IBM Host Access Transformation Services

# User's and Administrator's Guide

*Version 9.7*

SC27-5904-03

IBM Host Access Transformation Services

# User's and Administrator's Guide

*Version 9.7*

SC27-5904-03

**Note**

Before using this information and the product it supports, be sure to read the general information under Appendix C, "Notices," on page 491.

**Eleventh Edition (November 2018**

# Contents

# Chapter 1. Using Host Access Transformation Services (HATS)

With IBM® Host Access Transformation Services (HATS), you can create Web applications, and rich client applications that provide an easy-to-use graphical user interface (GUI) for your 3270 applications running on IBM z Systems platforms and your 5250 applications running on IBM i operating system platforms. HATS applications can access 5250 applications without requiring Online Transaction Processing (OLTP) capacity. In this book, these character-based 3270 and 5250 applications are referred to as host applications. You can also create service-oriented architecture (SOA) assets using Web services that provide standard programming interfaces to business logic and transactions contained within host applications. Data from video terminal (VT) emulation screens can also be accessed.

HATS applications can be given a modern appearance. HATS Web applications can be developed with an interface that matches your company's Web or portal pages, and your users can access them through their Web browsers. HATS Web applications can also be developed to provide access to your host applications from mobile devices such as cellular phones, data collection terminals, and personal digital assistants (PDAs).

HATS rich client applications can be developed to run in an Eclipse Rich Client Platform (RCP) implementation, in Lotus Notes®, or in the Lotus® Expeditor Client to provide native client applications targeted for a user's desktop. For more information about the Eclipse RCP environment see http://www.eclipse.org/home/categories/rcp.php. For more information about Lotus Notes see the Lotus Notes Knowledge Center at http://www-01.ibm.com/support/knowledgecenter/SSKTMJ_8.5.3/welcome_Domino_8_5_3.html. For more information about the Lotus Expeditor Client see the Lotus Expeditor Knowledge Center at http://www-01.ibm.com/support/knowledgecenter/SSVHEW_6.2.0/welcome.html.

For information on what is new in HATS V9.7 see *HATS Getting Started*.

**Note:** Throughout this document, configuration settings that apply only to HATS Web applications and EJB applications unless otherwise noted, are marked with **Web-only** . Configuration settings that apply only to HATS rich client platform applications are marked with **RCP-only** .

HATS has two components:

- The HATS Toolkit is a set of plug-ins for the Eclipse-based IBM Rational® Software Delivery Platform (Rational SDP). For more on which versions of Rational SDP are supported, see the section, Prerequisites, in *HATS Getting Started*.

  The HATS Toolkit enables you to develop new applications, a step at a time, previewing and saving each change you make. Over time, or as quickly as you like, you can streamline your HATS application, making it easier to use than the host applications whose data it presents, and possibly moving functions from the host applications into your HATS applications. The development process for building HATS Web and rich client applications is similar. For more information, see "Understanding HATS application development" on page 10.

After you have developed a HATS application, you deploy it to a production runtime environment.

- The HATS runtime code runs as part of a HATS application deployed in the production runtime environment, WebSphere® Application Server, WebSphere Portal, Apache Geronimo web application server with the Tomcat web container, Oracle WebLogic Server, or rich client platform. Users interact with the HATS application through the HATS GUI and data is sent back and forth between the user and the host application.

For more information see Chapter 3, "Developing and deploying HATS Web applications," on page 27 and Chapter 4, "Developing and deploying HATS rich client applications," on page 53.

## Understanding HATS key concepts and objects

This section explains key concepts and objects of HATS, some of which are described in "Understanding HATS application processing" on page 8. Many key objects in HATS are created using a wizard, but are viewed or modified after creation using an editor.

**Project**
A collection of HATS resources (also called artifacts) that are created using HATS Toolkit wizards and customized using HATS Toolkit editors. These resources are exported as a HATS application. There are five types of HATS projects: Web, EJB, rich client, and for purposes of administering HATS Web (including and EJB) applications, HATS administrative console projects. For more information see Chapter 3, "Developing and deploying HATS Web applications," on page 27.

**Event**
A HATS resource that performs a set of actions based on a certain state being reached. There are two types of HATS events, application events and screen events.

**Application event**
A HATS event that is triggered by state changes in the application's life cycle. Examples of application events include a user first accessing a HATS application (a Start event), or an application encountering an unrecognized screen (an Unmatched Screen event). For more information see "Application events" on page 102.

**Screen event**
A HATS event that is triggered when a host screen is recognized by matching specific screen recognition criteria. There are two types of screen events, screen customizations and screen combinations.

**Screen customization**
A HATS screen event designed to perform a set of actions when a host screen is recognized. Examples of screen customizations include recognizing a screen and transforming it into a GUI for the user or playing a macro to skip the screen. The screen customization definition includes a set of screen recognition criteria and a list of actions to be taken when a host screen matches the screen recognition criteria. Screen-level global rules and text replacement settings are also included. For more information see Chapter 7, "Working with screen events," on page 147.

**Screen combination**
A HATS screen event designed to gather output data from consecutive, similar host screens, combine it, and display it in a single output page. An

example of a screen combination includes recognizing a screen that contains only partial data and navigating through all subsequent screens to gather all of the remaining data to display for the user. The screen combination definition includes a set of screen recognition criteria for both the beginning and ending screens to be combined, how to navigate from screen to screen, and the component and widget to use to recognize and render the data gathered from each screen. Also, it includes a list actions to be performed one or more times, screen-level global rules and text replacement settings. For more information see Chapter 7, “Working with screen events,” on page 147.

**Screen recognition criteria**
When you create a screen event, you set screen recognition criteria that HATS uses to match one or more screens. Host screens can be recognized by any combination of criteria including how many fields or input fields are on the screen, the coordinates of the cursor's position, and text strings on the screen within a defined rectangle or anywhere on the screen. You can also use global variables, color, compare region to value and compare region to region criteria for screen recognition criteria.

When a host displays a screen, HATS determines whether the current host screen matches any of the screen recognition criteria you set in any screen event in your project. If HATS finds a match, the defined actions for the screen event are performed.

For more information about setting screen recognition criteria, see “Screen Recognition Criteria or Begin Screen” on page 148.

**Action**
A step that is taken when an event occurs, such as when a host screen is encountered that matches the screen recognition criteria that is specified for a screen event. A list of actions is part of the definition of each application event and screen event.

**Transformation**
A JavaServer Pages (JSP) file (for Web projects) or a Standard Widgets Toolkit (SWT) composite (for rich client projects) that defines how host components should be extracted and displayed using widgets in a GUI. Applying a transformation is one of the possible actions of a screen event. You can edit transformations in a WYSIWYG fashion, dragging HATS components from the palette, and modifying settings from the HATS Properties view. The Window Builder which is installed with HATS can be used to edit rich client transformations in a WYSIWYG fashion.

For more information about creating transformations, see Chapter 8, “Working with transformations,” on page 173.

**Host components**
Host components are HATS objects that are responsible for recognizing elements of the host screen (such as command lines, function keys, and so on) that you present to the user of the HATS application. You can use the set of host components that HATS supplies, or you can create your own.

For more information about selecting host components to use with your HATS project, see “Insert Host Component” on page 178.

For more information about using the wizard to create custom components and widgets, see the *HATS Web Application Programmer's Guide* or the *HATS Rich Client Platform Programmer's Guide*, depending on your application environment.

**Widgets**

Widgets are HATS objects that are responsible for creating the GUI output for host components in the HATS presentation. For example, you can convert function key host components into button widgets allowing the user to see the function keys as push buttons in the HATS application. You can use the widgets that come with HATS, or you can create your own.

For more information about selecting widgets to use with your HATS project, see "Insert Host Component" on page 178.

For more information about using the wizard to create custom components and widgets, see the *HATS Web Application Programmer's Guide* or the *HATS Rich Client Platform Programmer's Guide*, depending on your application environment.

**Connection**

A set of parameters used by HATS to connect to host applications. There are two types of connections in HATS, default (also referred to as transformation) and background. Each HATS application has one default connection for the host application whose screens HATS will transform. Background connections are any connections in a HATS application other than the default connection. HATS does not transform screens from background connections. It is possible, however, to dynamically choose which connection will be treated as the default connection. For more information, see Chapter 6, "Managing connections," on page 131.

**Global rules**

Rules specifying the screen-level or project-level replacement of a certain type of host input field with a particular widget using transformation fragments. The transformation fragment contains the content you want to use to replace all occurrences of a pattern in any given transformation.

**Note:** Global rules can be turned off for a particular usage of a transformation by editing the transformation action of a screen event.

For more information, see "Rendering" on page 90.

**Default rendering**

The method used by HATS to render a host screen for which no specific screen event exists. Default rendering can also be used in a specific transformation to apply to a defined area of a host screen. For more information, see "Rendering" on page 90.

**Text replacement**

Text replacement is used at the project level or the screen level to transform text on your host screens into images, HTML **Web-only**, or other text on your GUI transformation. For more information, see "Text replacement" on page 97.

**Template**

A JSP file (for Web projects) or a Java™ SWT composite class (for rich client projects) that enables you to enhance the appearance of your project. When creating a HATS project, you select a template to use as the default template for your project. The template can contain company logos and information and links to other pages. You can select your default from the sample templates that are provided with HATS, or you can design custom templates for your projects using the template wizard and editor in the HATS Toolkit. You can also create a template prefilled from an existing file, or for Web projects, prefilled from a URL.

In the GUI, the template surrounds the area where the transformation appears. You can apply a template other than the default template as a result of a screen event. For more information about associating a template with a transformation, see Chapter 10, "Using templates," on page 311.

**Business logic**
Any Java code that is invoked as an action in an event, such as a screen customization. Business logic is specific to the application.

For more information about business logic, see the *HATS Web Application Programmer's Guide* or the *HATS Rich Client Platform Programmer's Guide*, depending on your application environment.

**Global variable**
A variable that is used to store a value that can be used throughout the lifetime of a HATS application instance. The value of a global variable can be extracted from a host screen or defined by the developer. Global variables can be used in templates, transformations, screen event actions, or recognition criteria. They can also be shared among Web applications in an enterprise archive (.ear) file or among rich client applications running in the same rich client environment. Global variables can be indexed and can also be used in macros, business logic, and Integration Objects.

For more information about global variables, see Chapter 12, "Interacting with global variables," on page 337.

**Macro**
An XML script that defines a set of screens and certain actions that should be taken on those screens. Use macros to automate user interactions with the host. You can record and play macros to skip screens, loop, prompt users for data input (or insert data yourself or with global variables), and extract host screen information.

Macros can be used in different ways in a HATS application:

- A macro can be played on the default connection as part of a **Play macro** action of a HATS event. It must be the last action that is defined for a screen event.
- A macro can be played on a background connection as part of a **Perform macro transaction** action of a HATS event.
- A macro can be played by adding Macro buttons to a transformation. This enables a user to control whether the macro is played or not.
- A macro can be played by an Integration Object.
- A connect macro can be used to prime a connection.
- A disconnect macro can be used to clean up a connection.

For more information about incorporating macros into the HATS environment, see Chapter 11, "Macros and host terminal," on page 323.

**Integration Objects**
Java objects that are created from a macro that can be used as building blocks for WebSphere applications. Integration Objects are Java beans that encapsulate interactions with host data sources. These data sources include terminal-oriented applications that use 3270, 5250, and video terminal (VT) data.

Integration Objects sharing the same connection can be grouped together to achieve a single major task within your HATS application. This is called Integration Object chaining. For more information, see Chapter 13, "Using Integration Objects," on page 341.

**Host terminal**
A connection in the HATS Toolkit to a live host. Using the host terminal, you can capture screens, create screen events and transformations, and record and edit macros. You can also play previously recorded or imported macros. The host terminal also allows you to preview your screen as a GUI.

**Display terminal**
A terminal window that displays host screens you can use while testing and debugging to observe interactions between a HATS application and a host application at runtime. You can also interact with the host application using host screens in the terminal window. For more information about the display terminal, see "Using display terminal for testing and debugging" on page 384.

**Screen capture**
An XML representation of a host screen, stored in a .hsc file, used to create or customize a screen customization, screen combination, transformation, global rule, or macro. Screen captures are useful because they enable you to develop a HATS project even when not connected to the host. They are also useful in creating macros which are the core of HATS Integration Object and Web services support.

Screen captures of video terminal (VT) host screens can be used to create or customize a macro using the Visual Macro Editor and as the check-in screen when configuring pooling. They cannot be used to create screen customizations, screen combinations, transformations, default rendering, or global rules.

**Run on Server** `Web-only`
A function in Rational SDP that enables you to test your HATS Web projects in a WebSphere Application Server or WebSphere Portal instance, as appropriate. In this mode you can modify and test the runtime settings, defined in the runtime.properties file, that are deployed to the runtime environment. Be aware that any changes made to the runtime settings while testing in this mode are retained and become effective when you deploy the HATS application to a runtime environment. For more information about changing runtime settings, see "Administering problem determination components" on page 381.

**Debug on Server** `Web-only`
Same as Run on Server, and in addition enables you to:

- Use the display terminal to see host screens as they are navigated while testing your project.
- See debug messages in the Rational SDP console.
- See changes you make to your project, for example changing the template or a transformation, without having to restart your application on the test server.
- Modify and test runtime settings, defined in the runtime-debug.properties file, without modifying the settings, defined in the runtime.properties file, that are deployed to the runtime environment.
- Step through Java code, such as HATS business logic.

**Profile on Server** `Web-only`
Same as Run on Server, and in addition enables you to locate the operations that require the most time, and identify actions that are repeated, to eliminate redundancy. You can use this function for performance analysis, helping you to get a better understanding of your application.

**Run** `RCP-only`

A function in Rational SDP that enables you to test your HATS rich client projects in an Eclipse, Lotus Notes, or Lotus Expeditor Client instance. In this mode you can modify and test the runtime settings, defined in the runtime.properties file, that are deployed to the runtime environment. Be aware that any changes made to the runtime settings while testing in this mode are retained and become effective when you deploy the HATS application to a runtime environment. For more information about changing runtime settings, see "Administering HATS rich client applications" on page 76.

**Debug** `RCP-only`

Same as Run, and in addition enables you to:

- Use the display terminal to see host screens as they are navigated while testing your project.
- See debug messages in the Rational SDP console.
- See changes you make to your project, for example changing the template or a transformation, without having to restart your application.
- Modify and test runtime settings, defined in the runtime-debug.properties file, without changing the runtime settings, defined in the runtime.properties file, that are deployed to the runtime environment.
- Step through Java code, such as HATS business logic.

**Profile** `RCP-only`

Same as Run, and in addition enables you to locate the operations that require the most time, and identify actions that are repeated, to eliminate redundancy. You can use this function for performance analysis, helping you to get a better understanding of your application.

**Print support**

The ability for a developer to specify a printer session to be associated with a host session, and enable the user to view host application print jobs, send them to a printer, or save them to disk. Print support is available only for the default connection.

For more information about print support, see Chapter 15, "Enabling print support," on page 353.

**Keyboard support**

The ability for a developer to enable a user to use a physical keyboard to interact with the host when the application is run in a GUI. The developer also decides whether to include a host keypad, an application keypad, or both, in a project. If the keypads are included, the developer decides which keys are included and how those keys and the keypad appear in the GUI.

For more information about keyboard support, see Chapter 16, "Enabling keyboard support," on page 359.

**Keypad support**

The host keypad is a table of buttons or links that enable users to interact with the host as if they pressed the physical keys on a keyboard. However, the users can still use the physical keys on the keyboard instead of the buttons or links on the host keypad. For more information about host keypad, see "Host keypad" on page 185

The application keypad is a table of buttons or links that enable users to perform tasks that are related to the HATS application, such as viewing their print jobs or refreshing the screen. For more information about application keypad, see "Application keypad" on page 320.

## Understanding HATS application processing

Before creating a HATS project, you should understand how HATS processes host applications. As users access each screen of an application, HATS processes the application as described in the following steps. Figure 1 on page 9 shows the flow of these steps. Key concepts are described in "Understanding HATS key concepts and objects" on page 2.

1. When the host displays a screen, HATS compares the host screen to the set of screen recognition criteria that is defined in each of the project's enabled screen events, in the order that is defined by event priority, until a match is found.
2. When a match is found, HATS performs the actions that are defined for the screen event. For both screen customization and screen combination events, these can include the actions listed below. For a screen combination event, in addition, HATS navigates through multiple screens gathering data from each screen to display to the user.
   - Apply a transformation using the associated template. HATS displays any host components (defined in the transformation) as GUI widgets.
   - Execute business logic
   - Extract global variable
   - Insert data
   - Set global variable
   - Remove global variable
   - Send global variable (HATS standard portlet projects only)
   - Show URL or SWT Composite `RCP-only`
   - Show URL `Web-only`
   - Forward to URL `Web-only`
   - Play macro
   - Perform macro transaction
   - Send key
   - Disconnect
   - Pause

   For more information, see "Actions" on page 154.

   **Notes:**
   1. When a screen match is found, at least one action must be performed, but a transformation of the screen is not required to be displayed to the user. In other words, the apply a transformation action is not required.
   2. You can perform actions against the host screen data before and after the user interacts with it. For example, you can extract some data as a global variable, replace some information with new data, and then apply the transformation before showing the transformation to the user. After the user performs an action that returns the screen data, you can perform additional extractions and insertions before the data is finally returned to the host.

3. If no screen recognition criteria match the host screen, HATS processes the unmatched screen application event. The default action of this event is to display the host screen using the default transformation and applying the default template. The default transformation uses the rendering settings that are defined in the project settings.

4. As the host presents each new screen of an application, HATS begins at step 1 again and proceeds through all steps. Figure 1 shows how HATS processes screens.

   **Note:** If a macro is used for skip-screen processing, those screens are not subject to this process.

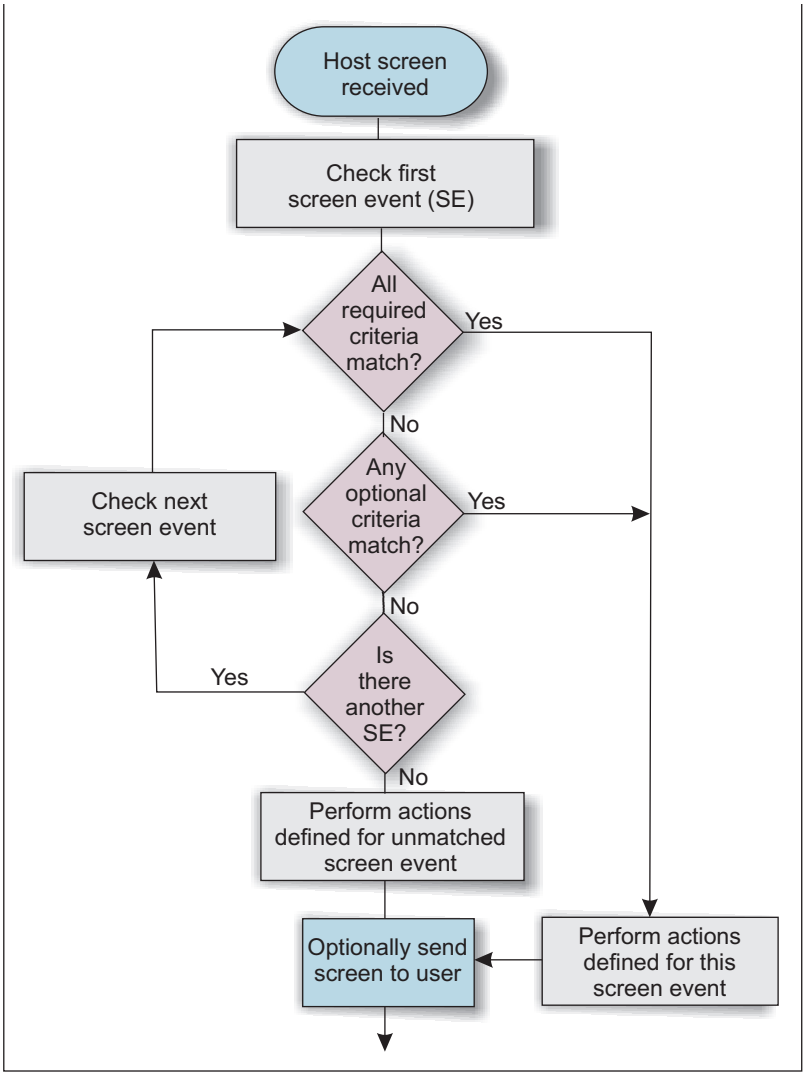



*Figure 1. HATS screen processing*

Screen events are an important concept in the development of a HATS project. Without defined screen recognition criteria in a screen event, HATS does not know what actions to take when the host screen is received.

You should familiarize yourself with the basic principles of screen events before beginning the development process. The core elements of setting screen recognition criteria are discussed in Chapter 7, “Working with screen events,” on page 147.

## Understanding HATS application development

HATS enables you to use an iterative approach to application development. You can start with a simple configuration and add refinements as you are ready. You can test each change as you make it using the test environment of Rational SDP. Most changes can be seen by a simple refresh of the HATS transformation. You can make modifications if needed and save your work before you proceed to the next change. This section describes one possible sequence of steps that you can use to design and develop your HATS application. You can go as far down this path as you like and stop at any point along the way.

1. Begin by creating a HATS application that uses all of the defaults and instantly see all of your host screens rendered in a GUI. Default rendering can render your host screens in a way that preserves the original layout of the screen and automatically adds GUI controls, such as links, buttons, and tables, to improve user productivity.
2. Create a template that matches your company's Web pages, or other company applications, and see all of your host screens rendered with your company appearance. You can start with, and modify, one of the templates supplied with HATS, or for HATS Web applications you can import your company's web site as a template. See Chapter 10, “Using templates,” on page 311 for more information. Now your HATS application matches the style and colors of your company's Web pages or other corporate applications.
3. Configure project-level (application-wide) settings and see how much of your host application can be handled without configuring specific screens. You can change project-level characteristics under the various tabs of the Project Settings editor. See Chapter 5, “Modifying a HATS project,” on page 87 for more information. For example, try making a few changes such as:
   - Change default rendering to render a selection list as a drop-down list. This will change all selection lists to drop-down lists.
   - Create one or more global rules to configure the way HATS transforms the input fields on your host screens. For example, change date fields to calendar widgets or location fields to drop-down lists.
   - Use text replacement to change one or more strings that appear on your host screens. For example, you might change the word `signon` to `logon`.
   - Look at the way your host screens are transformed. Are there elements that appear on most screens that you want to see transformed differently? Those might be candidates for modifying the settings of some components (to change how host components are recognized) or widgets (to change how the GUI controls are rendered on the transformation). You can change these settings at the project level to establish defaults to be used on your host screens. You can also change the settings for individual instances of components and widgets when you add them to transformations. For more information see “Rendering” on page 90.
4. Create screen-level events to recognize specific host screens and trigger a set of actions to be performed.
   - If you use BMS maps, you can start by importing your BMS map sets to create screen captures for your host screens. Otherwise, use the HATS host terminal function to connect to the host application and create screen captures for the specific screens you want to recognize and act upon. Then from each screen capture, create a screen event.
   - Use the Create a Screen Customization wizard to customize some of the host screens to simplify them, remove unused items, and automatically extract and populate data to simplify user input. Provide screen-specific GUI

controls for a higher degree of productivity for the user. For more information see Chapter 7, “Working with screen events,” on page 147.

- It is a common requirement to be able to gather data from multiple host screens and present it in a single output page, or to provide a single input page, which then supplies multiple host screens with data. HATS provides several methods for performing these operations. The different methods include the use of screen combinations, screen customizations, transformations, macros, global variables, and Integration Objects. Which method you use depends upon the operation to be performed and the location of the data. For starters, try using the Create a Screen Combination wizard to combine data that does not all fit on a single host screen and display it in a single output page. For more information, see Chapter 14, “Combining screens,” on page 351 and Chapter 7, “Working with screen events,” on page 147.

5. Create simple macros to aid the user in navigating the host application. Macros can be used to perform skip-screen functions, to prompt users for input data, or to extract data from one or more host screens. For more information, see Chapter 11, “Macros and host terminal,” on page 323.
6. For Web applications, create Integration Objects, which are built from macros and are Java beans that encapsulate interactions with a host application. Then use the Integration Objects to create Model 1, Struts, or JavaServer Faces (JSF) Web pages to drive interactions with your host application. You can also run Integration Objects from your own Java business logic or extend them to provide Enterprise JavaBeans (EJB) or Web services interfaces to your host application. For more information, see Chapter 13, “Using Integration Objects,” on page 341.

## How do I know what version of HATS I have?

You can determine the version of your HATS Toolkit and of each of your HATS projects.

