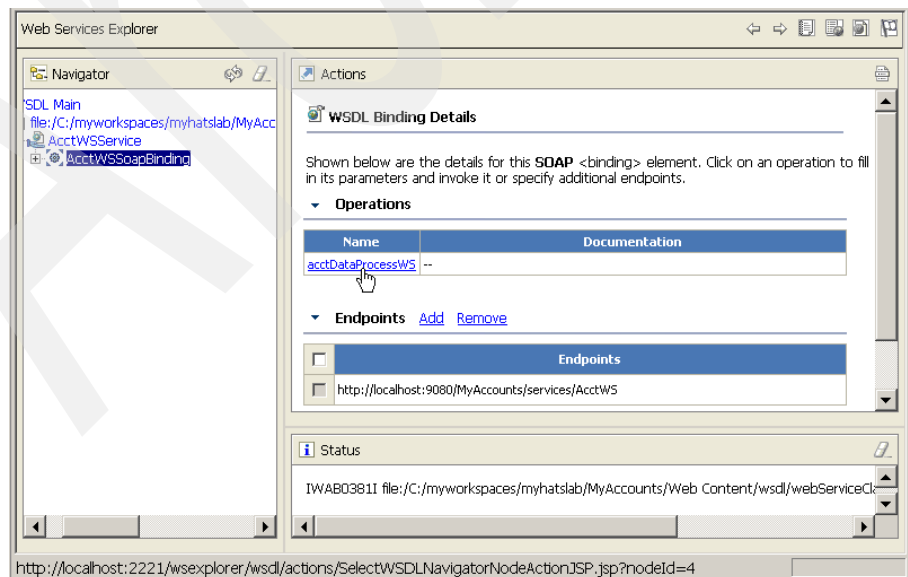


Figure D-45 Web Services Explorer

4. Under `inputFromClient`, notice `acctSearchValue`. Remember, this is the name of the macro prompt you created in your Data macro to allow the client to provide an account number to search.

In this field, enter account number 11111; see Figure D-46.

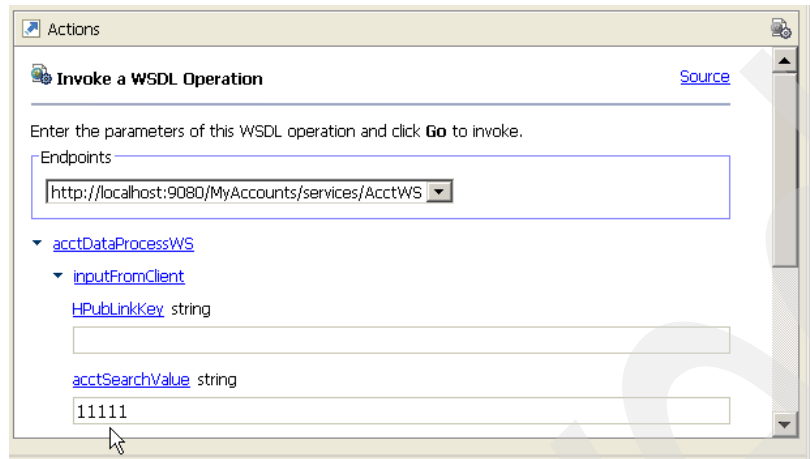


Figure D-46 Web Services parameter entry panel

5. So that you can actually see the communication with the host application when the Web service is invoked, you will need to reduce the size of the studio window so that you can see both the studio's Web browser and the host terminal screen at the same time.

In the Web Services Explorer Actions panel, scroll down and click the **Go** button, as shown in Figure D-47.

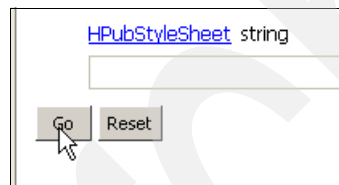


Figure D-47 Run Web Service

At this point, a connection is allocated from the connection pool. Your Integration Object is instantiated and then navigates through the host application, using the search value supplied by the client. Notice the host terminal screen as the Integration Object runs.