To determine what version of HATS Toolkit you are using, check the IBM Installation Manager Installed Packages:

1. From the Start menu, select **All Programs > IBM Installation Manager > View Installed Packages**.
2. The IBM Installation Manager - Installed Offerings page lists all of the installed packages (including version and build levels) and features.

You can determine the version (including maintenance level) of each HATS project. This version information is maintained in each project in a file named .serviceHistory.xml. This file can be viewed from the Navigator view of the HATS perspective in the Web Content\WEB-INF\profiles folder (for Web projects), the profiles folder (for rich client projects), and the ejbModule folder (for EJB projects).

The version information for HATS .ear projects and for the HATS RCP runtime extension project is maintained in the product.xml file. This file can be viewed from the Navigator view in the root folder for the project.

## Gathering problem determination information

Sometimes you might require assistance in determining HATS related problems. For instructions on how to gather problem determination information for HATS support personnel see the section, Gathering problem determination information, in *HATS Troubleshooting*.

## Where can I find information about HATS?

The following HATS information is available before you install HATS:

- HATS documentation is available on the HATS installation CD. From the installation launchpad, select links to the following documents:
  - *Release Notes*
  - *Getting Started (pdf)*
- The HATS Knowledge Center at http://www.ibm.com/support/knowledgecenter/SSXKAY_9.7.0 includes information from the Release Notes, frequently asked questions (FAQs), API reference information, tutorials, WebFacing information, and HTML and PDF versions of the documents listed below. Available translated PDF versions are also included.
  - *HATS Getting Started*
  - *HATS User's and Administrator's Guide*
  - *HATS Web Application Programmer's Guide*
  - *HATS Rich Client Platform Programmer's Guide*
  - *HATS Advanced Macro Guide*
  - *HATS Messages*
  - *HATS Troubleshooting*
  - *Developing WebFacing Applications*
- The HATS product Web site at http://www.ibm.com/software/products/us/en/rhats includes additional product information such as feature overview, trial download, system requirements, and more.
- The HATS HotSpot at https://www.ibm.com/developerworks/mydeveloperworks/groups/service/html/communityview?communityUuid=2ce1fd8d-d706-4afd-b9ef-9000ad21218d includes a resource library, discussion forums, blogs, and more.
- The IBM Education Assistant Web site at http://publib.boulder.ibm.com/infocenter/ieduasst/rtnv1r0/index.jsp integrates narrated presentations, demonstrations, tutorials, and resource links to help you successfully use IBM products. Search for Host Access Transformation Services in the Contents.

In addition to the information available before installing HATS, the following information is available on the system after installing HATS:

- Links to the HATS Web site, the Knowledge Center, and *Release Notes* are available on the Windows Start menu.
- HATS documentation, including WebFacing, is available from the Help menu on the Rational SDP menu bar. Select **Help > Help Contents** and in the Contents panel select **Developing HATS Applications**.
- The HATS Welcome page tutorial guides you through the process of developing a HATS application, and if installed on the IBM Rational Developer for Power Systems Software™ product, introduces the IBM WebFacing Tool for IBM i feature of the HATS Toolkit.
- If the WebFacing Tool is installed, the WebFacing Welcome page tutorial provides information similar to the HATS Welcome page tutorial.
- Context-sensitive help is available on all fields in the HATS wizards and editors. Press the F1 key to see help in the HATS Toolkit.
- Tips are provided at key points in the process of developing a HATS project. You can control whether you want to see tips by modifying HATS preferences.

# Chapter 2. Migrating to HATS V9.7

If you are a Host Publisher V4, HATS V5, V6, V7.0, V7.1, V7.5, V8.0, V8.5, or V9.0 user, you can migrate your projects to HATS V9.7.

HATS V4 LE, HATS V5 LE, and HATS V4 projects cannot be migrated directly to HATS V8.0, V8.5, V9.0, or V9.7. To migrate these projects you must first migrate them to an interim release of HATS, for example, V5, V6, V7, V7.1, or V7.5, and then migrate them from the interim release to HATS V8.0, V8.5, V9.0, or V9.7. See the documentation for your previous release of HATS for information about migrating HATS projects.

If you are using a Host Publisher version before V4.0, you must first follow the instructions for migrating your Host Publisher applications to Host Publisher V4.0. Go to http://www.ibm.com/support/entry/portal/overview/software/rational/rational_host_access_transformation_services and enter the search string "How to migrate Host Publisher". For more information, see “HATS for Host Publisher users” on page 22.

## HATS migration

Migrating a HATS project from a previous release of HATS is a two step process.

1. Import the project into your HATS V9.7 workspace using one of the following methods:
   - Import an existing project selecting either its root directory or an archive file.
   - Check the project out from a code repository (such as CVS or IBM Rational ClearCase®).
   - Open a Rational SDP V7.0, or later, workspace containing the project, in Rational SDP V9.7.
2. Migrate the project using the HATS migration wizard.

### Importing a HATS project

If you have a HATS V5, or later, release of HATS installed, you can export your project to either a zip file or a project interchange file and then import and migrate it to HATS V9.7. See the documentation for your previous release of HATS for information about exporting HATS projects. For more information about importing HATS Web and rich client projects into HATS V9.7, see “Exporting and importing HATS Web projects” on page 30 and “Exporting and importing HATS rich client projects” on page 60.

If you have stored your HATS projects in Rational ClearCase, you should create a snapshot view containing these projects and then import them into your HATS V9.7 workspace using **File > Import > General > Existing Projects into Workspace**. During the import, do not select the **Copy projects into workspace** option. After the projects are imported into your HATS V9.7 workspace, you have the option to migrate them to HATS V9.7. During migration, files that need updating are automatically checked out of Rational ClearCase.

#### Rational SDP workspace migration

The Rational SDP Workspace Migration wizard displays when a project is imported into the workspace and needs some level of Rational SDP migration. This

wizard performs some basic Rational SDP project migration and if the project targets an unsupported runtime, forces you to choose a supported target runtime. In preparation for running the HATS Migration wizard, you should first run the Rational SDP Workspace Migration wizard.

**Note:** Rational SDP V9.7 workspace migration only supports projects created from Rational SDP V7.0, or later. When importing a pre-HATS V7.0 project, Rational SDP workstation migration fails and an error message is displayed. HATS migration will still be able to migrate the project to HATS V9.7.

## Using the HATS Migration wizard

When you switch to the HATS perspective, if you have a project created from a previous HATS release in your HATS V9.7 workspace, you will get a message stating that the workspace contains a HATS project that needs to be migrated.

You should use the HATS Migration wizard to migrate this project to HATS V9.7 before investigating any errors shown in the **Problems** view because migration may eliminate some or all of these errors.

To use the HATS Migration wizard follow these steps:

1. In the **HATS Projects** view, right-click on the project and select **Migrate Project**.
2. If you have other projects from a previous HATS release that are associated with the HATS project you have selected to migrate, those projects will also be migrated, as well as any associated HATS .ear projects.
3. Click **OK** to begin the migration.
4. If the project does not have an associated .ear project, you must associate the .war project after the migration. To associate the .war project with an .ear project, see "Moving HATS Web projects to a different .ear file" on page 29.

**Notes:**

1. After migration check the details on the Migration Report.
2. If a HATS Web project being migrated has an unsupported target runtime (if the Rational SDP Workspace Migration wizard has not been run) or unsupported WebSphere facets, the HATS Migration wizard sets the target runtime and facets to the lowest supported WebSphere Application Sever level for Web projects. A message is displayed and logged stating the fact that the change was made.
3. If, after migration, you receive an error indicating that the target runtime is not defined, edit the properties for the project and select the appropriate target runtime. To do this, right-click on the project and select **Properties**. From the Properties window select **Targeted Runtimes**. From the Runtimes list, select the appropriate runtime.
4. Migrating your project from a previous HATS release to HATS V9.7 cannot be undone.

## Migrating HATS transformations

When you migrate HATS projects from a previous release of HATS to HATS V9.7, transformations in the projects are migrated as well. Changes made to the transformations depend on the release of HATS you are migrating from. All transformations are saved in the MigrationBackup folder before being migrated.

# Special migration considerations

## Auto advance

If in previous versions of HATS you enabled the auto advance function by adding the line `turnAutoTabOn();` to the lxgwfunctions.js file, you must now use the **Enable automatic field advance** setting in the Client Settings section of the **Other** tab in the Project Settings editor. For more information, see "Client settings" on page 116.

## Backup files

HATS migration creates a backup folder in the project directory called **MigrationBackup**. This folder contains copies of the files from the old project before they were overwritten by the migration process. These files are saved allowing you to compare and merge them with your new HATS V9.7 files. Do not worry if problems occur; these files are no longer used by the application. When you are satisfied that all the saved files have been compared and merged, you can delete the **MigrationBackup** folder.

**Note:** Before you delete the MigrationBackup folder, you can publish a migrated project to WebSphere Application Server to test how it behaves. However, errors in the task list caused by the MigrationBackup folder may create problems when publishing to WebSphere Application Server. To avoid this, you can either remove the MigrationBackup folder from the project or allow applications containing errors to be published on a server. To allow applications containing errors to be published on a server, select **Window > Preferences > Server > WebSphere Application Server** from the Rational SDP menu bar and select **Allow applications containing errors to be published on a server**.

## Button widget

If you migrate a project created prior to HATS V7.0.0.2, and you choose to implement the **Enable foreground colors** option, if you use a template that uses the blacktheme.css style sheet, you must manually update the blacktheme.css style sheet in one of the following ways:

1. Remove the following line from the input.HATSBUTTON declaration:

   ```
   color: lime;
   ```

   This enables foreground colors to be rendered on function keys; however it causes other buttons generated by a HATS widget to render in a default color.
2. Combine the color-related CSS declarations for each color. For example, change:

   ```
   .HBLUE {
          color: #3c9dff;
   }
   input.HBLUE {
          white-space: normal;
          letter-spacing: normal;
   }
   ```

   to

   ```
   .HBLUE, input.HBLUE {
           color: #3c9dff;
           white-space: normal;
           letter-spacing: normal;
   }
   ```

   Repeat this change for each color.

An alternative to manually editing the blacktheme.css style sheet is to create a new dummy project and copy the theme CSS files from this project into your project. Be aware that any changes you have made to your CSS files are overwritten.

## EJB access beans

If you have existing HATS V5 projects that have EJB Access Beans, you may have compile or runtime errors that begin with "HPubReqCompleteEvent cannot be resolved." To correct this, regenerate the EJB Access Beans.

**Note:** HATS EJB application support is deprecated in HATS V9.7. While support for HATS EJB applications continues for now, IBM reserves the right to remove this capability in a subsequent release of the product. Alternatives are:

- Use Web services to access your Integration Objects. For more information see Developing Web services, in the *HATS Web Application Programmer's Guide*.
- Create custom EJB beans to access your Integration Objects. For more information see Using an Integration Object in an EJB container (from your own EJB), in the *HATS Web Application Programmer's Guide*.

## Field widget

If you want to use the **Render using monospace font** setting for the Field widget, and your project was originally created in a release earlier than HATS V7.0, you must update your CSS files as follows:

1. Add the following class to the whitetheme.css, graytheme.css, monochrometheme.css, tantheme.css, and blacktheme.css files:

```
.HF {
  font-family: courier new, monospace;
}
```

2. Remove `font-family: monospace` from all of the `H`-color (HATS color) classes of all CSS files.

## Global rules

A new setting, **enforceImmediacy**, is added to any global rule imported into HATS V9.7 from HATS V5. When this setting is true, the global rule behaves as it did in HATS V5 when **Nearest input field only** is selected for the **Transform** option of the **Find input fields by surrounding text** pattern. When this setting is false, the default, the global rule behaves as it does in HATS V9.7.

To make newly defined global rules behave as the HATS V5 global rules did, you must add and set the **enforceImmediacy** setting to true. To do this, on the **Source** tab of the Project Settings editor, add the following setting to the globalRules componentSettings tag:

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

## Global variables

Before HATS V7, the value of a global variable, entered into a prompt inserted onto a transformation using the **Prompt for global variable with input box** option, might have been truncated because some characters were not escaped properly.

In this HATS version, global variable values prompted for by a transformation are properly escaped to prevent truncation. Global variables inserted onto a transformation using the **Display global variable value as static text** option are not affected by this change. Global variable prompts inserted onto a transformation

in a version of HATS before HATS V7 are not automatically updated during migration, so they are not affected by this change.

## HTTP compression

If you want to use HATS support for HTTP compression in projects migrated from HATS V5.0.x, V6.0, and V6.0.1, you must manually add the compression filter to the Web Deployment Descriptor file (web.xml). To add the compression filter to the web.xml file:

1. From the HATS Toolkit, switch to the **Navigator** view of the HATS perspective.
2. Open the web.xml file located in the Web Content\WEB-INF folder of your project.
3. Click the **Source** tab to view the source of this file.
4. Copy the following statements after the last defined servlet mapping (search for the last `</servlet-mapping>`).

```
<filter>
   <description>Provides compression for output from HATS entry
                servlet</description>
   <display-name>CompressionFilter</display-name>
   <filter-name>CompressionFilter</filter-name>
   <filter-class>com.ibm.hats.runtime.filters.CompressionFilter</filter-class>
</filter>
```

5. Copy the following statements after the last defined filter mapping (search for the last `</filter-mapping>`).

```
<filter-mapping>
   <filter-name>CompressionFilter</filter-name>
   <servlet-name>EntryServlet</servlet-name>
   <dispatcher>ERROR</dispatcher>
   <dispatcher>FORWARD</dispatcher>
   <dispatcher>INCLUDE</dispatcher>
   <dispatcher>REQUEST</dispatcher>
</filter-mapping>
<filter-mapping>
   <filter-name>CompressionFilter</filter-name>
   <url-pattern>/</url-pattern>
</filter-mapping>
<filter-mapping>
   <filter-name>CompressionFilter</filter-name>
   <url-pattern>/index.jsp</url-pattern>
</filter-mapping>
```

6. Save the file.

**Note:** If you have already run this project on the server, you will need to republish the application for WebSphere Application Server to pick up the change to the web.xml file.

## Enable CSRF Protection

In order to use HATS support for CSRF protection in projects migrated from HATS V8.0.x, V9.0.x, manually add the HatsCSRFValidationFilter, to the Web Deployment Descriptor file (web.xml). To add the filter to the web.xml file, follow the below steps:

1. From the HATS Toolkit, switch to the **Navigator** view of the HATS perspective.
2. Open the web.xml file located in the Web Content\WEB-INF folder of your project.
3. Click the **Source** tab to view the source of this file.
4. Copy the following statements after the last defined filter (search for the last `</filter>`).

```
<filter>
   <description>
        This filter will be invoked to validate CSRF attack
  </description>
          <display-name>HatsCSRFValidationFilter</display-name>
           <filter-name>HatsCSRFValidationFilter</filter-name>
   <filter-class>com.ibm.hats.runtime.filters.HatsCSRFValidationFilter</filter-class>
        <async-supported>false</async-supported>
        <init-param>
          <param-name>source.origin</param-name>
          <param-value></param-value>
        </init-param>
</filter>
```

5. Copy the following statements after the last defined filter mapping (search for the last `</filter-mapping>`).

```
<filter-mapping>
   <filter-name>HatsCSRFValidationFilter</filter-name>
    <url-pattern>/entry</url-pattern>
</filter-mapping>
<filter-mapping>
   <filter-name>HatsCSRFValidationFilter</filter-name>
    <url-pattern>/hatsadmin/admin</url-pattern>
</filter-mapping>
<filter-mapping>
   <filter-name>HatsCSRFValidationFilter</filter-name>
    <url-pattern>/index.jsp</url-pattern>
</filter-mapping>
<filter-mapping>
   <filter-name>HatsCSRFValidationFilter</filter-name>
    <url-pattern>/</url-pattern>
</filter-mapping>
```

6. Save the file.

**Note:** If this project is already running on the server, republish the application so that the WebSphere Application Server can pick up the changes in the web.xml file.

## Enable XSS Protection

In order to use HATS support for XSS protection in projects migrated from HATS V8.0.x, V9.0.x , manually add the HatsHeaderSecurityFilter, to the Web Deployment Descriptor file (web.xml). To add the filter to the web.xml file, follow the below steps:

1. From the HATS Toolkit, switch to the **Navigator** view of the HATS perspective.
2. Open the web.xml file located in the Web Content\WEB-INF folder of your project.
3. Click the **Source** tab to view the source of this file.
4. Copy the following statements after the last defined filter (search for the last `</filter>`).

```
<filter>
   <description>
        This filter will be invoked to create the security header for HATS
  </description>
          <display-name>HatsHeaderSecurityFilter</display-name>
           <filter-name>HatsHeaderSecurityFilter</filter-name>
   <filter-class>com.ibm.hats.runtime.filters.HatsHeaderSecurityFilter</filter-class>
        <async-supported>false</async-supported>
        <init-param>
          <param-name>Content-Security-Policy</param-name>
          <param-value>NO</param-value>
        </init-param>
        <init-param>
```

```
            <param-name>X-XSS-Protection</param-name>
            <param-value>NO</param-value>
          </init-param>
          <init-param>
            <param-name>X-Content-Type-Options</param-name>
            <param-value>NO</param-value>
          </init-param>
</filter>
```

5. Copy the following statements after the last defined filter mapping (search for the last `</filter-mapping>`).

```
<filter-mapping>
       <filter-name>HatsHeaderSecurityFilter</filter-name>
       <url-pattern>
     </url-pattern>
</filter-mapping>
```

   The User has to provide the URL-pattern value, which will be secured from XSS attack.

6. Save the file.

**Note:** If this project is already running on the server, republish the application so that the WebSphere Application Server can pick up the changes in the web.xml file.

## IBM license management tools

HATS, WebFacing, and HATS/WebFacing linked applications that are migrated to HATS V9.7 must be redeployed to the production environment to pick up support for the new HATS V9.7 signature file by the IBM license management tools, for example, IBM License Metric Tool and IBM Tivoli® Asset Discovery for Distributed.

**WebFacing Web applications:** IBM license management tools detect Web applications deployed as Enterprise Archives (EARs) to supported WebSphere Application Servers. Since WebFacing Web projects are created independently of associated EARs, the HATS V9.7 signature file must be included manually. Follow the steps below to enable the proper detection of WebFacing Web applications.

1. Create a folder named `itlm` directly under your WebFacing project's associated EAR.
2. Locate the signature file named `Host_Access_Transformation_Services-9.0.0.swidtag` in the HATS plugin directory

   `<`*`shared_install_directory`*`>\plugins\com.ibm.hats_`*`nnn`*`\`

   where *shared_install_directory* is the shared resources directory where you installed the HATS offering using IBM Installation Manager and *nnn* is the version and build level of HATS.
3. Copy the signature file from the plugin to the `itlm` folder you created earlier.
4. Export your project as an EAR and redeploy.
5. The IBM license management tools will now be able to detect your application.

If you enabled detection for an old WebFacing project and migrate that project to V9.7, you must remove the existing signature file (for example, `WDHT0700.sys2` if the old WebFacing project is from V7.0) from the project's associated EAR and add the V9.7 signature file (`Host_Access_Transformation_Services-9.0.0.swidtag`) before redeployment.

## Java 2 security

During migration to HATS V9.7, the WebSphere Application Server Java 2 security was.policy file is overwritten. If you have customized the was.policy file in a pre-HATS V9.7 project, then you must re-customize the file after migration.

## Keyboard mappings

In earlier versions of the HATS host terminal, the Pause key was mapped to the host's [clear] action. In HATS V7, and later, host terminal, the Esc key is mapped to the host's [clear] action.

In versions of HATS earlier than HATS V6, Ctrl+R was used in bidirectional sessions to reverse the screen. In HATS V6 and later, Ctrl+R is mapped to a host RESET for both bidirectional and non-bidirectional sessions, and Alt+Enter is mapped to reverse for bidirectional sessions.

## Rich client applications

**Lotus Notes keyboard bindings:**

If you migrate to HATS V9.7 a pre-HATS V7.5.1 rich client project that is targeted for Lotus Expeditor, the keyboard mappings for Lotus Notes are added to the project.

**Launch configurations:**

After migrating a rich client project, if you are unable to launch it in the test environment, create a new launch configuration to use in the test environment. To do this, in the HATS Projects view, right-click the project and select either **Run** or **Debug**. The Run (or Debug) Configurations window is displayed. If the project is targeted for Eclipse RCP, right-click on **Eclipse Application** and select **New**. If the project is targeted for Lotus Notes or Lotus Expeditor , right-click on **Client Services** and select **New**. Modify the launch configuration name and location if desired. You can delete old launch configurations from this window as well. Click **Run** (or **Debug**) to launch the project in the test environment.

**Themes:**

HATS V7.0 rich client applications that use the **Classic terminal emulator** theme will have the **Arrow key navigation** setting enabled after migration.

**Non-US English strings in templates and transformations:**

Non-US English strings included in templates and transformations of a HATS rich client project built with HATS V7.0 or V7.0.0.1 are not compiled correctly when the project is exported as a feature. To work around this problem, after migrating the project to HATS V9.7, edit the build.properties file located in the root folder of the project. Add the following line at the bottom of the build.properties file:

```
javacCustomEncodings.library.jar = src/rcpproject/templates/[UTF-8],
 src/rcpproject/transformations/[UTF-8]
```

Where *rcpproject* is the name of the rich client project.

## Runtime enablement

To fully enable the HATS V9.7 runtime for production, you must specify your license settings using the License Settings wizard. You must do this even for

projects whose runtimes were fully enabled in previous versions of HATS. For information about specifying license settings, see Enabling HATS runtime and license settings in *HATS Getting Started*

**Note:** HATS Web applications whose runtimes are not fully enabled are restricted to just two runtime host connections. HATS rich client applications whose runtimes are not fully enabled allow unlimited host connections in the local test environment but no host connections in a deployed production environment.

## Secure Sockets Layer (SSL)

During migration, HATS V9.7 converts SSL certificate class files, `CustomizedCAs.class` files, to PKCS12 keystore files with the names *appname*`-CustomizedCAs.p12` and passwords of `hats`, where *appname* is the project name. After migration, you should use the Certificate Management tool (also known as IBM Key Management or iKeyMan) to change the passwords for the new keystore files and verify them using the connection editor. For more information, see "Security" on page 139.

## Testing modes

Starting with HATS V7, the Run on Server (for Web projects) and Run (for rich client projects) testing modes can be used to modify and test the runtime settings that are deployed to the production environment. Be aware that any changes made to the runtime settings while testing in this mode are retained and become effective when you deploy the HATS application to a production environment.

The Debug on Server (for Web projects) and Debug (for rich client projects) testing modes can be used to modify and test the runtime settings without modifying the settings that are deployed to the production environment.

For more information about changing runtime settings for Web projects, see "Starting the administrative console while in HATS Toolkit" on page 379. For more information about changing runtime settings for rich client projects, see "Administering HATS rich client applications" on page 76.

## URL host component

Before HATS V7, the URL host component searched within both input and protected fields. Starting with HATS V7, the URL host component searches only within protected, non-hidden fields.

## Web Express Logon (WEL)

When a HATS Web project is migrated to HATS V9.7 with no associated EAR, Web Express Logon (WEL) does not function properly. The WEL configuration information for a Web project is kept in its associated EAR project. If the associated EAR project is not imported and migrated then this information is lost.

If you import the Web and EAR projects as archive files into the workspace and migrate the Web project, the associated EAR project is also migrated and the WEL configuration information is not lost.

Starting with HATS V7, Java Secure Socket Extension (JSSE) is used by both the HATS DCAS/RACF/JDBC and certificate-based DCAS/RACF credential mapper plug-ins for secure connections to the DCAS server. As a result, consider the following changes to the plug-in initialization parameters.

**CMPI_DCAS_KEYRING_FILE**
This parameter is deprecated and should not be used. However, if used, it

is supported in HATS V9.7 in conjunction with deprecated parameter, CMPI_DCAS_KEYRING_PASSWORD, and with the assumption that the keyring type is pkcs12. Use CMPI_DCAS_TRUSTSTORE instead. For more information see “Initialization parameters” on page 412.

This parameter specifies a keyring database. A keyring must be specified to provide access to the DCAS client certificate as well as the DCAS server's certificate. The certificates establish a client authenticated secure connection with the DCAS server. This parameter is a file reference to the keyring to be used. The DCAS plug-in is the DCAS client. The keyring file must be stored in the .ear file.

**CMPI_DCAS_KEYRING_PASSWORD**

This parameter is deprecated and should not be used. However if used, it is supported in HATS V9.7 in conjunction with deprecated parameter, CMPI_DCAS_KEYRING_FILE, and with the assumption that the keyring type is pkcs12. Use CMPI_DCAS_TRUSTSTORE_PASSWORD instead. For more information see “Initialization parameters” on page 412.

This parameter specifies the password for the keyring database.

## HATS for Host Publisher users

If you are an experienced user of IBM WebSphere Host Publisher, you will need to adjust your approach in order to develop projects with HATS. Here are some key differences between the HATS and Host Publisher approaches to developing projects:

- There is no HATS server install. The HATS executable code, known as the *HATS runtime*, is embedded in each HATS .ear when it is exported in HATS Toolkit. HATS projects also allow you to have multiple .war files within an .ear file.
- Host Publisher does not transform host screens or present them to the user; rather it retrieves specific information from host applications and displays that information. While you can create a HATS project that performs this function, the basic function of HATS is to transform host screens and present them to the user. Of course, you can enhance your HATS project to combine or skip screens, and combine data from multiple host applications. HATS enables you to work from the ground up, and makes it easy to add to your project over time.
- The functions of your Host Publisher project were configured explicitly; you controlled the behavior of your users by specifying the host interactions that they can perform. In the rules-based approach used by HATS, you configure rules by which your HATS project transforms whatever host screens the user accesses, and processes the user's interactions with those screens. Where your Host Publisher project enabled the user to perform a limited set of host interactions, your HATS project applies its rules to handle any sequence of interactions your user performs. HATS enables you to define how freely the user can interact with the host; interaction can range from a free-form host emulator experience to a completely dictated screen navigation.
- In Host Publisher, all host interactions are performed within macros that are encapsulated into Integration Objects. In HATS, you can use macros for many purposes, but they are not required for the basic transformation of host screens. In Host Publisher, an Integration Object requires a connect macro and a disconnect macro, as well as the data macro that is used to extract data from the host application. In HATS, a macro can be run on a host to which the HATS application is already connected. Thus in HATS, Integration Objects do not always require connect and disconnect macros.

**Note:** If you created Host Publisher Integration Objects using a modified template, you will need to re-create the Integration Objects by making similar modifications to the HATS Integration Object template, and then regenerating the Integration Objects from macros. For more information about using templates to customize Integration Objects, refer to the *HATS Web Application Programmer's Guide*.

If you are importing a Host Publisher V4.0 application (versus a Host Publisher V4.0.1 application):

- Remote Integration Object (RIO) support has been removed. All Host Publisher V4 applications contained the RIO servlet. If you import a Host Publisher V4 application that contains RIO support, the RIO servlet is removed, and you will receive a migration message that this has been done. If you used RIO to access Integration Objects remotely, you need to modify those applications to use Web services. Refer to the *HATS Web Application Programmer's Guide* for more information.
- If you try to import Host Publisher applications that contain EJB 1.0 support, you receive an error message that indicates that the application contains EJB 1.0 support. If you want to import the application, you must go back to Host Publisher Studio and regenerate your EJB Access Beans using the EJB 1.1 specification level, and repackage the application using Host Publisher Application Integrator.

## Migrating from Host Publisher Version 4

If you are currently a Host Publisher V4 user, you can migrate whole projects or just the Integration Objects from your existing projects. When you import Host Publisher .ear files, the Integration Objects remain packaged in the .jar file.

If the Integration Object was in a Web Module, the .jar file is in the WEB-INF/LIB directory. If the Integration Object is in an EJB module, the .jar file is in the imported_classes\IntegrationObject directory. The EJB Access Beans also remain packaged in the .jar file in the WEB-INF/LIB directory of the Web Module.

If you import an .ear and modify the macro for which an Integration Object has been created, you will have the original Integration Object .jar that was imported, as well as the Java source. After you have regenerated the Integration Object for HATS, you can delete the imported .jar file.

Below is a list of considerations when migrating from Host Publisher V4:

- If you import a Host Publisher EJB application, an error will show up when you try to run the application. You receive a java.lang.NoClassDefFoundError for com/ibm/HostPublisher/IntegrationObject/HPubReqCompleteEvent. To correct this, regenerate the EJB Access Beans and delete the EJB Access Bean .jar files in the WEB-INF\LIB directory of your Web application. In addition, to fix compile errors for name mismatches, you must update the EJB Access Bean names to reference newly generated names in any JavaServer Pages or Java code .
- If you have Integration Objects or EJB Access Beans with Web service support, you need to regenerate the Web services. For information about regenerating Web services, refer to the *HATS Web Application Programmer's Guide*.
- Database Access Integration Objects are deprecated in HATS V6 or later. If you import a Host Publisher application that contains Database Access Integration Objects, the Integration Objects are preserved in a .jar file. However, you cannot create new Database Access Integration Objects in HATS V6 or later. You can use the relational database tools in Rational SDP to access your relational databases.

- Defined XML Gateway connections are not migrated. You can define HATS applications using the default transformation to achieve the same function as the XML Gateway connections.
- The XML Legacy Gateway SDK is not available in HATS.
- Host Publisher Express Logon is not supported in HATS. Integration Objects that were configured to use Express Logon must be configured to use HATS Web Express Logon (WEL). If you import an Integration Object that contains Express Logon Feature (ELF) support, the Integration Object is migrated without error, but a warning is issued that ELF is not supported and that WEL must be used to achieve the equivalent function. You must re-record your connect macros to use WEL actions, associate the Integration Object connections with the new WEL macros, and also do the WEL configuration. See "Using Web Express Logon (WEL)" on page 406 for more information. The Integration Objects will not perform as intended unless this is done.
- Encrypted user lists are now supported in HATS. However, encrypting user lists with a user-chosen encryption key is not supported in HATS. Importing a Host Publisher user list that was encrypted with a user-chosen encryption key requires supplying the key that was used to encrypt the user list. The user list is decrypted during import. After import you may encrypt the user list for use with HATS. To do this, open the connection editor, click the **User List** tab, and select **Encrypt user list properties**. Only the User ID, Description, and Password fields can be edited on the User List tab. If you must edit any user-defined properties you added to your Host Publisher user list, you must first clear **Encrypt user list properties** on the User List tab, and then click the **Source** tab to edit the property values. Do not edit the schema element. If you wish to encrypt the user list again, click the **User List** tab, select **Encrypt user list properties**, and save the file. Regardless of whether you choose to encrypt your user list data, you should use file system security and physical security measures to protect any sensitive data.
- If you have Integration Objects that were created using a modified template, the following sequence should be used for migrating the Host Publisher application that contains these Integration Objects:
  1. Import the Host Publisher EAR. You will receive warnings that you are importing Integration Objects that contain a modified template.
  2. If any of the Integration Objects that you imported contained data that was extracted as a table, import those individual Integration Objects.
  3. Modify the Integration Object templates that are provided with HATS to achieve the same customizations you defined in Host Publisher. Refer to the *HATS Web Application Programmer's Guide* for more information

## Import Host Publisher EAR wizard

You can import Host Publisher V4 projects into HATS as new HATS projects, or as part of an existing HATS project.

1. Click **File > Import > HATS > Host Publisher V4 EAR** to open the Import Host Publisher EAR file wizard.
2. Specify the EAR file you want to import in the **EAR file name** field.
3. Decide whether the EAR file is to be used for a new HATS project or an existing HATS project.
4. In the **Options** section, you can set **Overwrite resources without warnings** by selecting the check box.
5. If you are using encryption in Host Publisher, you can enter your encryption key in the **User list encryption key** text box.
6. Click **Finish**.

### Import Host Publisher Integration Object

You can also import individual Host Publisher Integration Objects into an existing HATS project.

1. Click **File > Import > HATS > Host Publisher V4 Integration Object** to open the Import Host Publisher Integration Object wizard.
2. Specify the location of the Integration Objects you want to import in the **Host Publisher Integration Objects directory** field.
3. Select which Integration Objects you want to import in the **Select Integration Object(s) to import** field.
4. Select the HATS project in the **Select HATS project** drop-down box.
5. You can **Overwrite resources without warnings** and **Regenerate imported Integration Object(s)** by selecting either check box.

   **Note:** If you have chained Integration Objects, you should be sure to select the **Regenerate imported Integration Object(s)** option. This ensures that the chaining information is preserved. After you have imported the chained Integration Objects, the state information is in the Java source and will be prefilled when you use the Create Chaining Integration Object wizard. For more information about chained Integration Objects, see "Integration Object chaining" on page 342.
6. Click **Finish**.

**Note:** When importing Integration Objects, the connection configuration that is associated with the Integration Object is added to the HATS project

## J2EE migration

If you want to migrate the level of J2EE support in your HATS projects, perform the following steps:

1. In Rational SDP switch to the Navigator view.
2. Right-click the .ear project that contains your HATS projects, and select **Java EE > Specifications Upgrade Wizard**.
3. On the Welcome Page panel, click **Next**.
4. On the Enterprise Application panel, select the appropriate J2EE version and clear the **Migrate all module projects** box.

   **Note:** Clearing this box allows you to individually check which projects to migrate. If a HATS EJB project is contained in the .ear project, you must not select it in order to migrate the .ear project successfully.
5. On the EJB Module panel (if one appears), do not select any HATS EJB projects.
6. On the Web Projects panel, select all of the HATS Web projects.
7. Click **Finish**.

# Chapter 3. Developing and deploying HATS Web applications

Types of HATS applications include: Web, portlet, EJB, rich client, and administrative console. This chapter includes information about HATS Web applications. For information about the other types of applications, see the following chapters:

- EJB, the chapter, "Creating and using a HATS EJB application", in the *HATS Web Application Programmer's Guide*
- Rich client, Chapter 4, “Developing and deploying HATS rich client applications,” on page 53
- Administrative console, Chapter 18, “Using HATS administrative console,” on page 375

In addition, a HATS Web application can be linked with a WebFacing application and both packaged together to interoperate with each other in a single enterprise application. For more information, see Chapter 20, “WebFacing and HATS,” on page 399.

HATS applications are created from HATS projects using the HATS Toolkit. When you create a new HATS project, a set of folders is created to help you organize your HATS application files. An example of a default project is shown below. The highest level folder has the same name as the name you give to your project when you create it. In that folder are other high-level folders that contain objects defined in your HATS project. Some folders do not appear until you create certain objects.

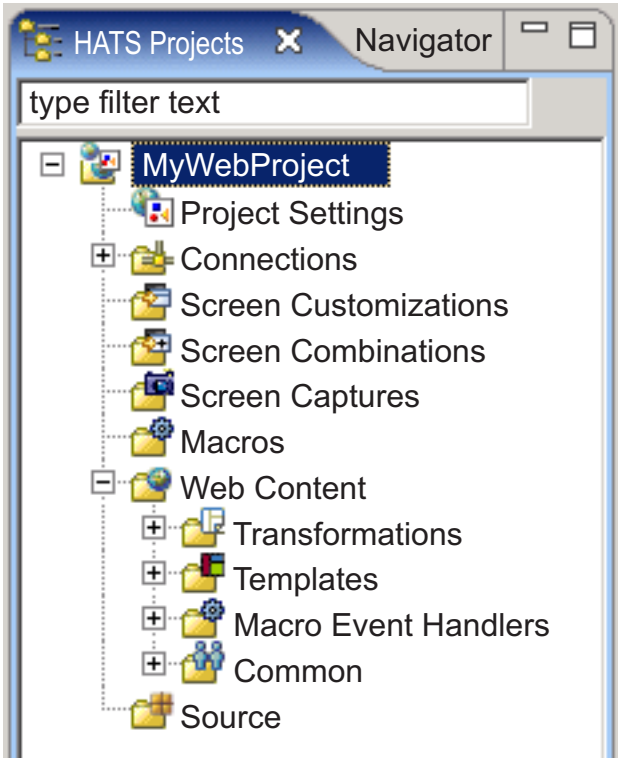



*Figure 2. HATS Web project view*

Depending on how you set up your HATS project, some or all of these folders appear in the **HATS Projects** view. You can also specify which folders appear in your **HATS Projects** view as well as hide file extensions. For more information, see "Using HATS preferences" on page 126.

**Note:** Different folders appear for different types of HATS projects. For example, the directory tree for a HATS EJB project has no **Screen Customizations**, **Screen Captures** or **Web Content** folders.

You can create subfolders within these high-level folders to help organize your project. For instance, as you create screen captures for your project, you might want to create folders under the **Screen Captures** folder to organize and group the captured screens. To create a folder, right-click on one of the high-level folders in the tree and click **New HATS > Folder**. To move a file into a different folder, right-click on the file and select **Move**, or you can use the drag-and-drop method.

**Note:** There is a limitation for subfolders. The transformation and the template files must be in subfolders of the same level. In order to use a transformation that resides at a certain level of subfolders, such as \transformations\Callup\, then the template that will be coupled with this transformation must be at the same level of subfolders, like \templates\Callup\.

HATS projects can be shared in a team environment by going to the **Navigator View** of your HATS perspective. Right-click the project and select **Team > Share Project**. Select the repository type from the list and click **Next**. Rational SDP supports several repositories. For more information, refer to the Rational SDP documentation and search on `repository`.

**Note:** When using a version control system with your HATS projects, set the system to ignore the `resourceUpdate.sts` file. This file is automatically generated when testing a project within the toolkit. The file should not be under version control and can safely be ignored or deleted before placing a HATS project under version control.

To exclude the file from version control, open **Window -> Preferences -> Team -> Ignored Resources** and click **Add Pattern** to add a new pattern. Enter `resourceUpdate.sts` and click **OK**. Make sure the new pattern is selected in the list of ignored patterns and click **OK** to save the settings.

HATS Web projects, created in HATS Toolkit, are extensions of Web projects in the Rational SDP workbench. For more information, click **Help > Help contents** from the Rational SDP menu bar, expand **Developing**, and select **Developing Web applications**.

## Organizing HATS Web applications

By default, all HATS Web applications are stored in one enterprise archive file, such as HATS_EAR8. When you export your applications and deploy them on WebSphere Application Server, the HATS .ear file contains a Web archive (.war) file with the resources to run each application, as well as one copy of the HATS runtime executable code. If you prefer, you can organize your applications differently, either each in its own .ear file, or in some other combination.

Consider the effect of the following on your server when deciding how to arrange your Web applications:

**Disk space**

If you create each application in its own .ear file, it has its own copy of the HATS runtime code. The runtime code is approximately 25 MB; multiply that by the number of applications you have to see how much disk space is consumed on your runtime system for all of your applications.

**Deployment**

If you redeploy a HATS .ear file, you are redeploying all the applications in that .ear file, even if some of the applications are unchanged.

**Logging and tracing**

Logging and tracing are controlled at the level of the .ear file, not at the individual HATS application level. If each HATS application is in its own .ear file, you can control its log and trace settings independently of any other applications. If you have several HATS applications in one .ear file, log and trace settings apply to all HATS applications in the .ear file. Messages for all HATS applications in the .ear file are inserted into the same log file, and trace information for all HATS applications is inserted into the same trace file.

You can add an additional optional keyword, traceLogDirectory, to the runtime.properties file. This enables you to specify a particular directory for the output files. The file is located in the *was_dir*/installedApps/*ear_name* directory for a HATS enterprise application.

**License tracking**

License tracking is also controlled at the level of the .ear file, not at the individual HATS application level. If each HATS application is in its own .ear file, license tracking is done independently of other applications. If you have several HATS applications in one .ear file, license tracking is performed for all HATS applications in the .ear file. Information about license usage is kept for all HATS applications in the .ear file, and is inserted into the same license usage file.

## Moving HATS Web projects to a different .ear file

Which .ear file your project files go into is determined when you create the project. After you create the project, you can move it from one .ear to another, using the following steps:

To add the project to an .ear file:

1. Click the **Navigator** tab of the HATS Toolkit to display the .ear files.
2. Expand the .ear file to which you want to add the project. Expand the **META-INF** folder and locate the application.xml file.
3. Start the Rational SDP application.xml editor by double-clicking the application.xml file.
4. In the application.xml editor Design view, in the Overview group, select **Application**, and click **Add**.
5. In the Add Item dialog, select **Module** and click **OK**.
6. Select the project you want to add to the .ear file and click **Finish**.
7. Close the Rational SDP application.xml editor.

To remove the project from an .ear file:

1. Click the **Navigator** tab of the HATS Toolkit to display the .ear files.
2. Expand the .ear file from which you want to remove the project. Expand the **META-INF** folder and locate the application.xml file.

3. Start the Rational SDP application.xml editor by double-clicking the application.xml file.
4. In the application.xml editor Design view, in the Overview group, select the project you want to remove from the .ear file, and click **Remove**.
5. Close the Rational SDP application.xml editor.

# Backing up your HATS projects

As with any software development activity, it is good practice to back up your HATS projects to ensure against data loss or corruption. In addition to creating backups during your own development cycle, you should also create backups prior to installing HATS maintenance. This allows you to revert to the previously installed maintenance level if necessary. To create a backup for your HATS project, use the Rational SDP export archive file function. For more information, see "Exporting a Web project."

To restore a HATS project from a backup, use the import archive file function. For more information, see "Importing a Web project."

# Exporting and importing HATS Web projects

## Exporting a Web project

You can export your HATS Web projects to save and use as backups or to move to another HATS Toolkit system. To perform the export, use the following method:

### Archive file

This function has multiple benefits. It allows you to export multiple projects and the associated .ear all at once. In addition, you do not have to create a project on the destination HATS Toolkit before you import the archive file. To export the project as an archive file perform the following steps:

1. From the menu bar select **File > Export** to open the Export wizard.
2. Select **General > Archive File** and click **Next**.
3. Select both the project and its associated EAR project.
4. Specify a file name and location for the file in which to save the project.
5. In the **Options** section, select the desired file format and compression options. To export the whole project, select **Create directory structure for files**.
6. Click **Finish**.

## Importing a Web project

To import a project in an archive file (or a project interchange file from an older release of Rational SDP) into HATS Toolkit:

1. From the menu bar click **File > Import** to open the Import wizard.
2. Select **General > Existing Projects into Workspace** and click **Next**.
3. Select the **Select archive file** option and click **Browse** to browse for the archive file.
4. In the **Projects** section, select the project (or projects) you want to import.
5. Click **Finish**.

After importing, if you get the following error:

```
Project 'xxxxx' is missing required source folder: 'Java Source'

The project cannot be built until build path errors are resolved
```

To fix this error perform these steps:

1. Select the project in the **HATS Projects** view.
2. From the Rational SDP toolbar, select **File > New > Other**.
3. On the "Select a wizard" page, expand **General**, select **Folder**, and click **Next**.
4. Ensure that your project is selected as the parent folder.
5. Enter `Java Source` as the folder name.
6. Click **Finish**.

For information about how to migrate a HATS project from a previous release of HATS, see "HATS migration" on page 13.

## Deploying HATS Web applications

The terms *HATS Web application*, *WebSphere application*, and *Java EE application* can be used interchangeably to refer to a Web application created from a HATS project. It is a HATS Web application because it was developed in the HATS Toolkit. It is a WebSphere application because it will be installed and run on WebSphere Application Server. It is a Java EE application because it conforms to the Java EE standards.

To deploy your HATS Web application in a runtime environment you must:

- Enable the HATS runtime.
- Export your project as a Java EE application.
- Install your application in a runtime environment.
- Make any changes necessary in the runtime environment.

The following sections describe how to perform each of these tasks.

**Note:** HATS Web applications can also be deployed to the Apache Geronimo web application server with the Tomcat web container, to the Oracle WebLogic Server, and to the IBM Bluemix Server. For information unique to developing and deploying HATS web applications for these servers, see:

- "Developing HATS applications for the Apache Geronimo Server" on page 36
- "Developing HATS applications for Oracle WebLogic Server" on page 37
- "Developing HATS applications for IBM Bluemix Server" on page 39
- "Developing HATS applications for Web Sphere Application Server Liberty Porfile" on page 42

### Enabling the HATS runtime

Regardless of where you obtain the HATS package (HATS CD, Web, or packaged with another product), you install the same version of the HATS Toolkit. This is a limited-use version that you can use to evaluate HATS. To fully enable the runtimes for production in accordance with your licensed proof of entitlement, you must specify your license settings using the License Settings wizard included in the HATS Toolkit. For more information see the section, Enabling HATS runtime and license settings, in *HATS Getting Started*.

## Exporting your project as a Java EE application

To deploy your HATS Web application to a WebSphere Application Server runtime environment, you must first package it into a Java EE application. To do this, you export your project as an EAR file.

To export the project, follow these steps:

1. Highlight (single-click) the name of your project in the HATS Projects view.
2. Click the **Export HATS Project** icon on the main tool bar, or right-click the project name and select **Export Project**.

   **Note:**

   If you have not specified your license settings, you will see a message that the runtime for this project is not enabled. Although this application can be deployed, users are restricted to two connections.

   HATS Web applications can be tested in a local test environment and deployed to a runtime (non-development) environment, but in either environment, these applications support only two host connections without specifying license settings. For information about specifying license settings, see Enabling HATS runtime and license settings in *HATS Getting Started*.

   If licenses have been purchased, click **Enable Runtime** to runtime enable your application. Otherwise, click **Continue** to continue the export process.
3. In the Export window, choose an Enterprise Application project from the drop-down list or type the name of an Enterprise Application project in the **EAR project** field.
4. Enter a destination location in the **Destination** field or click the **Browse** button to select the destination of the exported .ear file.
5. If you want to export the project's source files along with the executable files, select the **Export source files** box. If you include the source files, another developer can extract them from your .ear file. This can make collaboration or service easier, but you must decide whether it creates a security risk.
6. If you have exported this project to this location before, the export process asks whether you want to overwrite the existing files. This is intended to protect you from overwriting files that you might want to keep, perhaps to archive a previous version of your project. If you want to overwrite previous files without being asked for confirmation, select the **Overwrite existing file** box.
7. Click **Finish**. The project is exported as a Java EE application, represented by an Enterprise Archive file with extension .ear, with the directory and file name you specified.

## Installing your application in a runtime environment

After exporting the HATS project as a Java EE application and transferring the application's .ear file to the production system, install it by launching the WebSphere administrative console and browse to the location of the .ear file. For more information about installing applications on WebSphere Application Server refer to the WebSphere Application Server library at http://www.ibm.com/software/webservers/appserv/was/library/ and select the link to the Knowledge

Center for your version of WebSphere Application Server. In the contents under your WebSphere Application Server product, refer to the chapter about deploying applications.

After the application has been installed, test it by bringing up the URL in a Web browser on another system. Then you can publish the URL to your users. As an example, the URL might look like `http://hostname/hatsappname/` where `hostname` is the IP host name and domain where WebSphere Application Server is installed and `hatsappname` is the name of your HATS application.

**Notes:**

1. For more information about installing HATS applications on specific platforms, see the HATS Knowledge Center at http://www.ibm.com/support/knowledgecenter/SSXKAY_9.7.0.
2. If your HATS applications are deployed on WebSphere Application Server with Java 2 security enabled, and they do not start, see "Java 2 security" on page 419 for more information.
3. HATS supports multiple browser instances from the same workstation accessing the same HATS application. However, these multiple browser instances must not share the same HTTP session id.

   The session between the browser accessing the HATS application and the WebSphere Application Server is assigned an HTTP session ID. The HTTP session ID enables subsequent requests from the same browser to return to the same HATS application. Subsequent requests use the same Telnet (3270 or 5250) session that exists between the HATS application on the server and the host.

   Different browsers implement the HTTP session ID differently. Microsoft Internet Explorer expects all browser sessions in the same process to share the same HTTP session ID. When you launch a new browser window using the same shortcut, a new browser session starts and shares the same process and the same HTTP session ID.

   One way to open a new Internet Explorer process with a unique HTTP session ID, in Windows XP for example, if Internet Explorer is in your start list, is to select **Start > Internet Explorer**. Selecting **Start > All Programs > Internet Explorer** also starts a new Internet Explorer process with a unique HTTP session ID.

   **Notes:**

   a. The Ctrl-N function of Internet Explorer does not start a new process, because it opens a new window that has the same process ID and the same HTTP session ID as the window from which it originated.
   b. For Internet Explorer 8, you can use **File > New Session** to open a new browser with a unique session ID.

   **Note:**

   In Internet Explorer, you can find the HTTP session ID for a browser window by selecting **View > Source** from the menu bar, or right-clicking in the Internet Explorer browser window and selecting **View Source** to view the source tagging for the page. Scroll down to the HATS Form tag in the source window, which is similar to the following:

   ```
   <FORM NAME="HATSform" dir="" METHOD="POST"
   ACTION="/YourHATSAppName/entry">.
   ```

   Within that Form tag, find the SESSIONID parameter, which is similar to the following:

```
<INPUT TYPE="HIDDEN" NAME="SESSIONID"
VALUE="PIOncS1GehNWxWo_49VCBzC" />
```

**Note:** When you access a HATS application that only displays a default transformation of the screen and doesn't run macros when it starts, the SESSIONID has a value of "INVALID." The SESSIONID value is assigned when you send something to the host, such as pressing the Enter key, and you get a response.