6. In the Web Services Explorer, double-click the **Status** panel to maximize it. In the Form view, scroll down and notice the outputs. Remember these are the macro extracts you created in your Data macro. For example, in Figure D-48 on page 203, notice `acctAmount` and `acctNumber`, and so forth.

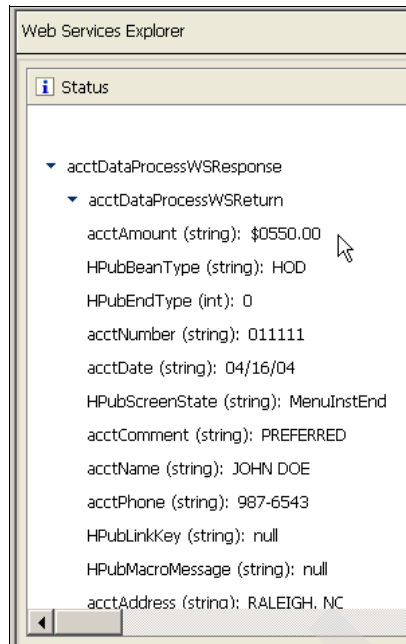


Figure D-48 Web Services Explorer - status

7. Now scroll back up and switch to the Source view, as shown in Figure D-49.

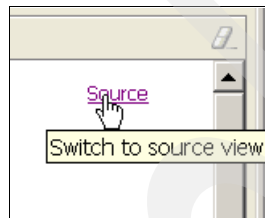


Figure D-49 Switch to source view

8. In the Source view, notice the SOAP Request Envelope and SOAP Response Envelope; see Figure D-50.

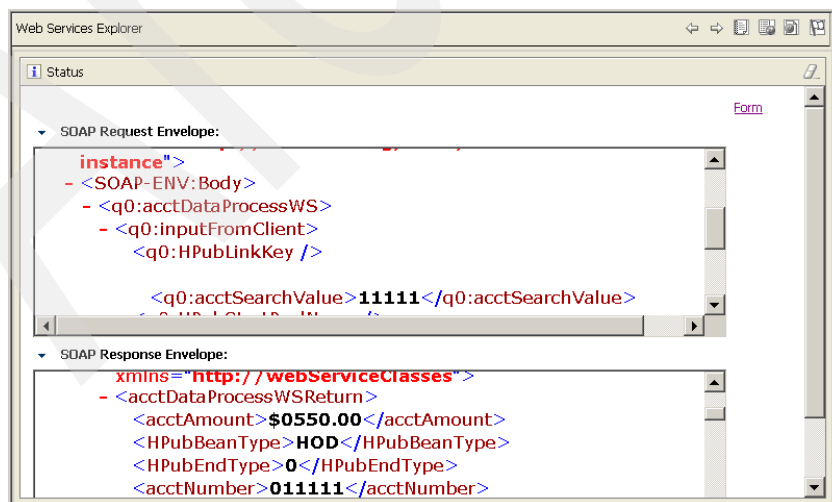


Figure D-50 SOAP Request/Response envelope view

You have now finished the creating and testing your Web service. Next, you can use the studio to create a sample client application that can use your Web service.

Create and test Web service client

To create a sample client application that can use your Web service, go back to the Navigator view and find your WSDL file AcctWS.wsdl in the Web Content\wsdl\webServiceClasses folder.