You can verify whether a new process is being started by using Task Manager in Windows. Start Task Manager, and click the 'Image Name' column heading on the Processes tab to sort the information by image name. Scroll down to see how many 'iexplore.exe' processes are in the list. Try starting some new Internet Explorer windows using the **Start > Internet Explorer** shortcut. If a new process is started, a new iexplore.exe process is added to the display. Otherwise, an existing process is in control of the new browser window.

4. HATS requires the application's display-name, as defined in the application's web.xml file, to match the Web application's context path used in the browser URL. If the display-name value is changed to be different than the context path of the application, you might encounter a java.lang.nullPointerException error or a javax.servlet.ServletException error when running the HATS application. For example, a HATS project named MyHatsProject defaults to a display-name of MyHatsProject. This project, when deployed as a Web application, must be reached with a URL similar to http://myServer/MyHatsProject/ to operate properly. Changing the display-name to something other than MyHatsProject causes the application to be inoperable.

# Changes necessary in the runtime environment

## Configuring class loader policy

When you deploy your HATS Web applications, or when running the applications in the local test environment, the WAR class loader policy must be configured on a per Java EE application basis for either of the following situations:

- Your application includes business logic. Your business logic might cause a ClassNotFoundException to occur.
- Your application includes custom components or custom widgets. The components or widgets might cause a ClassNotFoundException to occur.

HATS automatically configures the class loader policy for these applications.

## Configuring HATS applications in a clustered environment

If you deploy HATS Web applications in a vertically clustered environment, each application server instance will create its own files for logging, tracing, and license tracking. This is accomplished by decorating the names of the output files with the fully qualified name of the application server instance. For example, the default pattern for the logging file is messages.txt, but the actual file name will be something like messages*_myCell_myNode_myAppServerInstance*_1.txt.

By default, all of the server instances read the same runtime.properties file for their settings. To properly control runtime settings you must configure each instance to have its own runtime properties file. This enables you to control tracing for each instance independently, and prevents runtime settings from changing spontaneously.

Follow these steps to configure each of your vertically clustered HATS application instances to have its own runtime.properties file:

1. Make a copy of the runtime.properties file for each application server instance in the vertical cluster.
   a. Locate the runtime.properties file for your HATS application. This file should reside in the installedApps*app_name.ear* directory under the directory where you installed WebSphere Application Server.
   b. For each instance, make a copy of the runtime.properties file, with a unique name, in the same directory. For example, you might name the files Clone1runtime.properties, Clone2runtime.properties, and so on. You can use any valid file name, but the name should help the administrator identify the application server instance with which this file is associated.
   c. At this point, for `n` server instances you have `n` unique runtime properties files.
   d. If you have more than one HATS .ear file in your vertically clustered environment, repeat this step in each .ear file directory.
2. Add a new configuration setting to identify the runtime.properties file that is used by each instance. For example, for WebSphere Application Server V6.x:
   a. Select **Application Servers** in the **Servers** item in the left navigation pane of the WebSphere Application Server administrative console.
   b. Select the server instance from the list of application servers.
   c. Select **Java and Process Management** from the **Configuration** tab for the server.
   d. Select **Process Definition > Java Virtual Machine > Custom Properties**.
   e. In the Custom Properties window, click **New**.
   f. In the **Name** field, enter `hats.runtime.properties`.
   g. In the **Value** field, enter the name of the properties file that you created for this server, for example, Clone2runtime.properties. Do not specify directory names or slashes in this value.
   h. The **Description** field does not require a value.
   i. Click **Apply**.
   j. Repeat this procedure for each server instance.

After adding the new custom property and ensuring that each server instance has its own, uniquely named, copy of the runtime.properties file, you must restart each application to begin using the new files.

The HATS administrative console can be used to control the settings of the cluster members. Use the **Getting Started** folder in the Navigation panel of the Administration Web page to select the **Management Scope**. After you have chosen the cluster, choosing to view the **Trace Settings** will prompt you for the particular cluster member you want to control.

**Note:** There are special considerations for the use of clustering and user lists. For more information, see "Clustering and user lists" on page 145

## Configuring HATS applications to use a proxy server

If users access your HATS application through a proxy server, you must configure both the proxy server and a context parameter in the HATS application.

To configure the proxy server:

1. On the proxy server, locate the httpd.conf file.
2. Using a text editor such as Notepad, add the following two lines to the file:

```
ProxyPass /application_name/ http://yyyy:port/application_name/
ProxyPassReverse /application_name/ http://yyyy:port/application_name/
```

Where *application_name* is the name of your HATS application, *yyyy* is the fully-qualified IP address of the application server where the HATS application is installed, and *port* is the port number.

To configure the context parameter in the HATS application:

1. Edit the web.xml file of the HATS application (located in the Web Content\WEB-INF folder in the **Navigator** view).
2. Add the following context parameter:

```
<context-param>
  <param-name>com.ibm.hats.proxyURL</param-name>
  <param-value>http://myproxyserver.com:port/application_name</param-value>
</context-param>
```

Where *myproxyserver.com* is the url of the proxy server, `port` is the port number, and *application_name* is the name of your HATS application.

# Developing HATS applications for the Apache Geronimo Server

You can develop, test, and deploy HATS web applications targeted for the Apache Geronimo web application server with the Tomcat web container. For information about which releases of Apache Geronimo are supported, see "System Requirements for Host Access Transformation Services" at http://www.ibm.com/support/docview.wss?uid=swg27011794. For detailed information about Apache Geronimo, including documentation and downloads, see the Welcome to Apache Geronimo site at http://geronimo.apache.org .

## Installing and configuring for Geronimo servers

This section describes a summary of how to install Apache Geronimo and configure Rational Software Delivery Platform (Rational SDP) to enable creating HATS web applications targeted for Apache Geronimo servers:

Download and install the Apache Geronimo server with the Tomcat web container:

1. Download Apache Geronimo from the Download site at http://geronimo.apache.org/downloads.html.
2. Install the server by extracting the .zip or .tar files.
3. Run the server from a command line window by changing the directory to *geronimo_home*/bin and entering the command: `geronimo run`.

The Geronimo Eclipse Plugin (GEP) provides integration between Eclipse, its Web Tools Platform (WTP) project, and Geronimo that enables you to utilize WTP features to develop, test, and deploy applications for the Apache Geronimo server using Rational SDP.

Install the Geronimo Eclipse Plugin (GEP):

1. Follow the instructions on the Installing Geronimo Eclipse Plugin site at https://cwiki.apache.org/GMOxDOC22/installing-geronimo-eclipse-plugin.html.
2. GEP v2.2 includes server adapters for v1.1.x, v2.0.x, v2.1.x, and v2.2 Apache Geronimo servers.

Configure Geronimo as a target server in Rational SDP for HATS web applications:

1. On the Rational SDP menu bar, click **Window > Preferences**.

2. In the left panel, expand **Server**.
3. Click **Runtime Environments**.
4. In the Server Runtime Environments panel, click **Add**.
5. In the New Server Runtime Environment dialog, expand the **Apache** folder.
6. Select the appropriate version of the **Apache Geronimo** server.
7. Optional. Select the **Create a new local server** box to create a new Geronimo server on your workspace **Servers** tab. Doing this step enables you to test your Geronimo application within the Rational SDP local test environment using the Run on Server, Debug on Server, and Profile on Server functions.
8. Click **Next** to go to the panel where you point to your local Geronimo server. In the *Application server* **installation Directory** field enter the root folder where you extracted the Geronimo server files. For example, if you extracted your Geronimo server zip to a folder named `C:\geronimo-tomcat6-javaee5-2.1.7`, enter this value in the installation directory field.
9. Click **Finish**.
10. Click **OK** to close the Preferences panel.

After following these steps, you can create HATS applications within Rational SDP targeted for Apache Geronimo servers.

## Considerations and limitations for Geronimo servers

There are some different considerations between developing, testing, and running HATS web applications on Apache Geronimo server versus WebSphere Application Server. For example, only one HATS web application is supported per .ear file running on an Apache Geronimo server.

For up-to-date support considerations, see "Host Access Transformation Services 9.7 - Known issues and workarounds" at http://www.ibm.com/support/docview.wss?uid=ibm10876092.

# Developing HATS applications for Oracle WebLogic Server

You can develop, test, and deploy HATS web applications targeted for the Oracle WebLogic Server. For information about which releases of Oracle WebLogic Server are supported, see "System Requirements for Host Access Transformation Services" at http://www.ibm.com/support/docview.wss?uid=swg27011794. For detailed information about Oracle WebLogic Server, including documentation and downloads, see the Oracle web site at http://www.oracle.com/technetwork/middleware/weblogic/overview/index.html .

## Installing and configuring for WebLogic servers

This section describes a summary of how to install Oracle WebLogic Server and configure Rational Software Delivery Platform (Rational SDP) to enable creating HATS web applications targeted for Oracle WebLogic Server:

Download and install the Oracle WebLogic Server using download links and instructions on the Oracle site at http://www.oracle.com/technetwork/middleware/weblogic/downloads/index.html.

Install the Oracle WebLogic Server tools in RAD versions that are earlier than 9.0:

1. On the Rational SDP menu bar, click **Help > Install New Software**.
2. Click **Runtime Environments**.

3. In the Server Runtime Environments panel, click **Add**.
4. Click **Download additional server adapters**.
5. Select **Oracle WebLogic Server Tools** from the list and click **Next**.
6. Accept the terms of the licensing agreement and click **Finish**.
7. When prompted to restart after the installation completes, click **OK**.
8. After the installation completes, when prompted to restart now, click **Yes**.

Install the Oracle WebLogic Server tools in RAD 9.0:

1. On the Rational SDP menu bar, click **Window > Preferences**.
2. In the upper right panel, click**Available Software Sites**.
3. Click **Add**.
4. Enter the following information to add the required repository:
   Name: Oracle Enterprise Pack for Eclipse Repository
   Location: http://download.oracle.com/otn_software/oepe/12.1.2/juno/repository
5. Click **OK**.
6. Enable the following additional repositories in the Available Software Sites list, or add them if you do not have them:
   Name: Eclipse Juno repository
   Location: http://download.eclipse.org/releases/juno
   Name: Eclipse Web Tools Platform Repository
   Location: http://download.eclipse.org/webtools/updates/
7. Click **OK** to close the Preference window.
8. Select the **Contact all update sites during install to find required software** check box.
9. Select **Work With > Oracle Enterprise Pack for Eclipse Repository**.
10. Expand the results and select the **Oracle WebLogic Server Tools** check box.
11. Click **Next**.
12. Follow the instructions in the wizard to complete the installation.

Install the Oracle WebLogic Server tools in RAD 9.7:

1. On the Rational SDP menu bar, click **Help > Eclipse Marketplace**.
2. In the Find textbox, search for **WebLogic**.
3. Install the Oracle WebLogic Server Tools.
4. Click **Confirm** on the Confirm Selected Features panel.
5. Read and accept the terms of the licensing agreement and click **Finish**.
6. After the installation completes, click **Yes** when you are asked if to restart.

Configure WebLogic as a target server in Rational SDP for HATS web applications:

1. On the Rational SDP menu bar, click **Window > Preferences**.
2. In the left panel, expand **Server**.
3. Click **Runtime Environments**.
4. In the Server Runtime Environments panel, click **Add**.
5. In the New Server Runtime Environment dialog, expand the **Oracle** folder.
6. Select the appropriate version of the **Oracle WebLogic Server**.
7. Optional. Select the **Create a new local server** box to create a new WebLogic server on your workspace **Servers** tab. Doing this step enables you to test

your WebLogic application within the Rational SDP local test environment using the Run on Server, Debug on Server, and Profile on Server functions.

**Note:** If you perform this step, you must modify the default Publishing mode so that your projects will deploy correctly when using the Run on Server, Debug on Server, and Profile on Server functions.

a. Find the instance of the WebLogic Server that was created in the **Servers** view.
b. Right-click on the WebLogic Server and click **Properties**.
c. In the left panel, expand **WebLogic**.
d. Click **Publishing**.
e. In the Publishing mode section, click **Publish as an exploded archive**.
f. Click **OK** to close the Properties panel.

8. Click **Next** to go to the panel where you point to your local WebLogic server. In the **WebLogic home** field enter the root folder where you installed the WebLogic server. For example, if your WebLogic installation uses the root folder named `C:\Oracle\Middleware`, enter `C:\Oracle\Middleware\wlserver_12.1` in the **WebLogic home** field. If you installed the JDK that was included with the WebLogic server installer then the **JAVA home** field should be filled in automatically with a value such as `C:\Oracle\Middleware\jdk160_29`. If you did not install the included JDK then you must enter into the **JAVA home** field the appropriate location of the JAVA runtime being used.
9. Click **Finish**.
10. Click **OK** to close the Preferences panel.

After following these steps, you can create HATS applications within Rational SDP targeted for Oracle WebLogic Server.

## Considerations and limitations for WebLogic servers

There are some different considerations between developing, testing, and running HATS web applications on Oracle WebLogic Server versus WebSphere Application Server. For example, only one HATS web application is supported per .ear file running on an Oracle WebLogic Server.

For up-to-date support considerations, see "Host Access Transformation Services 9.7 - Known issues and workarounds" at http://www.ibm.com/support/docview.wss?uid=ibm10876092.

# Developing HATS applications for IBM Bluemix Server

You can develop, test, and deploy HATS web applications targeted for the IBM Bluemix Server. For detailed information about IBM Bluemix Server, including documentation, see the Bluemix web site at http://www.ibm.com/cloud-computing/bluemix/. For additional details on Rational SDP bluemix features, seehttp://www.ibm.com/support/docview.wss?uid=swg27046332.

## Configuring Rational SDP for IBM Bluemix server

This section describes how to configure Rational Software Delivery Platform (Rational SDP) to enable creating HATS web applications targeted for IBM Bluemix server.

Configure IBM Bluemix server as a target server in Rational SDP for HATS web applications:

1. On the Rational SDP menu bar, click **Window > Preferences**.
2. In the left panel, expand **Server**.
3. Click **Runtime Environments**.
4. In the Server Runtime Environments panel, click **Add**.
5. In the New Server Runtime Environment dialog, expand the **IBM** folder.
6. Select **IBM Bluemix Runtime**.
7. Optional: Select the **Create a new local server** box to create a new Bluemix server on your workspace Servers tab. Doing this step enables you to test your Bluemix application within the Rational SDP local test environment using the Run on Server, Debug on Server, and Profile on Server functions.
8. Click **Next** to go to the panel where you enter your Bluemix account details. Enter your Bluemix account email address and password and validate the account.
9. Click **Next** to go to the panel where you get Organizations and Spaces. Select the space you want.
10. Click **Finish**, and then click **OK** to close the Preferences panel.

After following these steps, you can create HATS applications within Rational SDP targeted for IBM Bluemix Server.

## Considerations and limitations for IBM Bluemix Server

There are some different considerations between developing, testing, and running HATS web applications on IBM Bluemix Server versus WebSphere Application Server.

- One HATS web application is supported per .ear file running on IBM Bluemix Server.
- The HATS administrative console is not supported.
- Display Terminal is not supported (trace.HOD.DISPLAYTERMINAL).
- IBM Bluemix does not support the IBM WebSphere JAX-WS Web service runtime using Rational SDP.
- How to view HATS trace/log files

Log file location in Bluemix server:

```
/home/vcap/app/wlp/usr/servers/defaultServer/apps/myapp.ear/logs/
```

Basically, you can observe the path of trace file in the Bluemix server console (HATS studio console) as given below:

```
SystemOut  O HAT0002
The message log file is:
/home/vcap/app/wlp/usr/servers/defaultServer/apps/
      myapp.ear/logs/messages_XXXXXXXX_XX.txt
```

To view log files using Bluemix website:

- Log in to Bluemix home page in the browser.
- Open the HATS application **Overview** page.
- Click **Files and Logs** in the left side menu.
- Navigate to the path given above, starting from:

```
/app/wlp/usr/servers/defaultServer/apps/myapp.ear/logs/
```

To view log files using Rational SDP Remote Systems View:

- In the Rational SDP Servers view, double click **IBM Bluemix Server**.
- By default, the **Overview** tab is open. Go to the **Applications and Services** tab.
- Select your HATS application under **Applications**.
- Click the **Remote Systems View** link at the bottom-right corner. The Remote Systems View opens for Bluemix Server. Navigate till you see your HATS application.
- Navigate to the path given above for the log files.

**Notes:**

1. While adding Bluemix Runtime in Rational SDP, you might see the following error:

   ```
   java.lang.ClassNotFoundException:
   Cannot find the specified class com.ibm.websphere.ssl.protocol.SSLSocketFactory
   ```

   The workaround for the SSLSocketFactory error message is to look for the `com.ibm.ws.ast.st.core.prefs` file available in `x:\workspace\.metadata\.plugins\org.eclipse.core.runtime\.settings`, where `x:\workspace` is the directory of your workspace, and replace `isUseIBMSSLSocketFactory=true` value with `'false'`. Then, restart the workbench.

2. Exceptions in Bluemix console while deploying HATS ear in Bluemix Server:

   ```
   FFDC1015I: An FFDC Incident has been created:
   "com.ibm.wsspi.adaptable.module.UnableToAdaptException:
   com.ibm.ws.javaee.ddmodel.DDParser$ParseException:
   CWWKC2262E: The server is unable to process the 7 version and the
   http://xmlns.jcp.org/xml/ns/javaee namespace in the /META-INF/application.xml
   deployment descriptor on line 2.
   com.ibm.ws.app.manager.ear.internal.EARDeployedAppInfoFactoryImpl 132"
   at ffdc_15.09.11_05.03.09.0.lo .....
   ```

   By Default HATS applications are created using J2EE 7 version for Bluemix server, so you can set the following environment variable in your HATS Bluemix application in order to include a major subset of the Java EE 7 feature set. You can then add or remove additional features as needed. While deploying HATS application in Bluemix server you can configure this environment variable.

   ```
   name: JBP_CONFIG_LIBERTY
   value: app_archive: {features: [ webProfile-7.0, cdi-1.2, jaxrs-2.0,
   jpa-2.1, websocket-1.1, servlet-3.1, jsp-2.3, ejbLite-3.2 ]}
   ```

   For more details, refer to the article https://developer.ibm.com/bluemix/2015/08/28/updates-to-ibm-eclipse-tools-for-bluemix-august-2015/.

To remote debug HATS business logic or any custom java files using Bluemix server, see http://www.ibm.com/support/docview.wss?uid=swg27046332#whatsnew_95_bluemix_debug.

## Developing HATS applications for Web Sphere Application Server Liberty Porfile

You can create HATS applications within Rational SDP targeted for WebSphere Application Server Liberty Profile by following the steps below:

1. On the Rational SDP menu bar, click **Window** > **Preferences**.
2. In the left panel, expand **Server**.
3. Click **Runtime Environments**.
4. In the Server Runtime Environments panel, click **Add**.
5. In the New Server Runtime Environment dialog, expand the **IBM** folder.
6. Select **WebSphere Application Server Liberty Profile**.
7. Optional: You can test your Liberty Profile application within the Rational SDP local test environment using the Run on Server, Debug on Server, and Profile on Server functions. Select the **Create a new local server** box to create a new Liberty Profile server on your workspace **Servers** tab.
8. Click **Next** to go to the panel where you point to your local Liberty Profile server. Click the **Choose an existing installation radio** button and enter the location where you have WebSphere Application Server Liberty Profile installed in the **Path** field.

   **Note:** If you do not have installed a Liberty Profile server installed, you can install one directly from the **New Server Runtime Environment** dialog box. The steps are listed below.

   a. Click the **Install from an archive or repository** radio button.
   b. Click **Next** to go to the Install Runtime Environment panel.
   c. Enter a path such as C:\Liberty in the **Enter the destination path** field.
   d. Click the **Download and install a new runtime environment from ibm.com** radio button and select **IBM WebSphere Liberty Repository** in the drop down menu.
   e. Select one of the WAS Liberty servers listed that include the runtime features, such as WAS Liberty V8.5.5.7 Runtime, and click **Next**.
   f. Choose any additional bundles you want on the Install Additional Content panel and click **Next**.
   g. Accept the terms of the licensing agreement and click **Finish**.
9. Click **Finish**.
10. Click **OK** to close the Preferences panel.

## Considerations and limitations for WebSphere Application Server Liberty Profile

When using Debug on server to test your application, a display issue might occur on the terminal window. You might receive the following error message:

```
ERROR HPS5018 An unexpected exception has occurred.
  Frame unable to display: java.awt.HeadlessException.
```

If this error occurs in the console, you can fix this problem by following these steps.

1. In the HATS Projects view, open the **Servers** tab.
2. Right-click on the WebSphere Application Server Liberty Profile server listed and click on **New** > **Server Environment File** > **jvm.options**.
3. Add the value to the file:

```
-Djava.awt.headless=false
```

4. Save and close the jvm.options file.
5. Restart the server to activate the changes.

# Developing HATS applications for the JBoss EAP Server

You can develop, test, and deploy HATS web applications targeted for the JBoss Enterprise application server. For information about which releases of JBoss EAP are supported, see "System Requirements for Host Access Transformation Services" at http://www.ibm.com/support/docview.wss?uid=swg27011794. For detailed information about Jboss EAP, including documentation and downloads, see the Red Hat JBoss Enterprise Application Platform Overview site at https://developers.redhat.com/products/eap/overview .

## Installing and configuring JBoss EAP

Configure JBoss EAP as a target server in Rational SDP for HATS web applications:

1. On the Rational SDP menu bar, click **Window > Preferences**.
2. In the left panel, expand **Server**.
3. Click **Runtime Environments**.
4. In the Server Runtime Environments panel, click **Add**.
5. In the New Server Runtime Environment dialog, expand the **Red Hat JBoss Middleware** folder for JBoss EAP.
6. Select the appropriate version of the **JBoss Enterprise Application** server.
7. Optional. Select the **Create a new local server** box to create a JBoss server on your workspace **Servers** tab. Doing this step enables you to test your JBoss application within the Rational SDP local test environment using the Run on Server, Debug on Server, and Profile on Server functions.
8. Click **Next** to go to the panel where you point to your local JBoss server. In the *Application server* **installation Directory** field enter the root folder where JBoss server is installed. For example, if JBoss server is installed in a folder named `C:\EAP-7.1.0`, enter this value in the installation directory field.
9. Click **Finish**.
10. Click **OK** to close the Preferences panel.

After following these steps, you can create HATS applications within Rational SDP targeted for JBoss servers.

## Considerations and limitations for JBoss servers

There are some different considerations between developing, testing, and running HATS web applications on JBoss server versus WebSphere Application Server. For example, only one HATS web application is supported per .ear file running on a JBoss server.

For up-to-date support considerations, see "Host Access Transformation Services 9.7 - Known issues and workarounds" at http://www.ibm.com/support/docview.wss?uid=ibm10876092.

## Developing HATS applications for mobile devices

You can develop HATS Web applications that can be accessed by mobile devices such as cellular phones, data collection terminals, and personal digital assistants (PDAs). The process is the same as developing any HATS Web application with some considerations due to the capabilities of the mobile device and its Web browser, for example screen size and interaction limitations.

To create a project for your application for mobile devices, launch the Create a Project wizard using one of the following actions:

- Select **HATS > New > Project** from the menu bar.
- Select **File > New > HATS Project** from the menu bar.
- Click the **Create HATS Project** icon on the toolbar.

In the Create a Project wizard, on the HATS Project panel:

1. Enter a name for the mobile device project.
2. Optionally, enter a description.
3. Accept the default location.
4. Select **Web** to indicate this is a Web project.

   **Note:** If the Web deployment option is disabled, this indicates that no server runtimes are defined. To define server runtimes, go to **Window > Preferences > Server > Installed Runtimes** and add at least one runtime definition.
5. For the Target server field, select one of the supported WebSphere Application Servers.
6. For the Enterprise application project field, enter the name of the EAR project to use for testing in the local test environment.
7. Select **Optimize options for mobile devices**.

   **Note:** If you are developing a HATS application for an iPad device, do not select this option. For more information see "Considerations and limitations for iPad devices" on page 51.
8. Click **Next** and continue creating your project as normal.

## Considerations and limitations for mobile devices

When you create a HATS project and select **Optimize options for mobile devices**, HATS initializes the project with options that work best for mobile devices. Some options, for example, printing, keyboard, asynchronous update, and other options, are not supported and therefore disabled. Keep in mind the following considerations and limitations as you develop your application for mobile devices.