1. Right-click the WSDL file, then select **Web Services** -> **Generate Client** as shown in Figure D-51.

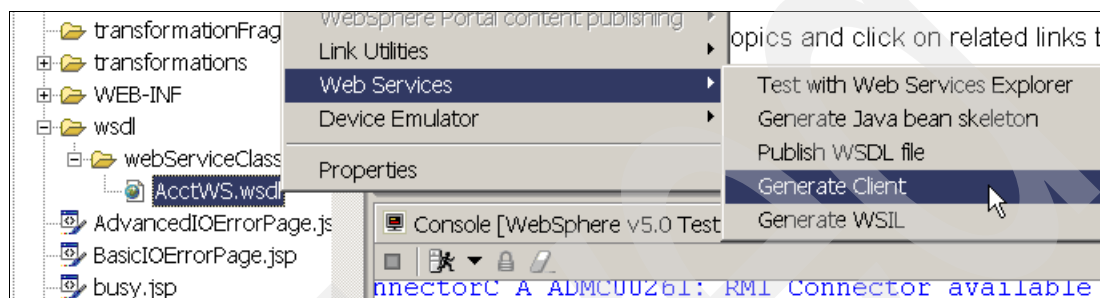


Figure D-51 Create Web Service client

2. On the Web Services panel, check Test the generated Proxy and click **Next**; see Figure D-52.

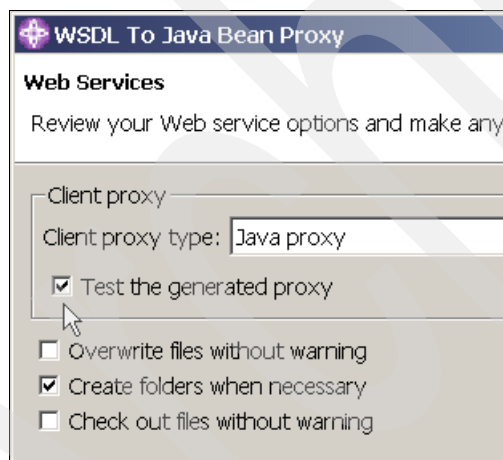


Figure D-52 Web Service options panel

3. On the Client Environment Configuration panel, do the following:
 - a. Select **Explore options**, as shown in Figure D-53 on page 205.
 - b. Under Web service runtime, be sure IBM WebSphere V5 is selected. This has to match the service runtime you selected for your Web service.
 - c. Under Server -> **Existing Servers** -> **WebSphere v5.0 Test Environment**. Remember, HATS 5.0.1 is required to use WAS v5.1. These instructions are written using HATS 5.0, so you must select a WebSphere v5.0 Test Environment. In this case, you will use the server that was created when you deployed your Web service.
 - d. Your sample client must be in a project different from your Web service project. So, for Client project, enter: MyAccountsWSClient, and for Client project EAR, enter: MyAccountsWSClient.ear.

e. Click Next.

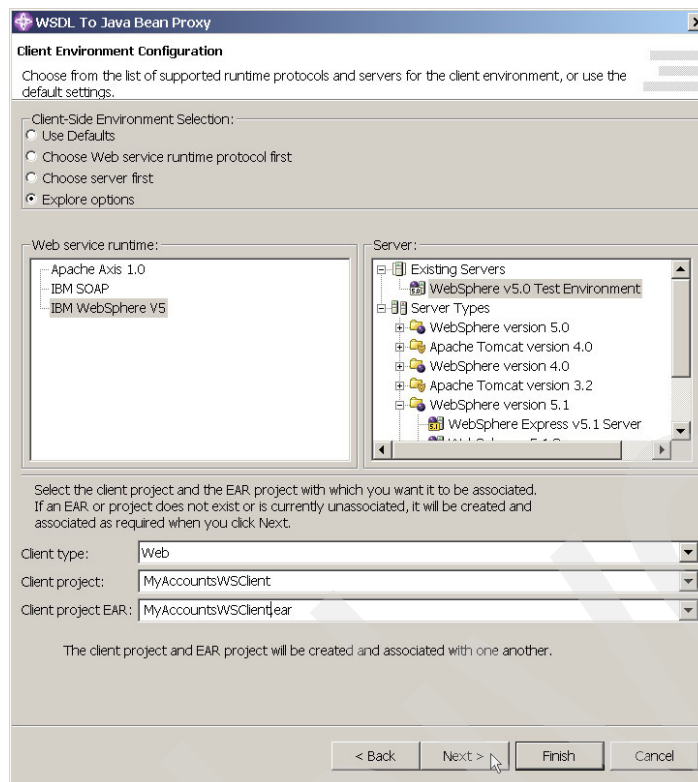


Figure D-53 Client Environment configuration panel

4. On the Web Service Selection Page, notice the URI to your WSDL file. Do not change this. Click **Next**; see Figure D-54.