- There is not an automated option that allows you to convert a HATS project that is not optimized for mobile devices to one that is optimized for mobile devices, or vice-versa.
- The option to specify a project theme is not provided. Instead, options are automatically initialized to work best for mobile devices.
- Only templates that are optimized for mobile devices are provided for use in the project.

  **Note:** If you do not want to have horizontal scrolling using the Modern template, add the `overflow: auto;` style to `.roundedcornermodule` in the Modern.jsp template file. For example, change `.roundedcornermodule`

`{margin: 0px;}` to `.roundedcornermodule { margin: 0px; overflow: auto; }`. This forces the page to keep its original size and keep within the visual screen on the device. Note that in some cases this might result in data on the page being truncated, and therefore the design of customized pages and widget choices should take the screen size into account.

- A second rendering set, named **compact**, is created in the project. This rendering set is specified as the default rendering set. It also has the **Use compact rendering** option selected which reduces the amount of HTML and blank space in default rendering, which in turn may display a different structure of the original host screen.
- The HATS preference, **Include a Free Layout Table**, that takes effect when you create a new blank transformation, has as additional modifier, **Except when the project is optimized for mobile devices**. This modifier is selected by default. Therefore, blank transformations added to your mobile project will not include a free layout table by default.
- Field Exit, Field+, and Field- can be used by using the **Enable cursor positioning option on input fields** option in combination with the Host Keypad or Host Key buttons.
- The Field widget provides a **Separated** layout option to render output using inline span tags, instead of using a table, to differentiate between fields. The goal of using this option is to reduce the amount of HTML and blank space. This is the default for mobile projects.
- HATS provides **Columns placement** support for subfile and table widgets. This is useful when displaying table data on a mobile device by allowing the arrangement and exclusion of columns from the display, as well as by allowing expandable details sections so the table can fit into a smaller space.

  The details section, when expanded, is displayed directly below the row containing the primary columns of data. Once a particular row of interest is identified by the user, the details of that row, when expanded, are displayed in a format that flows down the screen rather than to the right. This enables small displays to view the needed details of an arbitrary number of columns without resorting to horizontal scrolling.

  The figures below show a subfile as displayed on a host screen, followed by the primary view of the subfile and a details view of the subfile using the default Columns placement support.

```
 1= Add license key  2=Change  5=Display detail  6=Print detail
 8=Work with license users ...

               License
Opt   Product    Term    Feature  Description
__    5722SS1   V5R4M0    5050    i5/OS
__    5722SS1   V5        5051    i5/OS
__    5722SS1   V5R4M0    5103    Media and Storage Extensions
__    5722SS1   V5        5109    NetWare Enhanced Integration
__    5722SS1   V5R4M0    5112    PSF 1-45 IPM Printer Support
__    5722SS1   V5R4M0    5113    PSF 1-100 IPM Printer Support
__    5722SS1   V5R4M0    5114    PSF Any Speed Printer Support
                                                            More...
```

*Figure 3. Subfile on a host screen*

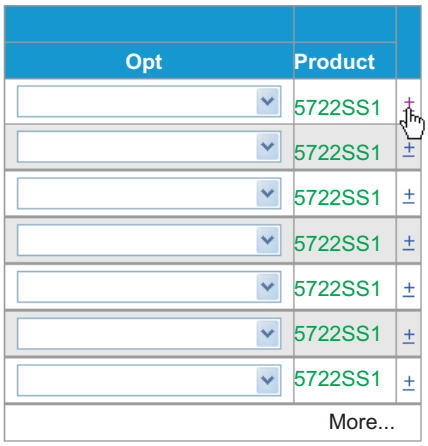



*Figure 4. Primary view of the subfile using default Columns placement support*

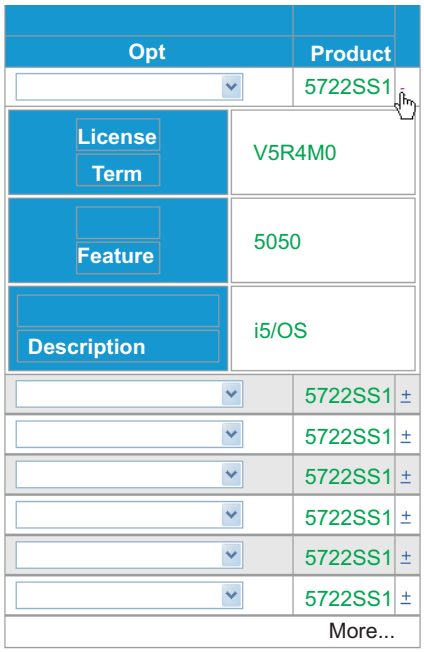



*Figure 5. Details view of a subfile row using default Columns placement support*

In addition to saving screen real estate, you can configure the widget to keep the detail columns on the server until requested rather than sending them using HTTP to the user's browser. This allows for a reduction in some cases of the amount of data transferred because unwanted detail data is never sent over HTTP to the end device. Only the details that are specifically requested by the user are retrieved on-demand and sent to the browser.

**Note:** Subfiles configured to use the field component to recognize the data portion of the subfile cannot effectively use primary and detail column capability due to the fact that such subfiles cannot distinguish columns, but only rows.

In addition to the **Columns placement** subfile and table widget settings described in "Widget settings" on page 218, the following settings can be used to further customize the appearance of the controls used for this option. These settings do not appear in the HATS Toolkit GUI. Instead, you must add these settings to the source for the widget. The source below shows examples of these settings using a Subfile widget as an example.

```
<class name="com.ibm.hats.transform.widgets.SubfileWidgetV6">
     <setting name="normalColumnLayout" value="1,2"/>
     <setting name="extendedColumnLayout" value="3*"/>
     <setting name="keepExpansionOnServer" value="true"/>
     <setting name="expandHeaderValue" value="fred"/>
     <setting name="expandRepresentation" value="button"/>
     <setting name="expandValue" value="click me"/>
     <setting name="collapseRepresentation" value="image"/>
     <setting name="collapseValue" value="/common/images/twisty1.gif"/>
     <setting name="collapseAltValue" value="hide"/>
</class>
```

**expandRepresentation**
Determines the graphical representation used for the control that shows the details. Specify **button**, **link**, or **image**. The default is **link**.

**expandValue**
If expandRepresentation is either **button** or **link**, this value is used as text on the button or link. If no value is supplied, a plus sign (**+**) is used by default. An empty string value"" is not accepted and defaults to a plus sign (**+**).

If expandRepresentation is **image**, this value is the path to and the name of the image file to use. The path is searched relative to the Web Content directory of the project. For example, if you want to specify the twisty1.gif in the Web Content/common/images directory, you set expandValue as shown below:

```
<setting name="expandValue" value="/common/images/twisty1.gif "/>
```

**expandAltValue**
This setting provides alternate text for the image. It is used only when **expandRepresentation** is set to **image** and the browser cannot display the specified image, for example, the **Menu > View > Show Pictures** option of Internet Explorer Mobile is not selected.

**collapseRepresentation**
Determines the graphical representation used for the control that hides the details. Specify **button**, **link**, or **image**. The default is **link**.

**collapseValue**
If collapseRepresentation is either **button** or **link**, this value is used as text on the button or link. If no value is supplied, a minus sign (**-**) is used by default. An empty string value"" is not accepted and defaults to a minus sign (**-**).

If collapseRepresentation is **image**, this value is the path to and the name of the image file to use. The path is searched relative to the Web Content directory of the project. For example, if you want to specify the twisty1.gif in the Web Content/common/images directory, you set collapseValue as shown below:

```
<setting name="collapseValue" value="/common/images/twisty1.gif "/>
```

**collapseAltValue**
This setting provides alternate text for the image. It is used only when **collapseRepresentation** is set to **image** and the browser cannot display

the specified image, for example, the **Menu > View > Show Pictures** option of Internet Explorer Mobile is not selected.

**expandHeaderValue**
This setting provides the header text for the column for the details controls. A value specified as "" creates a header with no text.

**expandStyle, expandClass, expandHeaderStyle, expandHeaderClass, expandAreaStyle, expandAreaClass, expandRowStyle, expandRowClass, collapseStyle, collapseClass, collapseAreaStyle, collapseAreaClass**
These are class and style overrides for the controls for this option. If not specified, the standard values for widgets apply. Standard HTML styles can be used, and classes are applied in order.

- The setting, **Enable cursor positioning option on input fields**, available for widgets used to render input fields, provides users a method of switching from data input mode to cursor positioning mode for input fields. When in data input mode, the user can enter data into the input field. When in cursor positioning mode, the user can tab to or otherwise position the cursor on any character in the input field. This setting is useful for devices that do not have other cursor positioning capabilities.

  In addition to the **Enable cursor positioning option on input fields** widget settings described in "Widget settings" on page 218, the following settings can be used to further customize the appearance of the controls used for this option. These settings do not appear in the HATS Toolkit GUI. Instead, you must add these setting to the source for the widget. The source below shows examples of these settings using a Field widget as an example.

```
<class name="com.ibm.hats.transform.widgets.FieldWidget">
     <setting name="dataModeCEPRepresentation" value="button"/
     <setting name="dataModeCEPValue" value="back"/>
     <setting name="dataModeCEPAltValue" value="data entry"/>
     <setting name="cursorModeCEPRepresentation" value="image"/>
     <setting name="cursorModeCEPValue" value="cursorOn.gif"/>
     <setting name="cursorModeCEPAltValue" value="cursor entry"/>
     <setting name="cursorCEPRepresentationStyle"
              value="text-underline:true"/>
     <setting name="useCursorExactPositioningOption" value="true"/>
</class>
```

  **dataModeCEPRepresentation**
  Determines the graphical representation used for the control shown next to the input field when in the data mode. Specify **button**, **link**, or **image**. The default is **link**.

  **dataModeCEPValue**
  If dataModeCEPRepresentation is either **button** or **link**, this value is used as text on the button or link. If no value is supplied, an asterisk character (*) is used by default. An empty string value"" is not accepted and defaults to an asterisk character (*).

  If dataModeCEPRepresentation is **image**, this value is the path to and the name of the image file to use. The path is searched relative to the Web Content directory of the project. For example, if you want to specify the sbarleftblk.gif in the Web Content/common/images directory, you set dataModeCEPValue as shown below:

```
<setting name="dataModeCEPValue"
         value="common/images/sbarleftblk.gif"/>
```

  **dataModeCEPAltValue**
  This setting provides alternate text for the image. It is used only when dataModeCEPRepresentation is set to **image** and the browser cannot

display the specified image, for example, the **Menu > View > Show Pictures** option of Internet Explorer Mobile is not selected.

**dataModeIconClass, dataModeIconStyle**
These are style and class overrides for the data mode representation icon. If not specified, the standard values for widgets apply. Standard HTML styles can be used, and classes are applied in order.

**cursorModeCEPRepresentation**
Determines the graphical representation used for the control shown next to the input field when in the cursor mode. Specify **button**, **link**, or **image**. The default is **link**.

**cursorModeCEPValue**
If cursorModeCEPRepresentation is either **button** or **link**, this value is used as text on the button or link. If no value is supplied, an asterisk character (*) is used by default. An empty string value"" is not accepted and defaults to an asterisk character (*).

If cursorModeCEPRepresentation is **image**, this value is the path to and the name of the image file to use. The path is searched relative to the Web Content directory of the project. For example, if you want to specify the sbarrightblk.gif in the Web Content/common/images directory, you set cursorModeCEPValue as shown below:

```
<setting name="cursorModeCEPValue"
         value="common/images/sbarrightblk.gif "/>
```

**cursorModeCEPAltValue**
This setting provides alternate text for the image. It is used only when **cursorModeCEPRepresentation** is set to **image** and the browser cannot display the specified image, for example, the **Menu > View > Show Pictures** option of Internet Explorer Mobile is not selected.

**cursorCEPRepresentationStyle**
This style is used to highlight the currently selected cursor position. The default is **border-bottom: 1px solid green;**. but an empty string "" is valid.

**cursorModeCEPStyle**
This is the style used around the input field while in cursor mode. The default style is **border: 1px solid #999999; height: 1.75em;**. It is not recommended to change this unless necessary.

**cursorCEPRepresentationClass, cursorModeIconClass, cursorModeIconStyle, cursorModeCEPClass, cursorModeCEPTextStyle, cursorModeCEPTextClass**
These are class and style overrides for decorating elements in the cursor mode area. If not specified, the standard values for widgets apply. Standard HTML styles can be used, and classes are applied in order.

**dataModeIconClass, dataModeIconStyle**
These are class and style overrides for decorating elements in the data mode area. If not specified, the standard values for widgets apply. Standard HTML styles can be used, and classes are applied in order.

- There are some instances during a wireless connection where the connection to the router or HTTP server is lost. If this occurs, the Web page may be locked, and HATS cannot resend any information. The following settings are available to resolve this issue. The settings are available in the RuntimeSettings class in the Project Settings source view (the application.hap file).

  **usePageUID**
  Specifies whether to enable a pageUID being put on the screen. This

allows the runtime to track the page to see if it has already been sent. If this value is false, the pageSubmitTimeout value is not read. The default value is **false**.

**pageSubmitTimeout**
Specifies how long to wait, in milliseconds, before unlocking the Web page for use by the user. A value of -1, or empty, results in this function being inactive. The default value is **-1**.

**incorrectPageUIDEvent**
Specifies what to do when the PageUID on the user's Web page HTTP request does not match the PageUID that is stored in the runtime for that sessionid and application instance. It has 2 options: REFRESHCOMMAND or ERROREVENT. The default is **REFRESHCOMMAND**.

The source below shows examples of these settings.

```
<class name="com.ibm.hats.common.RuntimeSettings">
   <setting name="autoEraseFields" value="true"/>
   <setting name="enableAutoAdvance" value="false"/>
   <setting name="enableAutoTabOn" value="false"/>
   <setting name="enableBusyPage" value="false"/>
   <setting name="enableCompression" value="false"/>
   <setting name="enableOverwriteMode" value="false"/>
   <setting name="incorrectPageUIDEvent" value="REFRESHCOMMAND"/>
   <setting name="pageSubmitTimeout" value="1000"/>
   <setting name="selectAllOnFocus" value="false"/>
   <setting name="suppressUnchangedData" value="false"/>
   <setting name="usePageUID" value="true"/>
</class>
```

- AJAX polling from a browser running on an iPhone or iPod touch device stops when you switch from the browser to another application. As a result, when you switch from a browser accessing HATS to another application on the device, HATS disconnects the browser session after the **Time to wait for disconnect (seconds)** interval has passed. You may wish to increase this interval or disable the auto-disconnect function for HATS applications accessed from iPhone or iPod touch devices. For more information see “Using the client pull (AJAX) method” on page 109.
- For DBCS considerations, see “Working with mobile device applications” on page 464.

In addition to options that HATS automatically disables, the following functions are not supported and should not be implemented in HATS applications for mobile devices:

- HATS Web projects cannot be migrated or automatically converted into HATS Web projects for mobile devices.
- Rich client applications.
- Portal applications (Standard or IBM).
- Interoperability with WebFacing.
- Bidirectional language support.
- Accessibility features.
- Any device with screen sizes smaller than 320x240.
- Keyboard host key support.

  As a result, using a keyboard works as a normal Web keyboard, host keys are not sent, instead the F1 key brings up the browser help, and the Enter key is not mapped to the host Enter key.

Also, without keyboard support, the HATS JavaScript cannot determine that the user has entered data on the HATS Web page. As a result, auto-refresh support overwrites any data that the user has entered into the GUI view of the HATS application if a new host screen is asynchronously received from the host application.

- Field specific help, right justify, capitalizes, or other attributes.
- Any other data entry limitations inherent to the device hardware or software.
- HATS administrative console page access by a mobile device.
- The disconnectOnClose connection parameter.
- Screen combinations.
- Calendar widget with Windows Mobile.
- Tabbed folder support.
- Spreadsheet support.
- Dojo widgets.

## Considerations and limitations for iPad devices

Your HATS application may appear or behave differently on iPad devices compared to other mobile devices because of differences such as screen size and web browser. Because the iPad device includes a large screen relative to some other mobile devices, it works better with applications that are not created with options optimized for mobile devices. When creating a HATS web project for an iPad device, do not select the **Optimize options for mobile devices** option.

For up-to-date support considerations, see "Host Access Transformation Services 9.7 - Known issues and workarounds" at http://www.ibm.com/support/docview.wss?uid=ibm10876092.

## Considerations and limitations for Android devices

Your HATS application may appear or behave differently on Android devices compared to other mobile devices because of differences such as screen size and web browser. If the screen size of your Android device is small, you might consider creating a HATS mobile project rather than a HATS web project for use with your device. This might provide a better user experience for a smaller screen. For more information, see “Developing HATS applications for mobile devices” on page 44.

For up-to-date support considerations, see "Host Access Transformation Services 9.7 - Known issues and workarounds" at http://www.ibm.com/support/docview.wss?uid=ibm10876092.

# Chapter 4. Developing and deploying HATS rich client applications

HATS gives you the ability to transform access to your existing host applications using rich client GUI applications that run in Eclipse Rich Client Platform (Eclipse RCP), Lotus Notes, or Lotus Expeditor Client environments.

The Eclipse RCP is a subset of plug-ins provided by the Eclipse platform, which is best known as an open source tooling platform. The introduction of Eclipse RCP enables you to use the core functionality provided by Eclipse to build native client applications targeted for a user's desktop. More information about the Eclipse RCP can be found at http://www.eclipse.org/home/categories/rcp.php.

Lotus Notes is powerful, multifaceted software that gives you instant access to all the information that is important to you. You can use Lotus Notes to send and receive Internet mail, schedule appointments, browse the Web, contribute to Internet newsgroups, and take advantage of the Home Page for tracking all your important daily information. For more information see the Lotus Notes Knowledge Center at http://www.ibm.com/support/knowledgecenter/SSKTMJ.

Lotus Expeditor Client provides a rich client runtime environment and integrated middleware components for extending many enterprise applications to server-managed laptop and desktop systems. For more information about developing and deploying applications for Lotus Expeditor Client, see the Lotus Expeditor Knowledge Center at http://www.ibm.com/support/knowledgecenter/SSVHEW.

**Note:**

Unless otherwise noted, the instructions documented here apply to developing and deploying HATS rich client applications in Eclipse RCP V3.7, Lotus Notes V8.5.3, and Lotus Expeditor Client V6.2.3 environments. For instructions that apply to earlier versions of Eclipse RCP and Lotus Expeditor Client, see earlier versions of the appropriate Knowledge Centers.

A rich client environment offers capabilities not available in a traditional Web environment, including:

- Improved response time
- A richer set of user interface (UI) widgets which provide for a more native application appearance, for example, native tab folder control, editable combo boxes, toolbars, menu bars, and tables
- No dependency on WebSphere Application Server or WebSphere Portal
- Client side processing (distributed, not centralized on a single server)
- Printing 3270E print jobs directly to a user's local printer

Additional information and help about developing rich client applications can be found in the Rational SDP Help Contents. Select **Help > Help Contents** on the menu bar and search for `rich client`.

When creating HATS rich client applications, the distinction between Eclipse RCP, Lotus Notes, and Lotus Expeditor Client environments is minimal. Differences are documented throughout this chapter.

The primary architectural difference between a HATS rich client application and a HATS Web application is in where the Telnet socket connection initiates. With a rich client application, the socket connection initiates from a user's workstation, not from the WebSphere Application Server, as it does with a traditional HATS Web application. Because rich client applications will require a client on a user's workstation, these applications are primarily targeted for internal users, who are accustomed to using a traditional, fat client terminal emulator.

## Developing HATS rich client applications

The steps for building and testing a HATS rich client application are essentially the same as those for a HATS Web application. You use HATS Toolkit with tools provided in the HATS perspective to develop a HATS rich client project from which a HATS rich client application is generated. The same wizards, editors and views are provided to support the various activities required during development.

As with HATS for the Web environment, you develop the appearance and operation of the application, including customizing default rendering settings, screen navigation (by way of skip screen macros), and the fonts and colors used to render screens.

The majority of features available in a HATS Web application are also available in a HATS rich client application. These include, but are not limited to:

- Default rendering
- Global rules
- Text replacement
- Custom screen transformations
- Prepopulated transformations
- Global variables
- Custom components and widgets
- Macros and macro handlers
- Background connections
- Java business logic
- Application-level events, for example start, stop, connect, and disconnect
- Keyboard support
- Asynchronous updating
- Local test environments analogous to Run on Server for a Web project

Features available in HATS Web applications that are not available in HATS rich client applications include:

- Creation of Integration Objects
- Creation of EJB access beans
- Web Express Logon (WEL)
- HATS administrative console

A HATS rich client project is an Eclipse plug-in project. As a project, it exists in a workspace and is developed by a HATS application developer. Once developed

and tested, the project is exported as a HATS rich client application, which is an Eclipse plug-in that can be installed into an Eclipse environment. An Eclipse plug-in is a component that provides a certain type of service within the context of an Eclipse environment. In the case with HATS, the plug-in provides the service required to transform a host application.

Each HATS rich client application plug-in depends on several runtime plug-ins. The dependencies are referred to in the rich client project's manifest file, MANIFEST.MF. You must ensure these required plug-ins are included in your local test environment and in the deployed production environment for your HATS rich client plug-in to run properly. For more information see "HATS RCP Runtime Extension project" on page 56, "Testing HATS rich client applications" on page 61, and "Deploying HATS rich client applications" on page 65.

# Target platform specifics

## Project contents

When you create a HATS rich client project, you are prompted to select a target platform. This selection indicates where you plan to deploy the application. The differences between a project targeted for deployment to Eclipse RCP and Lotus Notes or Lotus Expeditor Client are minor. The following characteristics are common to every HATS rich client project:

- Project structure including templates, transformations, images, Java source, macro handlers, macros, and screen captures folders
- HATS-specific project artifacts including the application.hap file, connection files, and event files
- The ComponentWidget.xml file, which is the component and widget registry file
- The plugin.xml file with one view, the transformation view, registered in it
- Plug-in dependencies on runtime plug-ins
- A Perspective class, included in the HATS RCP Runtime Extension project, responsible for laying out views on the page. See "HATS RCP Runtime Extension project" on page 56.

Addition for projects targeted for deployment to Eclipse RCP:

- An Application class, included in the HATS RCP Runtime Extension project, because Eclipse RCP requires an application to launch. See "HATS RCP Runtime Extension project" on page 56.

Additions for projects targeted for deployment to Lotus Notes and Lotus Expeditor Client:

- Registrations for the application switcher entry in the plugin.xml of the HATS RCP Runtime Extension project. This allows a user to launch the HATS application from the application switcher in the Lotus Notes and Lotus Expeditor Client environments.
- Key bindings for the Lotus Notes platform in the plugin.xml of the HATS RCP Runtime Extension project. This allows a user to have key mappings defined for use in the Lotus Notes environment.

## Setting the compiler compliance level

The Java runtime environment (JRE) of both your Rational SDP workspace and your intended runtime environment must be the same. For example, if you are developing a HATS rich client plug-in to run on an Eclipse environment that is

running a 1.6 Java runtime environment (JRE), you must set the compiler compliance level of your workspace to 1.6. To do this, from the Rational SDP menu bar, select **Window > Preferences > Java > Compiler** and set the **Compiler compliance level** to `1.6`.

If you have other Java or plug-in projects in your workspace that require other compliance levels, you can set the compiler compliance level for individual Java or plug-in projects. To do this, right-click on the project and select **Properties**. In the Properties panel select **Java Compiler** on the left. Then select the **Enable project specific settings** box and set the **Compiler compliance level** to the appropriate level.

**Note:** The HATS RCP Runtime Extension project, com.ibm.hats.rcp.runtime.extension, must be at the same compiler compliance level as your HATS rich client project. See "HATS RCP Runtime Extension project."

## HATS RCP Runtime Extension project

When you create a HATS rich client project, a project for the HATS RCP Runtime Extension plug-in (from here on referred to as the runtime extension plug-in), is automatically created if one does not already exist in the workspace. The runtime extension plug-in has a name and plug-in ID of com.ibm.hats.rcp.runtime.extension and an initial version of 1.0.0.

The runtime extension plug-in is an extension of the HATS RCP Runtime (com.ibm.hats.rcp.runtime) plug-in. The HATS RCP Runtime plug-in contains the HATS runtime classes, and it must be in the environment for HATS rich client applications to function. The runtime extension plug-in provides you the ability to configure settings related to the runtime environment.