2.

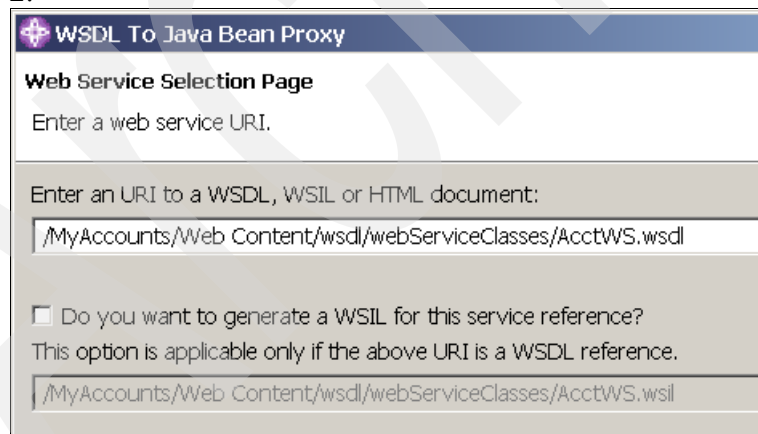
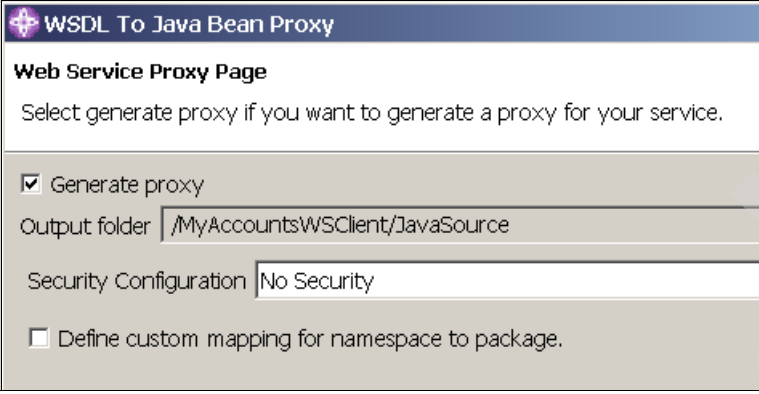


Figure D-54 Web Service Selection panel

5. On the Web Service Proxy Page, be sure to check **Generate proxy** and notice the output folder for your proxy client. Click **Next**; see Figure D-55 on page 206.



WSDL To Java Bean Proxy

Web Service Proxy Page

Select generate proxy if you want to generate a proxy for your service.

☒ Generate proxy

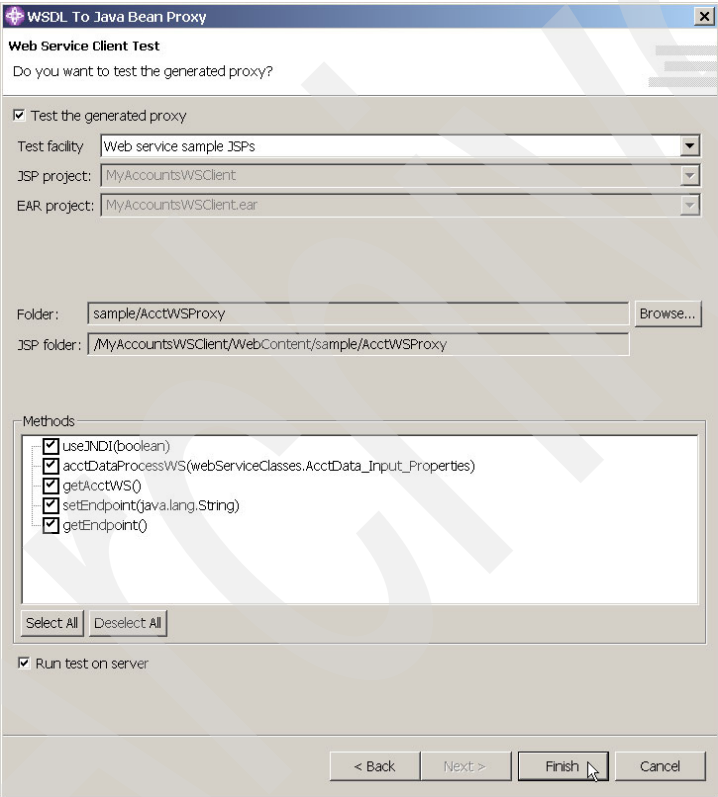
Output folder: /MyAccountsWSClient/JavaSource