This plug-in project contains the following configurable artifacts:

- The runtime.properties file – used for specifying log, trace, and other settings used by HATS in the runtime environment and in the local test environment running in Run mode. See "Administering HATS rich client applications" on page 76 for more information about these settings.
- The runtime-debug.properties file – used for specifying runtime settings used by HATS in the local test environment running in Debug mode. See "Administering HATS rich client applications" on page 76 for more information about these settings.
- The plugin.xml file – registers the Applications view with the environment as well as defines initial keyboard mappings
- The product.xml file – identifies the HATS version (same file as in HATS EAR projects)

This project is similar to a HATS EAR project in the Web environment, but there is only one of these plug-ins per runtime environment. That is, there is only one runtime extension plug-in running in the user's environment. Every HATS rich client project plugin.xml file declares a dependency on this plug-in. The runtime extension plug-in does not refer to individual HATS application plug-ins.

Because only one copy of the runtime extension plug-in can run in the runtime environment, you must ensure only one copy of the plug-in exists in a user's environment. If more than one runtime extension plug-in exists in the

environment, the plug-in with the highest version number is used by the environment. This ensures that the newest code is always used.

The runtime extension project is not visible in the **HATS Projects** view but is visible in the **Navigator** view. It is included as part of the feature project generated during export time. For more information see "Deploying HATS rich client applications" on page 65.

**Note:** This project is not deleted when the last HATS rich client project is deleted. You must manually delete it if you want a new runtime extension project to be created.

## Working with HATS rich client projects

As with a HATS Web project, when you create a new HATS rich client project, a set of folders is created to help you organize your HATS files. An example of a default project is shown below. The highest level folder has the same name as the name you give to your project when you create it. In that folder are other high-level folders that contain objects defined in your HATS project. Some folders do not appear until you create certain objects.

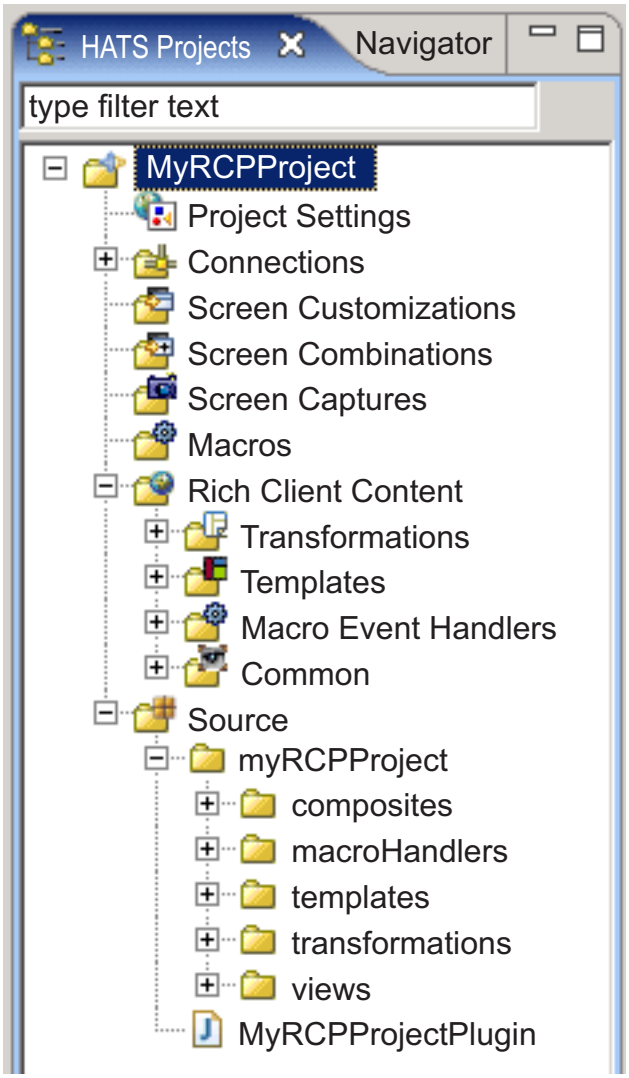



*Figure 6. HATS rich client project view*

Depending on how you set up your project, some or all of these folders appear in the **HATS Projects** view. You can also specify which folders appear in your **HATS Projects** view as well as hide file extensions. For more information, see "Using HATS preferences" on page 126.

**Notes:**

1. A rich client project displays a **Rich Client Content** folder, where a Web project displays a **Web Content** folder.
2. The **Rich Client Content** folder is displayed at the root level of the rich client project. However, there is no actual **Rich Client Content** folder in the project. One is dynamically displayed only to create consistency with the **Web Content** folder displayed for Web projects.
3. Some of the folders displayed under the dynamically displayed **Rich Client Content** folder are also displayed under the **Source** folder.
4. Label decorators for the dynamically displayed rich client folders may not decorate properly since the folders do not map to physical folders in the project.

You can create subfolders within these high-level folders to help organize your project. For instance, as you create screen captures for your project, you might want to create folders under the **Screen Captures** folder to organize and group the captured screens. To create a folder, right-click on one of the high-level folders in the tree and click **New HATS > Folder**. To move a file into a different folder, right-click on the file and select **Move**, or you can use the drag-and-drop method.

HATS projects can be shared in a team environment by going to the **Navigator View** of the HATS perspective. Right-click the project and select **Team > Share Project**. Select the repository type from the list and click **Next**. Rational SDP supports several repositories. For more information, refer to the Rational SDP documentation and search on `repository`.

**Note:** When using a version control system with your HATS projects, set the system to ignore the `resourceUpdate.sts` file. This file is automatically generated when testing a project within the toolkit. The file should not be under version control and can safely be ignored or deleted before placing a HATS project under version control.

To exclude the file from version control, open **Window -> Preferences -> Team -> Ignored Resources** and click **Add Pattern** to add a new pattern. Enter `resourceUpdate.sts` and click **OK**. Make sure the new pattern is selected in the list of ignored patterns, and click **OK** to save the settings.

## Copying resources between Web and rich client projects

You can copy, paste, delete, move, and rename rich client resources using the pop-up menu in the HATS Projects view. The following resources can be copied or moved between HATS Web and HATS rich client projects:

- Connection profiles
- Macros
- Screen captures
- Screen customizations
- Images
- Java source files
- BMS files
- Host simulation files

**Note:** You are not allowed to copy transformations, transformation fragments, or templates between HATS Web and HATS rich client projects.

Some resources copied from projects of different types will need modification. For example, a macro copied from a HATS Web project to a HATS rich client project will need to be updated to refer to a valid SWT Java macro handler (for HATS Web projects, macros refer to a JSP macro handler). Screen customizations will also need to be updated because of references to JSP transformations specified in the apply action and because of URLs specified in the forward action. A warning message will appear in the **Problems** view to help you find and fix these problems. The following scenarios will be flagged:

- Screen customization action referencing an incorrect artifact (typically an apply action referencing a transformation that does not exist in the project).
- Macro referencing an incorrect macro handler.
- Forward or block actions in a rich client screen customization.

There is no built-in function to migrate a HATS Web application to a HATS RCP application. Some artifacts, such as screen customizations, macros, connections, screen captures, images, business logic source files, and custom component source files can be manually migrated, with minimal changes, from one project type to another. Other artifacts, such as transformations, custom widgets, templates, and custom macro handlers require more extensive changes to work in a project of a different type.

## Exporting and importing HATS rich client projects

To move a HATS rich client project to another system with Rational SDP installed, or to save and restore a project for any other reason, you can use the export and import capabilities of Rational SDP.

### Export

Export a project and its dependent projects as a zipped file using the Rational SDP archive file function. This function has multiple benefits. It allows you to export multiple projects and the required runtime extension project (com.ibm.hats.rcp.runtime.extension) all at once. In addition, you do not have to create a project on the destination HATS Toolkit before you import the archive file. To export the project(s) from HATS Toolkit as an archive file:

1. From the menu bar click **File > Export** to open the Export wizard.
2. Select **General > Archive File** and click **Next**.
3. Select the project(s) to export.

   **Note:** A HATS rich client project requires the runtime extension project (com.ibm.hats.rcp.runtime.extension) in order to run. Therefore, you must either export the runtime extension project as part of the archive file, or create a new runtime extension project on the destination system by creating a new HATS rich client project.
4. Specify a file name and location where you want to save the project(s).
5. In the **Options** section, select the desired file format and compression options. To export the whole project, select **Create directory structure for files**.
6. Click **Finish**.

### Import

To import an archive file (or a project interchange file from an older release of Rational SDP) into HATS Toolkit:

1. From the menu bar click **File > Import** to open the Import wizard.
2. Select **General > Existing Projects into Workspace** and click **Next**.
3. Select the **Select archive file** option and click **Browse** to browse for the archive file.
4. In the **Projects** section, select the project (or projects) you want to import.

   **Note:** A HATS rich client project requires the runtime extension project (com.ibm.hats.rcp.runtime.extension) in order to run. Therefore, you must either import the runtime extension project from the archive file, or create a new runtime extension project by creating a new HATS rich client project.
5. Click **Finish**.

# Testing HATS rich client applications

To test a HATS rich client application plug-in in a local test environment, you launch the plug-in, and required runtime plug-ins, in an instance of the target platform.

**Note:** In addition to testing in the local test environment, you should export your project as an application and deploy it in a runtime environment to test out the deployment process. For more information see "Deploying HATS rich client applications" on page 65.

## Configuring the target platform

The target platform refers to the Eclipse product (for example, Eclipse RCP, Lotus Notes, or Lotus Expeditor Client) against which the plug-in you are developing will be compiled and tested.

By default the target platform is Rational SDP itself, therefore no configuration is necessary for Eclipse RCP.

To configure the target platform for Lotus Notes (or Lotus Expeditor), perform the following steps.

1. Install Lotus Notes (or Lotus Expeditor) on the same development workstation as Rational SDP.
2. From the Rational SDP menu bar select **Window > Preferences > Plug-in Development > Target Platform** and click **Add**.
3. Select **Nothing: Start with an empty target definition** and click **Next**.
4. On the Target Content page, enter a target name, for example, `Lotus Notes`, or `Lotus Expeditor`, and click **Add**.
5. On the Add Content page, select **Directory** and click **Next**.
6. The target definition you are creating consists of Lotus Notes (or Lotus Expeditor) plug-ins, shared plug-ins, and HATS runtime plug-ins. On the Add Directory page do the following:
   a. Click **Browse** and browse for the Lotus Notes (or Lotus Expeditor) base plug-ins. For Lotus Notes browse to, for example, `C:\Program Files\IBM\Lotus\Notes\framework\rcp\eclipse`. For Lotus Expeditor browse to, for example, `C:\Program Files\IBM\Lotus\Expeditor\rcp\eclipse`. Click **Finish**.
   b. On the Target Content page, click **Add**.
   c. On the Add Content page, select **Directory** and click **Next**.
   d. Click **Browse** and browse for the Lotus Notes (or Lotus Expeditor) shared plug-ins. For Lotus Notes browse to, for example, `C:\Program Files\IBM\Lotus\Notes\framework\shared\eclipse`. For Lotus Expeditor browse to, for example, `C:\Program Files\IBM\Lotus\Expeditor\shared\eclipse`. Click **Finish**.
   e. The HATS runtime plug-ins cannot be added here. The HATS Toolkit copies the runtime plug-ins when you change the target platform and the plug-ins are not already installed in the target platform. This occurs in a later step.
7. On the Target Content page, click **Finish**.
8. On the Preferences - Target Platform page, select the box for the newly created target definition, ensure it is designated as `Active`, and click **OK**.

9. The HATS runtime plug-ins are now installed into the target platform and you are asked to reload the target platform. See "Installing runtime plug-ins" for how to reload the target platform. Click **OK**.

## Installing runtime plug-ins

In order for a HATS rich client application plug-in to run in the target platform's local test environment, the required runtime plug-ins must be installed in the target platform.

The Eclipse RCP target platform is Rational SDP itself. Because the HATS Toolkit is installed in Rational SDP, the HATS runtime plug-ins are already installed for the Eclipse RCP target platform. The remainder of this section pertains to installing runtime plug-ins when using the Lotus Notes (or Lotus Expeditor) target platform.

When you create a new project and select the Lotus Notes (or Lotus Expeditor) target platform, or later change the target platform, the HATS Toolkit will detect if the required runtime plug-ins are installed and at the latest level in the selected target platform. If not, the toolkit will prompt you to have the latest runtime plug-ins installed. You can enable or disable this prompt using HATS Preferences. See "Using HATS preferences" on page 126 for more information. Following are the required runtime plug-ins that are installed:

- HATS Core (com.ibm.hats.core)
- HATS RCP Runtime (com.ibm.hats.rcp.runtime)
- HATS RCP UI (com.ibm.hats.rcp.ui)
- HATS RCP Documentation (com.ibm.hats.rcp.doc)
- Host Simulation (com.ibm.hostsim)
- Host On-Demand beans (com.ibm.eNetwork.beans.HOD)
- Host Access SSLite (com.ibm.etools.hasslite)
- Terminal Beans (com.ibm.etools.terminal.beans)
- SWT Bidirectional extension (com.ibm.editors.swtbidiextension)
- Host Screen Rendering (com.ibm.hsrendering)
- WebFacing Common (com.ibm.etools.iseries.wfcommon)
- EMF Service Data Objects (SDO) (org.eclipse.emf.ecore.sdo)
- CommonJ SDO (org.eclipse.emf.commonj.sdo)

After you configure a new target platform for Lotus Notes (or Lotus Expeditor) and install the latest HATS runtime plug-ins in the target platform, see "Configuring the target platform" on page 61, you must reload the new plug-ins. To do this, perform the following steps .

1. From the Rational SDP menu bar select **Window > Preferences > Plug-in Development > Target Platform**.
2. On the Preferences - Target Platform page, select the Lotus Notes (or Lotus Expeditor) target definition you configured. Click **Edit**.
3. On the Target Content page, click **Add**.
4. On the Add Content page, select **Directory** and click **Next**.
5. On the Add Directory page, browse to the HATS location in the appropriate path. For Lotus Notes browse to, for example, `C:\Program Files\IBM\Lotus\Notes\framework\rcp\hats\eclipse`. For Lotus Expeditor browse to, for example, `C:\Program Files\IBM\Lotus\Expeditor\rcp\hats\eclipse`.
6. Click **Finish** twice to close the wizard.

7. On the Preferences - Target Platform page, click **Reload** then click **OK**.

## Setting the default JRE

In addition to configuring the target platform and installing the runtime plug-ins, you may need to set the default JRE for your workspace in Rational SDP.

### Eclipse RCP

When testing on an Eclipse RCP target platform, set the default JRE to **jdk**. To do this perform the following steps:

1. From the Rational SDP menu bar select **Window > Preferences > Java > Installed JREs**.
2. In the list of installed JREs, select the box next to **jdk**.
3. Click **OK**.

Ensure your JRE's match. For example, if you are testing on an Eclipse environment that will be running Java 1.6, you should update the JRE of the workspace to match the JRE of the runtime environment.

### Lotus Notes

When testing on a Lotus Notes target platform, perform the following steps:

1. From the Rational SDP menu bar select **Window > Preferences > Java > Installed JREs** and click **Add**.
2. On the JRE Type page, select **Standard VM** and click **Next**.
3. On the JRE Definition page, click **Directory** and browse to the Lotus Notes JRE location, for example, `C:\Program Files\IBM\Lotus\Notes\jvm`. Click **Finish**.
4. On the Preferences - Installed JREs page, select the newly added JRE and click **OK**.

### Lotus Expeditor Client

When testing on a Lotus Expeditor target platform, perform the following steps:

1. From the Rational SDP menu bar select **Window > Preferences > Java > Installed JREs** and click **Add**.
2. On the JRE Type page, select **Standard VM** and click **Next**.
3. On the JRE Definition page, click **Directory** and browse to the Lotus Expeditor Client JRE location, for example, `C:\Program Files\IBM\Lotus\Expeditor\rcp\eclispe\plugins\com.ibm.rcp.j2se.win32.x86_1.6.0.20101125a-201012221418\jre`. Click **Finish**.
4. On the Preferences - Installed JREs page, select the box for the newly added JRE and click **OK**.

## Launching your project

To launch your project, a launch configuration is required. HATS creates a default launch configuration for you when you click either **Run** or **Debug** for your project (see the instructions in the sections below). A launch configuration is not created when the Run or Debug dialog is opened from another view, for example, the Package Explorer view. The default launch configuration is based on the target platform of the project, which is set in the New Project wizard or in the Project Settings editor. If you modify the default launch configuration to the point where it can no longer launch a runtime workbench, you can delete the launch configuration, and HATS will re-create it when **Run** or **Debug** is clicked again. To launch your project, follow the instructions in the sections below for your target platform.

## Eclipse RCP

To launch your project in the local test environment for Eclipse RCP, perform the following steps:

1. From the HATS Projects view, right-click on the project and select either **Run** or **Debug**.
2. If you selected **Debug**, respond **Yes**, **No**, or **Cancel** to the Display Terminal dialog to specify whether you want to see the host display terminal while testing your application.
3. In the Run (or Debug) Configurations dialog, HATS automatically creates and configures a default launch configuration named hostaccess.
4. On the **Main** tab, in the **Program to Run** section, be sure **com.ibm.hats.rcp.runtime.extension.product** appears in the **Run a product** field.
5. Click **Run** (or **Debug**) to launch the default configuration.
6. A new instance of the Eclipse workbench is launched.
7. The Host Access perspective is automatically opened.
8. Your HATS application is listed in the **Applications** view. To start your application, either double-click on it or right-click and select **Open**.
9. A new instance of the selected application is opened and its transformation view is displayed.

## Lotus Notes

Lotus Notes must be installed on the same development workstation as Rational SDP in order to locally test a HATS plug-in targeted for Lotus Notes. Lotus Notes must not already be running when you attempt to test your project because only one instance of Lotus Notes can be running at any given time. See “Configuring the target platform” on page 61.

To launch your project in the local test environment for Lotus Notes, perform the following steps:

1. From the HATS Projects view, right-click on the project and select either **Run** or **Debug**.
2. If you selected **Debug** , respond **Yes**, **No**, or **Cancel** to the Display Terminal dialog to specify whether you want to see the host display terminal while testing your application.
3. In the Run (or Debug) Configurations dialog, HATS automatically creates and configures a default launch configuration named hostaccess.
4. On the **Main** tab, in the **Program to Run** section, be sure **com.ibm.notes.branding.notes** appears in the **Run a product** field.
5. On the **Main** tab, in the **Java Runtime Environment** section, confirm that **Notes 8.5.1 VM** is listed in the **Runtime JRE** field. For more information, see “Lotus Notes” on page 63.
6. Click **Run** (or **Debug**) to launch the configuration.
7. A new instance of Lotus Notes is launched.
8. Click **Open** on the main toolbar and select **Host Access**.
9. Your HATS application is listed in the **Applications** view. To start your application, either double-click on it or right-click and select **Open**.
10. A new instance of the selected application is opened and its transformation view is displayed

### Lotus Expeditor Client

Lotus Expeditor Client for Desktop must be installed on the same development workstation as Rational SDP in order to locally test a HATS plug-in targeted for Lotus Expeditor. See "Configuring the target platform" on page 61.

To launch your project in the local test environment for Lotus Expeditor Client, perform the following steps:

1. From the HATS Projects view, right-click on the project and select either **Run** or **Debug**
2. If you selected **Debug**, respond **Yes**, **No**, or **Cancel** to the Display Terminal dialog to specify whether you want to see the host display terminal while testing your application.
3. In the Run (or Debug) Configurations dialog, HATS automatically creates and configures a default launch configuration named hostaccess.
4. On the **Main** tab, in the **Program to Run** section, be sure **com.ibm.rcp.platform.personality.branding.DefaultProduct** appears in the **Run a product** field.
5. On the **Main** tab, in the **Java Runtime Environment** section, confirm that **Java SE 6.0 Win32 x86** is listed in the **Runtime JRE** field. For more information, see "Lotus Expeditor Client" on page 63.
6. Click **Run** (or **Debug**) to launch the configuration.
7. A new instance of Lotus Expeditor Client is launched. You may be prompted to enter a new password the first time the client starts.
8. Click **Open** on the main toolbar and select **Host Access**.
9. Your HATS application is listed in the **Applications** view. To start your application, either double-click on it or right-click and select **Open**.
10. A new instance of the selected application is opened and its transformation view is displayed.

## Deploying HATS rich client applications

Regardless of where you obtain the HATS package (HATS CD, Web, or packaged with another product), you install the same version of the HATS Toolkit. This is a limited-use version that you can use to evaluate HATS. To fully enable the runtimes for production in accordance with your licensed proof of entitlement, you must specify your license settings using the License Settings wizard included in the HATS Toolkit. For more information see the section, Enabling HATS runtime and license settings, in *HATS Getting Started*.

This section describes two different methods for deploying HATS rich client applications. The first method can be used to package a complete Eclipse client environment for distribution that has everything necessary to run your HATS rich client application. The second method can be used to package your HATS rich client application for clients that already have an Eclipse environment installed.

## Packaging an Eclipse client environment for distribution

The Eclipse SDK, on which the Rational SDP is based, provides the ability to generate a complete Eclipse client which includes your HATS application plug-ins and all other required plug-ins. This client can be distributed to individual client systems and run as a stand-alone application.

**Note:** Prior to following these instructions, it is assumed that a HATS rich client plug-in project has been created and run in the local test environment.

To create the client package, perform the following steps:

1. In the **HATS Projects** view, right-click your HATS rich client project and from the pop-up menu select **New HATS > Product Configuration**. This launches the New Product Configuration wizard. The wizard can also be launched from the Rational SDP menu bar by selecting **File > New > Other > Plug-in Development > Product Configuration > Next**.
2. Select the **com.ibm.hats.rcp.runtime.extension** plug-in project folder as the parent folder.
3. Enter `hostaccess.product` in the File name field.
4. Select the **Use a launch configuration** radio button and select **hostaccess**.

   **Note:** In this example, hostaccess is the default name used for the Run, or Debug, configuration you used when you tested your application in the local test environment.
5. Click **Finish**.
6. Click the **Overview** tab and click the **Eclipse Product export wizard** link.
7. Under Destination, in the Directory field, enter the name of the directory to export the Eclipse product to. For example `c:\hostaccess`. Alternatively, you can select to export to an archive (.zip) file which is useful when you want to make the client package available as a one file download.
8. Under Export Options, clear **Generate metadata repository**.
9. Click **Finish**.

A complete Eclipse client environment is created. To test your application, use Windows Explorer to navigate to the c:\hostaccess directory. From the eclipse folder double-click **eclipse.exe**. This directory can be zipped up and downloaded to individual client machines. All that is required on the client system is a compatible Java Runtime Environment (JRE).

## Packaging for existing Eclipse clients (Eclipse RCP, Lotus Notes, or Lotus Expeditor Client)

To deploy your HATS rich client application in an existing Eclipse client runtime environment you must:

- Export your project as an Eclipse feature.
- Export the HATS runtime features.
- Create an update site.
- Install your application in a runtime environment.

The following sections describe how to perform each of these tasks.

### Exporting your project as an Eclipse feature

To deploy your HATS rich client application plug-in to a runtime environment, you must first package it into an Eclipse feature. An Eclipse feature is a deployment artifact used to package associated plug-ins. To package your plug-in into an Eclipse feature, perform the following steps.

1. Create an empty directory, for example c:\myHATS, on your local hard drive to store your exported HATS application feature. You will use this same directory to store the HATS runtime features and the site.xml file that contains the definition for an update site. For more information, see “Exporting HATS runtime features” on page 68 and “Creating an update site” on page 69.
2. From the **HATS Projects** view, right-click your rich client project and from the pop-up menu select **Export > Export Feature.**

**Note:**

If you have not specified your license settings, you will see a message that this application is not runtime enabled and cannot be run in a production environment.

HATS rich client applications can be tested in a local test environment (Eclipse, Lotus Notes, or Lotus Expeditor Client), but cannot be run in a runtime (non-development) environment without first specifying license settings. This means that you can create and fully test an application in the HATS Toolkit development environment. You can also export the application, and a user can install it, but the user will not be able to connect with the application. That is, the runtime will be locked down and will not allow for any connections. For information about specifying license settings, see Enabling HATS runtime and license settings in *HATS Getting Started*.

If licenses have been purchased, click **Enable Runtime** to runtime enable your application. Otherwise, click **Continue** to continue the export process.

3. Assuming your project is not already part of a feature, a dialog will prompt you to create the feature now. Click **Yes** to launch the Create a HATS Feature Project wizard.
4. In the Create a HATS Feature Project wizard, complete the **Feature ID**, **Name**, **Version**, and **Provider** fields.