Security Configuration: No Security

☐ Define custom mapping for namespace to package.

Figure D-55 Web Service Proxy panel

- On the Web Service Client Test panel, be sure to check Test the generated proxy. Click **Finish**; see Figure D-56.



WSDL To Java Bean Proxy

Web Service Client Test

Do you want to test the generated proxy?

☒ Test the generated proxy

Test facility: Web service sample JSPs

JSP project: MyAccountsWSClient

EAR project: MyAccountsWSClient.ear

Folder: sample/AcctWSPProxy Browse...

JSP folder: /MyAccountsWSClient/WebContent/sample/AcctWSPProxy

Methods

- ☒ useJNDI(boolean)
- ☒ acctDataProcessWS(webServiceClasses.AcctData_Input_Properties)
- ☒ getAcctWS()
- ☒ setEndpoint(java.lang.String)
- ☒ getEndpoint()

Select All Deselect All

☒ Run test on server

< Back Next > Finish Cancel

Figure D-56 Web Service Client test panel

- At this point your client application is generated, published to the server, and started. This may take a few moments.

In the Navigator view, notice your new client project `MyAccountsWSClient`, as shown in Figure D-57 on page 207. Also notice `TestClient.jsp`. When this jsp is run on the server, it can be used to drive your client application.

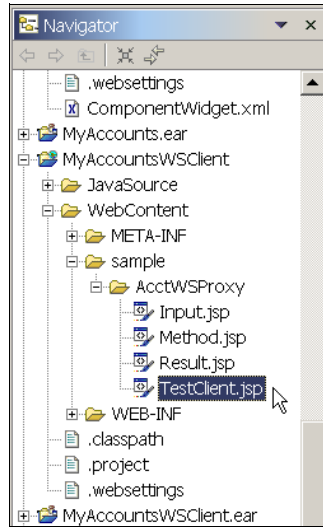


Figure D-57 HATS project navigator - TestClient.jsp

8. Now notice that the TestClient.jsp has automatically been run on the server and is displayed in the studio's Web Browser. To test the client application, in the Methods frame, click **acctDataProcessWS**; see Figure D-58.

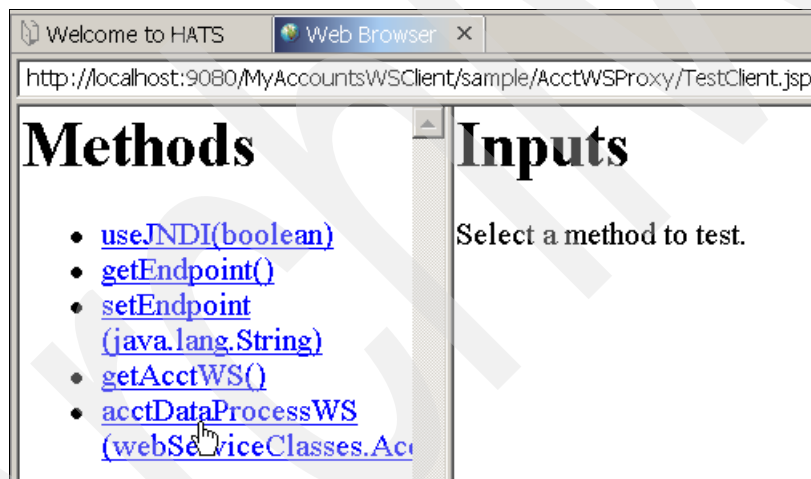


Figure D-58 Browser - TestClient.jsp

9. As you did when you tested using the Web Services Explorer, reduce the size of the studio's window so that you can see both the studio and the host terminal screen. Then in the Inputs frame for acctSearchValue, enter: 11111 and click **Invoke**, as shown in Figure D-59 on page 208.

Welcome to HATS Web Browser x

http://localhost:9080/MyAccountsWSClient/sample/AcctWSProxy/TestClient.jsp

Methods

- [useJNDI\(boolean\)](#)
- [getEndpoint\(\)](#)
- [setEndpoint\(java.lang.String\)](#)
- [getAcctWS\(\)](#)
- [acctDataProcessWS\(webServiceClasses.AcctDataProcessWS\)](#)

Inputs

inputFromClient:

hPubStartPoolName:

acctSearchValue:

hPubAccessHandle:

hPubStyleSheet:

hPubLinkKey:

Invoke Clear

Figure D-59 Test client parameter input panel

Notice the activity in the host terminal screen.

Look for the results in the Result frame of your TestClient.jsp. You may need to scroll down to see the results. Notice the same outputs that correspond to the macro extracts of your Data macro as shown in Figure D-60.

acctNumber: 011111

hPubScreenState: MenuInstEnd

acctDate: 04/16/04

acctComment: PREFERRED

acctName: JOHN DOE

hPubAccessHandle: null

acctPhone: 987-6543

hPubLinkKey: null

acctAddress: RALEIGH, NC

hPubStartChainName: null

hPubXMLData: null

acctSearchValue: 11111

Figure D-60 TestClient.jsp results

Congratulations! You have successfully created a HATS Web service, tested it with the Web Services Explorer, created a sample client to use your Web service, and tested it, as well.

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

IBM Redbooks

For information on ordering these publications, see “How to get IBM Redbooks” on page 210. Note that some of the documents referenced here may be available in softcopy only.

- ▶ *WebSphere Studio Application Developer Programming Guide*, SG24-6585
- ▶ *WebSphere Studio Application Developer Version 5 Programming Guide*, SG24-6957
- ▶ *Host Access Transformation Server Concepts and Architecture*, REDP-3706
- ▶ *Creating and Modifying HATS Projects*, REDP-3698

Other publications

These publications are also relevant as further information sources:

- ▶ *IBM Host Access Transformation Server InfoCenter*
<http://www-3.ibm.com/software/webservers/hats/library/infocenter/>
- ▶ *IBM Host On-Demand InfoCenter*
<http://www-3.ibm.com/software/webservers/hostondemand/library/infocentergafinal/hod/en/help/2tabcontents.html>
- ▶ *IBM WebSphere Host Access Transformation Services V5 Getting Started*, SC31-6574
- ▶ *IBM WebSphere Host Access Transformation Services V5 User's and Administrator's Guide*, SC31-6575
- ▶ *IBM WebSphere Host Access Transformation Services V5 Programmer's Guide*, SC31-6576
- ▶ *IBM WebSphere Host Access Transformation Services V5 Advanced Macro Guide*, SC31-6590

Online resources

These Web sites and URLs are also relevant as further information sources:

- ▶ HATS Information Center
<http://www.ibm.com/software/webservers/hats/library/version5/infocenter>
- ▶ HATS general information
<http://www.ibm.com/software/webservers/hats/>
- ▶ HATS Forum
<http://news://news.software.ibm.com/ibm.software.websphere.hats>

- ▶ HATS demonstrations
<http://websphere.dfw.ibm.com/whidemo/>
- ▶ WebSphere Application Developer information
<http://www.ibm.com/software/awdtools/studioappdev/library>
- ▶ WebSphere Application Server V5 Information Center
<http://publib.boulder.ibm.com/infocenter/wasinfo/index.jsp>
- ▶ WebSphere Application Server for z/OS information
http://www.ibm.com/software/webservers/appserv/zos_os390/library/