   **Note:** The **Name** field is automatically filled in as you fill in the **Feature ID** field. Validation is performed on the **Feature ID** and **Name** fields as you fill them in. The value you enter for **Feature ID** is also the name used for the feature project that is created, and all values are stored in the project's feature.xml file. HATS supplies the name, `hostaccess`, as a default for the **Feature ID** and **Name** fields and the version, `1.0.0`, as a default for the **Version** field .
5. The plug-in list is populated from all HATS rich client plug-in projects in the workspace. Select the plug-ins to include in the new feature. These plug-ins will be installed into the client when the feature is installed. Click **Finish**.
6. Next, the Eclipse Export wizard appears. In the Deployable features panel, select your feature.
7. On the **Destination** tab, select **Directory** and click **Browse** to locate and select your export directory, for example c:\myHATS.
8. On the **Options** tab, ensure **Package as individual JAR archives (required for JNLP and update sites)** is selected. The other options can remain at their default selections.
9. Click **Finish**.

Your exported HATS rich client application feature automatically includes the runtime extension plug-in and automatically includes a dependency on the HATS runtime features. The feature project is not displayed in the HATS Projects view, but is displayed in the Navigator view.

If you want to perform more advanced configuration of your HATS rich client application feature, instead of using the Create a HATS Feature Project wizard, you can use the standard Eclipse New Feature wizard provided by Eclipse. To launch this wizard, from the Rational SDP toolbar select **File > New > Other > Plug-in**

**Development > Feature Project**. This wizard performs basically the same functions as the Create a HATS Feature Project wizard except it does not automatically include the runtime extension plug-in, and it does not include the necessary dependencies on the HATS runtime features.

## Exporting HATS runtime features

In a production environment, a HATS rich client application feature depends on the HATS runtime features being installed on the client. When you create a HATS rich client application feature using the Create a HATS Feature Project wizard, it automatically includes a dependency on the HATS runtime features. This dependency is what causes the feature install and update process to automatically pull down the HATS runtime features from the update site when a HATS rich client application feature is installed or updated. The HATS runtime features include the following features and plug-ins:

- Feature: HATS Core (com.ibm.hats.core)
  - Plug-ins:
    - com.ibm.hats.core
    - com.ibm.hats.rcp.runtime
    - com.ibm.hats.rcp.ui
    - com.ibm.rcp.doc
    - com.ibm.hostsim
- Feature: SWT Terminal (com.ibm.etools.terminal.beans)
  - Plug-ins:
    - com.ibm.eNetwork.beans.HOD
    - com.ibm.etools.hasslite
    - com.ibm.etools.terminal.beans
    - com.ibm.editors.swtbidiextension
- Feature: Host Screen Rendering (com.ibm.hsrendering)
  - Plug-ins:
    - com.ibm.hsrendering
- Feature: WebFacing Common (com.ibm.iseries.wfcommon)
  - Plug-ins:
    - com.ibm.etools.iseries.wfcommon
- Feature: EMF Service Data Objects (SDO) (org.eclipse.emf.ecore.sdo)
  - Plug-ins:
    - org.eclipse.emf.ecore.sdo
    - org.eclipse.emf.commonj.sdo

To export the HATS runtime features, perform the following steps:

1. In the **HATS Projects** view, right-click your HATS rich client project and from the pop-up menu select **Export > Export HATS Runtime Features**. This launches the Export HATS Runtime Features wizard. The wizard can also be launched from the Rational SDP menu bar by selecting **File > Export > HATS > HATS Runtime Features**.
2. On the Export HATS Runtime Features panel, for **Deploy as** select **Directory** and click **Browse** to locate and select your export directory, for example c:\myHATS.

   **Note:** The Export HATS Runtime Features wizard allows you to specify either an archive file or a directory on the local file system for saving the HATS

runtime features. If you choose to export to an **Archive** file, all of the features and plug-ins are packaged into a single .zip archive. The root directory structure inside the .zip archive starts with two subdirectories, one named, features, and the other named, plugins. If you choose to export to a **Directory**, features and plug-ins are exported to the selected directory. The features are placed in a subdirectory named, features, and the plug-ins are placed in a subdirectory named, plugins. Select **Directory** if you are planning to upload the archives to an update site directory on a Web server.

3. Click **Finish**.

## Creating an update site

After exporting both your HATS application feature and the HATS runtime features, you must create an update site that can be used by the feature install and update process to deploy your application. To create an update site, perform the following steps:

1. In the **HATS Projects** view, right-click your HATS rich client project and from the pop-up menu select **New HATS > Update Site Project**. This launches the New Update Site wizard. The wizard can also be launched from the Rational SDP menu bar by selecting **File > New > Other > Plug-in Development > Update Site Project > Next**.
2. On the Update Site Project panel:
   a. Give your project a name, for example myHATSUpdateSite.
   b. Clear the **Use default location** check box and click **Browse** to locate and select the directory to where you exported your HATS application feature and the HATS runtime features, for example c:\myHATS.
   c. Click **Finish**.
3. In the site.xml editor view on the **Site Map** tab, click **New Category**.
4. In the Category Properties section:
   a. Enter a Name, for example, `HATS`.
   b. Enter a Label, for example, `HATS`.
   c. Optionally enter a Description.
   d. Click the **Save** icon on the Rational SDP toolbar.
5. In the site.xml editor view on the **Site Map** tab, click **Add Feature**.
6. In the Feature Selection panel, scroll down and select your HATS application feature, for example `hostaccess (nnnn)`, where `nnnn` represents the version number of the feature, and click **OK**. For more information about version numbers, see "Updating plug-in and feature version numbers" on page 76.

   **Note:** If you begin typing the name of a feature in the Select a feature field, the data you enter acts as a wildcard, and the list of features in the list box narrows until the desired feature is showing.
7. Click **Add Feature** to add each of the HATS runtime features:
   - com.ibm.etools.terminal.beans (nnnn)
   - com.ibm.hats.core (nnnn)
   - com.ibm.hsrendering (nnnn)
   - com.ibm.iseries.wfcommon (nnnn)
   - org.eclipse.emf.ecore.sdo (nnnn)
8. Click **Build All** to complete building the update site site.xml file.

**Note:** If an error message is displayed stating that `Compilation errors occurred during the build. Logs can be found in 'logs.zip' at the root of the site project.`, then check the logs.zip file. If it contains only warnings, you can safely ignore them and continue.

You have now created an update site, in this example c:\myHATS, including the site.xml file that indicates what features, and versions of those features, are available from the site for download by a client. You can now move this update site directory to a Web server that can be accessed by clients. For instructions on how to define an update site on the client system and install your HATS rich client applications in a runtime environment, see "Installing your application in a runtime environment."

**Note:** After installing a HATS service update, you should re-export and redeploy the HATS runtime features to pick up any changes. You must modify the update site's site.xml file to reflect any version number changes. The versions are listed on the first panel of the Export HATS Runtime Features wizard. For more information see "Updating plug-in and feature version numbers" on page 76.

## Installing your application in a runtime environment

HATS rich client applications can run in Eclipse RCP, Lotus Notes, and Lotus Expeditor Client environments. The following sections describe how to install your HATS rich client application in each of these environments.

**Eclipse RCP:**

To download and install your HATS rich client application in an Eclipse rich client environment, the client environment must be configured to access the update site containing the application. By default, since automatic install and update is disabled, the user or administrator must manually install the new features or updates using the standard Eclipse Software Update functions. To do this perform the following steps on the client system.

**Note:** The following instructions are for Eclipse 3.6 and later. If you are using an earlier version of Eclipse, follow the appropriate Eclipse documentation to install the features.

1. From the Eclipse client menu bar, select **Help > Install New Software**.

   **Note:** Depending on your Eclipse client configuration, this menu item may be located under a different menu.
2. Click the **Add** button and then the **Local** button.
3. Browse to the location of the update site you created for the HATS project and click **OK**.
4. Select your HATS update site. If you gave the update site a category named `HATS`, select the check box beside **HATS** and make sure all the check boxes underneath it are also selected, or you can click the **Select All** button. If you don't see your update site category listed, make sure to have the **Group items by category** check box selected. Click **Next**.
5. Review and confirm the items being installed. Click **Next**.
6. Select **I accept the terms in the license agreement** and click **Finish**.
7. If you see a security warning that the software being installed contains unsigned content, click **OK**.
8. Your HATS application is installed.

9. When prompted to restart the workbench, click **Yes**.
10. You can then run your application by first opening the Host Access perspective (**Window->Open Perspective-> Host Access**). Your HATS application is listed in the Applications view. To start your application, either double-click it or right-click and select **Open**.

For more information about the Eclipse RCP environment see http://www.eclipse.org/home/categories/rcp.php.

**Lotus Notes:**

In the Lotus Notes environment, a widget is used to install applications that are provided as features and plug-ins. The widget can later be published to a Widget catalog for others to use. For more information about widgets, see the Widgets and Live Text section in the Lotus Notes Knowledge Center at http://publib.boulder.ibm.com/infocenter/domhelp/v8r0/index.jsp?topic=/com.ibm.notes85.help.doc/wid_app_overview_c.html .

To download and install your HATS rich client application in a Lotus Notes environment, perform the following steps on the client system:

1. Start the Lotus Notes client and select **File > Preferences > Widgets**.
2. Select **Show Widget Toolbar and the My Widgets Sidebar panel**. Click **OK**.

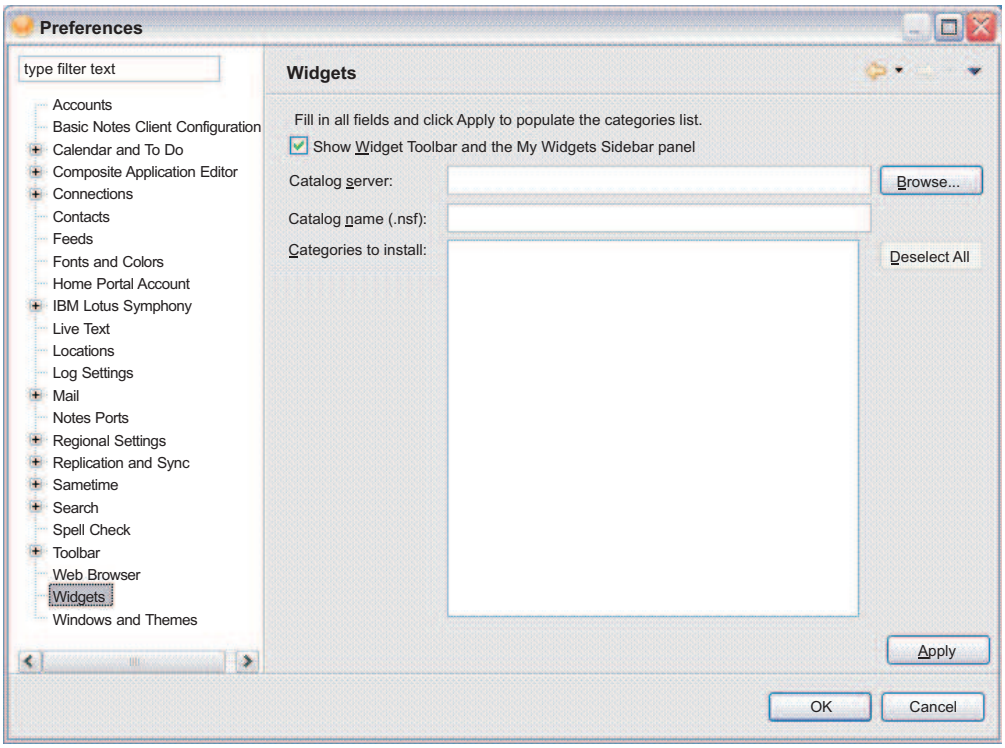



*Figure 7. Lotus Notes Widgets preferences*

3. The Widget toolbar looks like this:

*Figure 8. Lotus Notes Widget toolbar*

4. The My Widgets Sidebar panel looks like this:

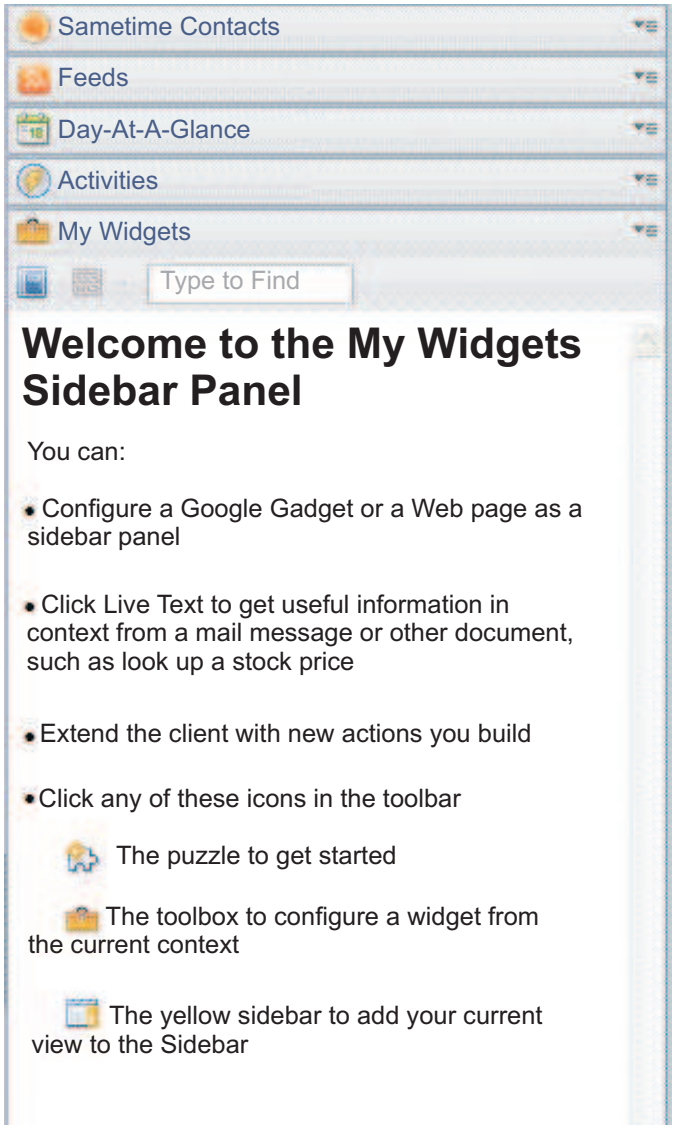



*Figure 9. My Widgets Sidebar*

5. Select **Getting Started with Widgets** from the Widget toolbar.
6. Select **Features and Plugins**, then select **Next**.
7. In the **Enter the URL** field, enter the path to the update site's site.xml file using this format as an example (must start with `file:///`): `file:///c:\hatsupdatesite\site.xml`. Click the **Load** button.
8. Under **Features to Install**, select all features and select **Next**.

9. Optional: In the **Widget name** field, you can enter `HATS Provisioning Widget` or another appropriate name.
10. Optional: If you want the widget to have an image associated with it, copy the desired image to your update site folder. Then in the **Image URL** field, enter the path to the image using this format as an example (must start with `file:///`): `file:///c:\hatsupdatesite\myhatsimage.png`.
11. Choose to install the plug-ins when asked.

    The My Widgets Sidebar panel will look similar to this:


*Figure 10. My Widgets Sidebar with example HATS Provisioning Widget*

12. Restart Lotus Notes when asked.

You can run your application by selecting **Open > Host Access**. Your HATS application is listed in the **Applications** view. To start your application, either double-click it or right-click and select **Open**.

To uninstall the HATS application, right-click the HATS Provisioning Widget (or whatever name you gave it) and select **Remove**. Restart Lotus Notes when asked.

**Lotus Expeditor Client:**

In the Lotus Expeditor Client environment application management and deployment can be controlled in a number of ways:

- Lotus Expeditor Server, which contains the Device Manager Server component
- WebSphere Portal, which allows for centrally administered, role-based access to applications
- Eclipse update site, which provides provisioning and updating using a standard Web server

For the Lotus Expeditor Client, the following features must be added to the default installation:

- Lotus Expeditor Desktop Development Utilities
- Core JVM Feature - J2SE (only if using Lotus Expeditor Client version 6.2.0 or earlier)

From the Lotus Expeditor Client, select **File > Application > Install > Search for new features to install > Add folder location**. Browse to the following folder in the Lotus Expeditor Client product files: desktop\updates\platform and select **Finish**. Expand **updates/platform** and **Development**. Put a check mark beside **Lotus Expeditor Desktop Development Utilities**. Select **Next** and complete the wizard.

For Lotus Expeditor Client versions 6.2.0 and earlier, to install the Core JVM Feature - J2SE, from the Lotus Expeditor Client, select **File > Application > Install > Search for new features to install > Add folder location**. Browse to the Desktop Runtime Environment site location, which is the update.site.dre.client directory and select **OK**. Accept the default site name and select **OK** and **Finish**. Expand the site and **Runtime Components**. Put a check mark beside **Core JVM Feature - J2SE**. Select **Next** and finish the wizard.

For more information, see the Assembling and Deploying Lotus Expeditor Applications section in the Lotus Expeditor Knowledge Center at http://publib.boulder.ibm.com/infocenter/ledoc/v6r2/index.jsp.

To download and install your HATS rich client application in a Lotus Expeditor Client environment, the client must be configured to access the update site containing the application. To do this perform the following steps on the client system.

1. From the Lotus Expeditor Client main menu select **File > Application > Install**.
2. On the Feature Updates panel, select **Search for new features to install** and click **Next**.
3. On the Application Locations panel, click **Add Remote Location** or **Add Folder Location**, as appropriate.
   - For Add Remote Location, on the New Update Site panel, enter the name of the update site and the URL for the Web server hosting the update site and click **OK**.
   - For Add Folder Location, on the Browse For Folder panel, browse to the folder containing the update site and click **OK**.
4. On the Application Locations panel, select your HATS update site and click **Finish**.
5. On the Search Results panel, select the features to install and click **Next**.
6. On the Feature License panel, select **I accept the terms in the license agreement** and click **Next**.
7. On the Installation panel, click **Finish**.
8. Choose to install the plug-ins when asked.
9. Your HATS application is installed.
10. When prompted to restart the workbench, click **Yes**.
11. You can then run your application by first opening Host Access (**Open->Host Access**). Your HATS application is listed in the Applications view. To start your application, either double-click it or right-click and select **Open**.

**Automatic update process:**

The rich client environment allows for the automatic update of installed applications. Preferences are available for controlling how often the server is checked for updates. Once you upload an update to an application, the rich client will download and install the update. Version checking is performed to make sure only new updates are downloaded and installed. In most cases, the user will need to restart the environment to activate the updated application. For more information, see the appropriate Eclipse RCP, Lotus Notes, or Lotus Expeditor documentation.

## Updating plug-in and feature version numbers

Before you can update a plug-in that is already installed, you must update the plug-in's version because Eclipse uses the version to determine if a plug-in has changed. For example, if you add a screen customization or modify a macro in your project, you must update your plug-in's version before re-exporting it.

To update your plug-in's version, in the Project Settings editor for your project, on the **Overview** tab, click the **open the plug-in manifest** link. On the **Overview** tab, under General Information, update the version. An Eclipse version is in the form `major.minor.service`. When making a minor change to the project, update the `service` segment of the version. For example, before exporting your plug-in a second time, set your plug-in's version to `1.0.1`.

Because the License Settings wizard updates the runtime extension project (com.ibm.hats.rcp.runtime.extension), after running the wizard you must update the version of this plug-in. To do this, in the Navigator view, open the plugin.xml file located at the root of the project. On the **Overview** tab, under General Information, update the version.

Also, because features are made up of plug-ins (and features also have versions), you must update the version of your feature whenever you update the version of one of the included plug-ins. To do this, switch to the Navigator view and open the feature.xml file located in your feature project. By default the name of your feature project is `hostaccess`. On the **Overview** tab, under General Information, update the version. In addition, ensure that your feature correctly lists the new versions of your plug-ins by clicking the **Versions** button on the **Plug-ins** tab and selecting **Copy versions from plug-in and fragment manifests**.

After your feature's version has been updated, you must update your site.xml file (if applicable) to ensure that it lists the new version of your feature.

**Note:** These steps are not required when running or debugging your application in the local test environment.

# Administering HATS rich client applications

For HATS rich client applications, runtime settings, such as log and trace settings, are specified in two files in the runtime extension plug-in (com.ibm.hats.rcp.runtime.extension):

- The runtime.properties file is used for specifying settings used by HATS in the runtime environment and in the local test environment running in Run mode.
- The runtime-debug.properties file is used for specifying runtime settings used in the local test environment running in Debug mode.

For more information about running in different test modes see Testing modes for rich client projects in *HATS Getting Started*.

While in the HATS Toolkit, you can edit these files to change settings, for example trace and trace level settings. The files are located in the root of the runtime extension plug-in project. Be aware that any changes made to the runtime.properties file in the HATS Toolkit are retained and become effective when you deploy the HATS application to a runtime environment. In the runtime environment, the runtime.properties file is located in the runtime extension plug-in directory, for example, *<eclipse_install_dir>*/eclipse/plugins/com.ibm.hats.rcp.runtime.extension_1.0.0. The runtime-debug.properties file is ignored in the runtime environment. For more information about log and trace settings in these files, see Appendix A, "Runtime properties files," on page 469.

For information about how the user can configure log and trace settings and gather problem determination data in the runtime environment, see "Preferences" on page 81.

There is just one runtime extension plug-in project in the workspace, and just one version of the plug-in in the runtime environment. Therefore, in any single HATS rich client runtime environment, log and trace settings are the same for all HATS rich client applications.

License settings are also stored in the runtime.properties file of the runtime extension plug-in (com.ibm.hats.rcp.runtime.extension). Therefore license tracking is performed at the rich client environment level and not at the level of an individual HATS rich client application. Warning messages will appear in the log if the user goes over the number of licenses purchased. For information about specifying license settings, see Enabling HATS runtime and license settings in *HATS Getting Started*

# Runtime environment

## Applications view

The Applications view displays for the user all active HATS rich client applications installed in the runtime environment. The view is sorted by application display name. By default, this view is displayed the first time the user opens the Host Access perspective. The user starts a new instance of a HATS application by double-clicking the application in the Applications view, or by right-clicking on the application name and selecting **Open**.

After starting a new instance of a HATS application, its transformation view is displayed. See "Transformation view" on page 79 for more information. By default, the name of the newly opened transformation view is the name of the application. If more than one instance of the application is started, the name of the transformation view will be a combination of the name of the application plus a number indicating the instance of the session. For example, the title of the second instance of the application named `Order Entry` will be titled `Order entry : 2`.

By default, a user can launch multiple instances of the same application. You can modify the plugin.xml file to disallow launching multiple instances. Also, you can decide not to show the Applications view and instead control what views are available to the user. This can also restrict the number of instances of an application. For more information see the *HATS Rich Client Platform Programmer's Guide*.

If the application is running in the local test environment and if you have selected to see the display terminal, a display terminal window will be displayed when the connection is made to the system. The window will close when the session has ended, or if the client is shut down.

## Pop-up menu

A pop-up menu is provided to enable the user to perform actions on one or more applications in the view. Select one or more application names and right-click to show the menu. The following menu items are available:

**Open** Selecting this menu item starts a new instance of the selected application. If multiple applications are selected, an instance of each selected application is started.

**View Print Jobs**
This menu item is only enabled when at least one print-enabled application is selected. Selecting this menu item opens either the Print Jobs (3270) or Print Jobs (5250) view, which displays a list of queued print jobs. See Chapter 15, “Enabling print support,” on page 353 for more information.

**Properties**
This menu item is enabled if only one application is selected. Selecting this menu item allows a user to see information about the application as well as configure some settings related to the application. The following pages are provided in the Properties dialog for a selected application:

**Info** The Info page displays information about the selected application, including its name, ID, version, and physical location of the plug-in. The page also shows the version of the HATS runtime plug-in being used by the application. This information is useful when a user must communicate information about the environment to a support team. The user cannot change any values on this page. It is for information only.

**Connection Parameters**
This item does not appear in the list if you have chosen to disallow the user from overriding any connection parameters, which is the default. See “Connection parameter overrides” on page 106 for more information.

If you allow overrides, this page allows the user to override connection parameters. This function is equivalent to the HATS function for Web applications that allows a user to override connection parameters using the URL when the application is first accessed. An example of using connection parameter overriding might be where the user needs to override a workstation ID or LU name.

The user sets an override by clicking the **Add** button, selecting a parameter from drop-down list for the **Name** field, and setting a value in the **Value** field. No validation within the dialog is performed to alert the user whether the supplied connection parameter name can be overridden. By default, the override setting is enabled when first added. It can be disabled by clearing the check box next to the parameter name.

When the user overrides a connection parameter, the settings are saved on the local workspace. Overridden parameter values are used when an instance of the session is started.