How to get IBM Redbooks

You can search for, view, or download Redbooks, Redpapers, Hints and Tips, draft publications and Additional materials, as well as order hardcopy Redbooks or CD-ROMs, at this Web site:

ibm.com/redbooks

Help from IBM

IBM Support and downloads:

ibm.com/support

IBM Global Services:

ibm.com/services

Index

A

- address space 68
- address spaces 67
- Application Assist Processor 89
- application server model 69
 - instances 70
 - nodes 70
 - regions 70
 - servers 70
- architecture 4

B

- business goals 79

C

- central processors 66
- Channel Subsystem 66
- CLASSPATH 76
- cluster 79
- configuring display terminal 92
- connection settings 27
- connection settings panel 28
- control region 70
- Coupling Facility 67
- create first project 29
- creating HATS project 26
- customized application 39
- customizing default rendering options 42

D

- default application 31
- default HATS application 25
- default template 28
- default template panel 29
- default transformation 32
- default transformation functionality 32
- deploying HATS application 80
- deployment 63

E

- editors 26
- emulation primer 14
 - character set 14
 - color 14
 - extended highlighting 14
 - field validation 14
- enterprise cluster 79
- executive overview 3

G

- graphics support 91

H

HATS

- architecture 4
- configure connection settings 27
- connection settings panel 28
- create first project 29
- create project panel 27
- creating project 26
- customized application 39
- default application 25, 31
 - communication settings 25
 - default template 25
 - introduction 25
 - project 25
- default template panel 29
- default transformation functionality 32
- deployment 63
- editors 26
- highlights 7
- modifying default template 40
- overview 8
- rendering 42
- selecting default template 28
- service packs 109
 - service pack 3 111
 - service pack 4 109
- version 5.0.4 97
 - new functions 97
- WebSphere Application Server 80
 - console 81
 - export 81
 - install application screen 86
 - installation 83
 - installation options 84
 - installation summary 86
 - map modules 85
 - map security roles 85
 - map virtual hosts 84
 - master config confirmation 87
 - master configuration 87
 - project 80
 - select application 83
 - start application 88
 - web server plug-in 88
- WebSphere for z/OS 77
- welcome panel 26
- wizards 26
- z/OS 78
- HATS applications on z/OS 80
- hierarchical file system (HFS) 68
- highlights 7

I

- introduction 1

J

Java Virtual Machine 76

JVM 76

L

lab 1

- create HATS application 115
 - change project settings 125
- CICS file browse screen 157
 - capture 157
 - customization 164
 - event handler JSP 166
 - record macro 157
 - test 166
- CICS file inquiry screen 170
 - error 170
- CICS NACT accounts menu error screen 172
- CICS NACT accounts screen 171
- CICS NACT details of account 171
- CICS operator screen 145
 - capture 145
 - customization 146
 - test 156
 - transformation 148
- default project 116
- screen customizations 128
 - blank 129
 - CICS menu 129
 - record macro 130
 - screen capture 130
 - sign-on 129
 - test screen transformation 137
 - transformation 133
 - welcome 129
- sign-on screen 137
 - capture 137
 - customization 138
 - test 145
 - transformation 140
- stylesheets 120
- template 120
- test and project setting changes 127
- test project 119
- welcome screen 130
 - customization 131
 - record macro 130
 - screen capture 130
 - test 137
 - transformation 133

lab1

requirements 115

lab2

- installation requirements 175
- legacy applications 175
 - connection parameters 188
 - enable pooling 188
- create macros 179
 - record connect 179
 - record data 181

record disconnect 187

test connect 181

test data 185

test disconnect 187

create project 176

integration object 193

Web service support files 195

web services 175

naming conventions 176

LDAP 75

logical partition (LPAR) 66

logical partitions 66

LPAR 67

M

modifying default template 40

N

node 71

O

overview 8

P

package 69

Parallel Sysplex 67

Performance 91

performance

settings 91

values 91

performance components 74

product overview 7

project 26

project panel 27

R

Redbooks Web site 210

Contact us xi

rendering 42

rendering options selections 49

runtime 65

S

screens

3270 11

security 75

server instance 70

server region 70

service pack 109

subtask 70

sysplex 79

Sysplex Timer 67

T

tasks 67

TCPIP network 74

thread 70

U

UNIX System Services 68

UNIX Systems Services 68, 75

W

WebSphere 65, 76

WebSphere for z/OS 77

WebSphere on sysplex 79

WebSphere on z/OS 66

WebSphere programming model 69

 java overview 69

 Java programming model 69

 Java runtime execution 69

welcome panel 26

wizards 26

Workload Manager 68, 75

X

XCF 67

Z

z/OS 75

 availability 78

 efficient 78

 HATS 78

 integration 78

 JVM 76

 LDAP 75

 secure 78

 security 75

 selectivity 78

 UNIX System Services 75

 WebSphere 76

 Workload Manager 75

z/OS architecture 78

z/OS benefits 80

z/OS components 68

 workload manager 68

z/OS updates 92

z/OS version updates

 1.5 92

 1.6 92

 z/OS.e 92

 zSeries 800 92

z/OS work prioritization 78

zAAP 89

 overview 89

 prerequisites 90

 use 89

 working 90

zSeries hardware 66

zSeries server 74

zSeries zAAP 89

ZYX Electronics

 application 11

 ordering system 13

 ZYX Parts application 15

ZYX Parts 15

 customized 39, 49, 59

 action panel 48

 add action panel 52

 button attributes 47

 button settings 50

 bypassing command screen 48

 create screen panel 44

 default template 42

 export project 56

 export zip file wizard 57

 exporting 56

 host component insert 54

 host key panel 46

 import 56

 import zip file wizard 58

 logon screen 43

 modifying default template 40

 new logon screen 60

 new main menu 60

 new product search page 61

 new template panel 40

 new template source panel 41

 parts main menu 49

 parts order form 52

 parts search screen 53

 rendering configuration panel 42

 rendering options 42, 45

 review 60

 screen option 43

 search results 55

 send key action 53

 WSAD style sheet editor 51

 default transformation 32

 logon screen 32

 main menu 34

 order screen 35

 search results screen 36–37

 search screen 36

 VM command screen 33

 existing user interface 13

 login 15

 login screen 15

 main menu 16

 main menu screen 17

 order form 17

 order screen 18

 search results 19

 search results screen 20–21

 search screen 19

 searching 18

 skills 13

 starting the application 15

 usability 13

 VM command screen 16

 workflow 13

Archived



Host Access Transformation Services on z/OS



Redbooks

**HATS version 5.0.4 for
z/OS**

**Enterprise
modernization
examples**

**Performance and
deployment guide**

This IBM Redbook provides an overview of HATS 5.0.4 on z/OS. Many of these improvements are available on the distributed platforms as well. Therefore, this publication should be of value to any HATS v5.0.4 user.

This redbook is intended for IT managers, application developers, system administrators and architects. The topics presented use as examples HATS applications that front-end 3270 programs and run on WebSphere for z/OS. Many of the concepts in this book also apply to 5250 applications, VT applications, and the other WebSphere runtime platforms.

The benefits of HATS are illustrated using examples. Readers then have opportunities to exercise what they have learned by completing hands-on labs.

Deployment examples and considerations are presented. This includes discussion of the performance, scalability and reliability benefits of running HATS applications on z/OS.

At the end of the book, readers should have a fundamental understanding of the product and be able to create, customize and deploy a HATS application.

**INTERNATIONAL
TECHNICAL
SUPPORT
ORGANIZATION**

**BUILDING TECHNICAL
INFORMATION BASED ON
PRACTICAL EXPERIENCE**

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

**For more information:
ibm.com/redbooks**

SG24-6479-00

ISBN0738492221