For information about programmatically starting a rich client application instance using connection parameter overrides, see the *HATS Rich Client Platform Programmer's Guide*.

**Variables**

This item will not appear in the list if you have chosen to disallow the user from overriding any global variables, which is the default. See "Global variable overrides" on page 115 for more information.

If you allow overrides, this page allows the user to override global variables. This function is equivalent to the HATS function for Web applications that allows a user to override global and shared global variables using the URL when the application is first accessed.

This capability allows user-specific variable data to be used in the application. For example, you can record a macro that prompts for a user name and password supplied from the values of global variables. The user can supply these values in the variables override page. These values are then used when the macro is run when the sign on screen is recognized.

The user sets an override by clicking the **Add** button, selecting a variable name from drop-down list for the **Name** field, setting a value in the **Value** field, and specifying whether the global variable is **Shared**. No validation within the dialog is performed to alert the user whether the supplied global variable name can be overridden. By default, the override setting is enabled when first added. It can be disabled by clearing the check box next to the variable name.

When the user overrides a global variable, the settings are stored in the data directory of the environment. Overridden global variable values are used when an instance of the session is started.

For information about programmatically starting a rich client application instance using global variable overrides, see the *HATS Rich Client Platform Programmer's Guide*.

### Print screen support

In the rich client environment, the user has the ability to print the current screen transformation by selecting **Print** from the **File** menu. By default, Eclipse displays the Print dialog where the user can select the printer and also configure other options. After the user clicks **OK**, the transformation area of the transformation view is sent to the selected printer.

**Note:** On Windows systems, this will print the entire transformation, including parts that are only visible by using the scroll bar. On Linux systems, this will print only the part of the transformation that is visible, and not the hidden parts that must be scrolled to see.

## Transformation view

The transformation view is the main area for transformation in the rich client environment. Each HATS rich client project has one transformation view associated with it. The view is registered in the plugin.xml file of the project when the project is created. The view is composed of four areas:

- Toolbar
- Template and transformation area
- Host keypad
- OIA status area

## Toolbar

The transformation view toolbar contains buttons for application-level actions as well as buttons for actions contributed by the Toolbar widget, during screen rendering. By default, the toolbar is displayed, and the buttons display with text-only captions. You can change these settings in the project settings. See "Toolbar RCP-only " on page 98 for more information.

The toolbar is divided into two sections. The right side of the toolbar is equivalent to the application keypad in a HATS Web application and contains application keypad buttons. See "Application keypad" on page 99 for how to configure which buttons to display. The left side displays buttons contributed by the Toolbar widget. See "Toolbar RCP-only " on page 294 for more information.

Tooltip text that provides a description of the button's function is available on all application keypad buttons. You can set tooltip text on buttons you add using the Toolbar widget.

Above the application keypad buttons on the toolbar, next to the Minimize and Maximize icons is a Menu icon. Clicking this icon displays a menu with the following menu items:

**Toggle Keyboard Support**
: This allows the user to enable and disable keyboard support.

**Connection Details**
: Displays the details of the host connection for this application.

**Properties**
: Displays the properties for this application.

## Template and transformation area

The template and transformation area is the main part of the transformation view and is where all screen transformations are displayed. It is also the area where error messages, disconnect messages, and macro handler panels, are displayed.

**Pop-up menu**
: A pop-up menu is provided in the transformation area that enables the user to view information for the current application. The following menu item is available:

  **Properties**
  : Selecting this item opens the properties dialog for the current application. See "Pop-up menu" on page 78 for more information.

**Runtime status panels**
: As with HATS Web applications, a panel is displayed to the user if the HATS rich client application stops, disconnects, has an error, or is busy.

## Host keypad

The host keypad displays the host keys you select to display in the project settings. See "Host keypad" on page 99 for how to configure your host keypad. By default, no host keypad is displayed.

## OIA status area

The OIA status area displays the information about the state of the session that you configure in the project settings. See "Operator information area" on page 99 for how to configure your OIA status area.

### Closing the view

The connection to the host system is closed when the view is closed. This occurs if the user explicitly closes the view or closes the rich client environment. When closed, the set of actions defined in the disconnect and stop events are processed.

**Note:** If the user attempts to close the view while HATS is still processing a request, for example, the user clicked a button which started playing a macro, a message is displayed indicating that the session is busy. If the user continues, the session is terminated and actions in the disconnect and stop events may not get run.

## Workstation ID prompting

When you create or edit a host connection, you can specify that the user be prompted to enter a workstation ID for connecting to a 5250 host. See “Creating a connection” on page 131 for more information. If you specify prompting for the workstation ID, the user is presented a prompt dialog when first connecting to the host. The user can then enter a workstation ID in the **Workstation ID** field.

A check box is provided that, if selected, causes the workstation ID to be saved and prevents future prompting. If the check box is selected, prompting can later be restored by disabling the workstationID connection parameter override in the application's properties. See “Connection Parameters” on page 78 for more information.

## LU name prompting

When you create or edit a host connection, you can specify that the user be prompted to enter an LU name for connecting to a 3270E host. See “Creating a connection” on page 131 for more information. If you specify prompting for the LU name, the user is presented a prompt dialog when first connecting to the host. The user must enter an LU name in the **LU Name** field.

A check box is provided that, if selected, causes the LU name to be saved and prevents future prompting. If the check box is selected, prompting can later be restored by disabling the LUName connection parameter override in the application's properties. See “Connection Parameters” on page 78 for more information.

## Preferences

By default, users can access preferences by selecting **File > Preferences > Host Access** from the runtime platform menu bar. The pages displayed are registered in the runtime extension plug-in. If you do not want to provide these functions to the user, you can remove the applicable declarations in the plugin.xml file of the runtime extension plug-in. For more information about how to do this, refer to the description of the org.eclipse.ui.preferencePages extension point ID in the HATS runtime extension plug-in section of the *HATS Rich Client Platform Programmer's Guide*. The following preference settings can be selected:

- Print
- Troubleshooting

### Print preferences

The following print preferences can be set by the user for applications that support 3270 printing:

**Default print job actions**

**On start**

The following preferences can be set when a print job starts:

- Activate print jobs view
- Display a message

  A message is displayed indicating to the user that a print job has started.
- None

**On complete**

The following preferences can be set when a print job completes:

- Activate print jobs view
- Display a message

  The message, `Print job {0} has completed.`, is displayed. Where {0} is the print job name.
- Open
- None

**On error**

The following preferences can be set when a print job fails:

- Activate print jobs view
- Display a message

  The message, `Print job {0} has encountered an error.`, is displayed. Where {0} is the print job name.
- Delete
- None

**Delete print jobs on exit**

If selected, print jobs are automatically deleted when the environment is closed. This may be useful for users who share a workstation with others.

## Troubleshooting preferences

The following troubleshooting preferences can be set by the user. For more information about log and trace settings, see "Administering HATS rich client applications" on page 76.

**Trace settings**

The following HATS runtime traces can be enabled by the user:

**Enable runtime tracing**

Enables tracing for the connections in your application.

**Enable widget tracing**

Enables tracing for widgets in your application.

**Enable action tracing**

Enables tracing for event actions in your application.

**Enable HOD tracing**

Enables Host On-Demand tracing in your application.

**Enable component tracing**

Enables tracing for components in your application.

**Enable util tracing**

Enables tracing for runtime utilities in your application.

**Enable macro tracing**
Traces the playing of macros. Because of its effect on system performance, you should use this trace only for debugging macros.

**Show display terminal**
Select this box to show a view of the host application in the display terminal while the HATS application is running.

**Enable host simulation recording**
Select this box to enable host simulation recording during runtime. Recording starts the next time the rich client application starts and ends when the host session is closed, for example when the user clicks the disconnect item on the transformation view toolbar.

The trace file is saved in the host simulations folder in the rich client environment. For example, in the Eclipse RCP environment on Windows it is saved in, <*Eclipse installation directory*>\plugins\ <*ApplicationName_version*>\profiles\hostsimulations\. In the Lotus Notes environment on Windows it is saved in, <*Lotus Notes installation directory*>\Data\workspace\applications\eclipse\plugins\ <*ApplicationName_version*>\profiles\hostsimulations\. In the Lotus Expeditor Client environment on Windows it is saved in, <*Lotus Expeditor Client installation directory*>\shared\eclipse\plugins\ <*ApplicationName_version*>\profiles\hostsimulations\. In the Lotus Expeditor Client environment on Linux it is saved in, /opt/IBM/Lotus/ Expeditor/shared/eclipse/plugins/<*ApplicationName_version*>/profiles/ hostsimulations/.

Trace files are named using the following template, `ApplicationName_ConnectionName_Date(yyyymmdd)_Time(hhmmss)_Number`, for example, MyApplication_main_20060101_134543_1.

**Trace output tab**

**File name**
The name used as a template to generate unique sets of trace files for each runtime environment. The default is **trace.txt**.

**Maximum file size**
Specifies the maximum file size in KB. The default is **10240**.

**Maximum number of files**
Specifies the maximum number of trace files. The default is **5**.

**View button**
Click this button to cause the trace files to be opened in the default system editor. All the trace files that match the specified template file name are combined into a single file and then opened. For example, if trace.txt is the template file name, and trace1.txt, trace2.txt, and trace3.txt are present, then they are all combined into a single file, in the correct order, and opened. If no default editor is configured, an error message is displayed.

**Clear button**
Click this button to be prompted whether to clear the current trace file.

**Messages output tab**

**File name**
The name used as a template to generate unique sets of log files for each runtime environment. The default is **messages.txt**.

**Maximum file size**
Specifies the maximum file size in KB. The default is **512**.

**Maximum number of files**
Specifies the maximum number of trace files. The default is **2**.

**View button**
Click this button to cause the current log file to be opened in the default system editor. All the log files that match the specified template file name are combined into a single file and then opened. For example, if messages.txt is the template file name, and messages1.txt, messages2.txt, and messages3.txt are present, then they are all combined into a single file, in the correct order, and opened. If no default editor is configured, and error message is displayed.

**Clear button**
Click this button to be prompted whether to clear the current log file.

**Simulation output tab**

**Maximum number of files**
When the **Enable host simulation recording** check box is selected, this field specifies the maximum number of simulation output files. The default is **10**. If a new simulation file exceeds the maximum setting, the oldest simulation file is deleted.

**Gather button**
Click this button to gather all the artifacts required by IBM Support to diagnose a problem. In the Gather Problem Determination Information wizard, select the check box for the application for which you want information gathered and specify the destination .zip file. The following information is collected and gathered into the destination .zip file:

- contents of every selected application
- contents of the runtime extension plug-in which includes the trace, log, and runtime.properties files
- the .log file from the current data directory

**Restore Defaults button**
Click this button to cause all settings in the panel to revert to default values.

**Apply button**
Click this button to save the settings in the runtime.properties file in the runtime extension plug-in.

## HATS rich client considerations and limitations

Following is a list of considerations and limitations when using HATS rich client applications:

- The jclDesktop JRE is installed by the Lotus Expeditor Toolkit version 6.2.0 and earlier. This JRE does not contain Swing and AWT libraries on which HATS rich client support depends. When HATS detects this JRE for the workspace, it displays a warning dialog that the current JRE is incompatible with HATS and provides an option to change the JRE to the default Eclipse JRE. If you choose not to change the JRE, then you must change it manually. Use the **Show incompatible JRE warning** preference setting to display or suppress the warning dialog. See "Using HATS preferences" on page 126 for more information.

- Limitations exist relating to the use of the text replacement function to replace text with images in combination with the use of certain SWT widgets in rich client applications. For more information see “Text Replacement” on page 166. The widgets and limitations are listed below:
  - Button - if any part of the caption text is replaced with an image, all other caption text is discarded.
  - Button table - same as button.
  - Check box - same as button.
  - Combo - no image replacement occurs within the drop-down itself. Image replacement does occur for the caption text. Other text, not replaced with an image, is preserved and displayed in the caption.
  - Drop-down (data entry) and Drop-down (selection) - same as combo.
  - Graph - no image replacement occurs.
  - Link - no image replacement occurs.
  - List - same as combo.
  - Popup - same as combo
  - Radio button (data entry) and Radio button (selection) - same as button.
  - Subfile (check box) - image replacement occurs within the table. If any part of the caption text on a button above or below the table is replaced with an image, all other caption text is discarded.
  - Subfile (drop-down) - image replacement occurs within the table only. No image replacement occurs within the drop-down.
  - Text input - same as combo.
  - Toolbar - since images are determined by the defined image mappings, text replacement with images does not occur when the toolbar widget is used.
- Java Visual Editor support for rich client applications is removed from HATS V9.7. You cannot use Java Visual Editor to edit HATS rich client transformations and templates.

# Chapter 5. Modifying a HATS project

When you create a project using the HATS Create a Project wizard, the settings you choose in the wizard are saved in a project application (.hap) file. You can invoke the project editor by double-clicking on **Project Settings** under the name of the project you want to modify in the **HATS Projects** view.

The settings displayed in the project editor are the settings that are used for the entire project. If you want to modify any of the settings, you can use the tabs of the project editor. Saved changes made in the project editor are automatically recognized when running your project in a test environment by clicking **Refresh** on the application keypad or by displaying a new host screen in the GUI. The **Refresh** will not pick up changes made to the connection (.hco) files. Connection setting changes are applied when the application server (or application) is restarted.

**Note:** Settings used for the entire HATS project should not be confused with settings applied throughout HATS. HATS-wide settings can be applied from **Window > Preferences > HATS**.

The following sections describe each tab of the project editor, and explain how they can be used to modify the project settings.

## Overview

The **Overview** tab of the project editor summarizes all of the settings you specified when you created your project.

The **General Information** section contains the project name and description, the chosen template and theme, and the date last modified.

**Note:** A theme is not displayed for projects optimized for mobile devices. For more information, see "Developing HATS applications for mobile devices" on page 44.

Click the link for the current theme to change the project theme. Four options are provided, **Standard**, **Modern Web application** `Web-only` or **Modern UI application** `RCP-only`, **Classic terminal emulator**, and **Custom**. Each function in the list of settings can be enabled elsewhere in the project settings. This list provides a convenient method of setting the functions in one place that together can be treated as a theme for the appearance and behavior of your application.

The following table shows how each theme setting maps to an individual project setting.

*Table 1. Theme setting to project setting map*

| Theme Setting | Project Editor Tab | Project Setting |
|---|---|---|
| Arrow key navigation `RCP-only` | Other | Client settings > Enable arrow key navigation |
| Automatic field advance | Other | Client settings > Enable automatic field advance |

*Table 1. Theme setting to project setting map (continued)*

| Theme Setting | Project Editor Tab | Project Setting |
|---|---|---|
| Cursor positioning on protected fields | Rendering | Widgets > Field > Allow cursor positioning on protected fields |
| Field extended attributes | Rendering | Widgets > Field > Enable extended attributes |
| Host keypad | Rendering | Host keypad > Show default host keypad |
| Keyboard support | Other | Keyboard support > Enable keyboard support |
| Operator information area | Rendering | Operator information area > Show OIA |
| Overwrite mode (initial) | Other | Client settings > Overwrite mode (initial) |
| Rendering using monospaced font | Rendering | Widgets > Field > Render using monospace font |

The Standard theme enables the functions that are enabled by default when a HATS project is first created. Selecting the Modern theme deselects all of the individual functions. Selecting the Classic theme selects all of the functions. Selecting the Custom theme allows you to select any other combination of functions.

**Note:** For more theme settings when using DBCS support see "Project theme settings" on page 459.

The **Testing** section contains links that can be used to test your project. For HATS Web projects the links are:

- Launch on WebSphere Application Server
- Launch on WebSphere Application Server in Debug mode

For HATS rich client projects the links are:

- Launch on target platform
- Launch on target platform in Debug mode

**Notes:**

1. Clicking either of these links will perform the same operation as if you right-click on your project in the HATS Projects view and select either **Run on Server** (**Run** if rich client) or **Debug on Server** (**Debug** if rich client).
2. For more information about testing modes see Testing your project in *HATS Getting Started*.
3. For more information about testing HATS rich client applications see "Testing HATS rich client applications" on page 61.

The **Deployment** `RCP-only` section includes a **Target platform** drop-down. Use this drop-down to change the target deployment for your rich client project. This allows you to start building an application targeted for one platform and switch to a different platform at a later time. Supported platforms are **Eclipse Rich Client Platform**, **Lotus Expeditor Client for Desktop**, and **Lotus Notes**.

If you change the target platform here, and it does not match the target platform of the workspace, a message is displayed informing you of this. The action to take in this case depends on which platform you have changed to for this project.

If you changed to a Lotus Notes or a Lotus Expeditor Client platform, see "Configuring the target platform" on page 61 for information about setting the target platform.

If you changed to the Eclipse Rich Client Platform, select **Window > Preferences > Plugin-in Development > Target Platform** and select your Rational SDP installation directory (the default is C:\Program Files\IBM\SDP) from the **Location** drop-down list.

When prompted to reload the target platform, select **Yes**.

Also in the **Deployment** section is a link, **open the plug-in manifest**, that launches the editor for the plug-in descriptor (plugin.xml) of the project. Advanced rich client development will require modifications to this file.

The **Connections**, **Rendering Settings**, and **Other Settings** sections summarize the settings specified on each of their corresponding tabs. The headings themselves are links to their corresponding tabs which are described in sections that follow.

## Connections

The **Connections** tab displays the information about your project's connections. There are two types of connections in HATS, default (also referred to as transformation) and background. Each HATS application has one default connection for the host application whose screens HATS will transform. Background connections are any connections in a project other than the default connection. HATS does not transform screens from background connections. For more information, see Chapter 6, "Managing connections," on page 131.

You can select your default host connection from the **Default** drop-down list.

There is also a list table displaying all of your project connections. The list displays the connection name, host, host type, port number and code page. You can add additional project connections by clicking the **Add** button which launches the **Create a Connection** wizard. HATS allows you to specify more than one connection which can be used for your project. For information about using the wizard see "Creating a connection" on page 131.

**Note:** If you select a bidirectional (bidi) code page, refer to "Additions to HATS files" on page 455.

To edit an existing connection, select the connection and click **Edit** to launch the connection editor. For more information see "Connection editor" on page 132.

The **Remove** button will delete the highlighted connection. If you click **Refresh**, the list will be updated.

## Template

The **Template** tab displays the template used to surround a transformation.

On this tab, you can change which template to use as the default template. It will be used for all screens in your application unless you specify a different template for a particular screen using an action in a screen customization.

**Note:** Newly supplied templates are marked with an asterisk.

The default template is also the template applied to all transformations in the project, and the template applied with the default transformation as the default action of an unmatched screen event. For information about how to modify the action of the unmatched screen event, see "Application events" on page 102.

When creating or modifying the actions of a screen customization, you can override the default template chosen by selecting a different template. You can also edit the template while you are on the **Template** tab.

You can create your own templates to use for your HATS projects. For more information, see "Create a Template wizard" on page 314.

# Rendering

The **Rendering** tab displays the settings for default rendering, global rules, text replacement, components, widgets, toolbar `RCP-only`, application keypad, host keypad and operator information area. This is where you can configure project-level default values for your HATS project.

## Default rendering

Default rendering is responsible for transforming host screens that have not been individually transformed. The selected default rendering set attempts to preserve the original host screen structure while extending the functionality of the application by applying GUI design principles. That is, it does more than render host screen non-protected fields as GUI input fields; it can transform function keys into buttons or links, selection lists into drop-downs, and tabular regions into tables.

**Notes:**

1. When you create an individual transformation for a host screen, you can prepopulate it with default rendering. For more information, see "Create a Transformation wizard" on page 173.
2. You can also later insert default rendering into a transformation. For more information, see "Insert Default Rendering" on page 180.

A rendering set is configured by creating a prioritized list of rendering items. Each rendering item defines a specific region in which a specified host component is recognized and then rendered using a specified widget. For example, you can look for function keys in the bottom region of the host screen and then render them as links.

You can create more than one rendering set for your default transformation. If you create additional rendering sets, only one will be used in the default transformation. To create a new set, click the **Add** button to open the **Add Rendering Sets** window. Specify a unique **Name** and **Description (optional)** for your new rendering set.

If you want the new rendering set as the default, select the **Use this rendering set for default rendering** check box.

A rendering set can either be blank (no rendering items defined) by selecting the **Create empty rendering set** button or you can select a set from the **Copy new rendering set from existing set** pull-down menu. Rendering sets can also be modified or deleted by selecting the **Edit** or **Remove** buttons beside your list.

The following list of rendering items are defined as part of the default rendering set:

- Dialogs
- ENPTUI windows
- Subfiles
- ENPTUI menu bars
- ENPTUI actions
- ENPTUI single selections
- ENPTUI multiple selections
- ENPTUI scrollbars
- Light pen - attention
- Light pen - selection
- Selection lists
- Function keys
- Field tables
- Visual tables
- URLs
- Remaining text and input fields

**Notes:**

1. ENPTUI rendering items appear only if you are using a 5250 or 5250W host and only if ENPTUI support was enabled by selecting the **Add graphical interface DDS keywords (ENPTUI) rendering support** check box during the initial creation of the HATS project.
2. Light pen rendering items appear only if you are using a 3270 or 3270E host and only if light pen support was enabled by selecting the **Add light pen rendering support** check box during the initial creation of the HATS project.

You can add a new rendering item to the selected rendering set for all screens in this project as well as edit or remove existing rendering items by selecting the item to be edited or removed and select the appropriate button to the right of the table.

If you select the **Add** or **Edit** buttons, enter a name and description for your rendering item. Next, define the screen region in which your rendering item is applied and then designate the host component to use for recognition and the widget to use for rendering. You can also change the order in which the rendering items are applied, select an item in the list and click the **Up** or **Down** buttons to move it towards the top or bottom of the list. Refer to the "Insert Host Component" on page 178 or "Edit Host Component `Web-only` " on page 179 for more details on the last two panels of the wizard.

**Notes:**

1. Dojo widgets are not supported in default rendering items.
2. The dialog component is a special component used in rendering to recognize modal frames (pop-up frames) on the host screen and render them. However, you can not use the dialog component to insert modal frame host components into individual transformations.

The list of rendering items that make up the rendering set is an ordered list. The ordering of the list matters because each rendering item may consume a part of the host screen that another item further down the list may have recognized as well.

The higher the rendering item on the list, the higher the priority. The check box next to each rendering item is used to enable or disable the selection.

Components do not always consume the entire region you specify. Suppose you have configured the selection list component to look for selection lists on the entire screen, but only one list was found in the middle of the screen. Only the region in the middle of the host screen will be marked as "consumed". Any remaining "unconsumed" regions of the host screen can still be transformed by lower priority rendering items.

Care should be taken when using the Table component in a default rendering set because the Table component will recognize almost every screen in the selected region, and no rendering items lower in the list will be recognized.

The final rendering item in the default list, **Remaining text and input fields**, will transform all remaining "unconsumed" screen regions using the Field component and Field widget. This item should be the last item in a rendering set.

In general, the default rendering set attempts to preserve the original host screen structure while extending the functionality of the application by applying GUI design principles. However, to allow default rendering of host screens to be displayed on mobile devices, a certain amount of compacting may be necessary. With compacting, the amount of HTML and blank space is reduced which may possibly display a different structure of the original host screen. To specify that the currently selected rendering set use compacting, select **Use compact rendering** `Web-only` .

**Note:** For Web projects optimized for mobile devices, a rendering set, named **compact**, is created as the default with this setting selected.

## Multiple rendering set example

Suppose you want to create one HATS application, and from that application, allow users to access two host applications that look very different from one another. You have decided you want different rendering sets for each, but want to use the power of HATS default rendering and have as few specific screen customizations as possible.

Call one application APPA and the other application APPB:

1. Create a rendering set specifically tailored to APPA
2. Create a rendering set specifically tailored to APPB
3. Create a transformation "APPA" that has a default rendering tag with a parameter for the APPA rendering set
4. Create a transformation "APPB" that has a default rendering tag with a parameter for the APPB rendering set
5. Create a screen customization that recognizes only the first screen of APPA and call it APPAfirst, with actions:
   a. set global variable "WhichApp" to "APPA"
   b. apply transformation APPA
6. Create a screen customization that recognizes only the first screen of APPB, call it APPBfirst, with actions:
   a. set global variable "WhichApp" to "APPB"
   b. apply transformation APPB

7. Create a screen customization called APPArest, that recognizes true if global variable "WhichApp" equals "APPA", with action: apply transformation APPA
8. Create a screen customization called APPBrest, that recognizes true if global variable "WhichApp" equals "APPB", with action: apply transformation APPB
9. Order the screen customizations in your project settings **Events** tab like so: APPAfirst, APPBfirst, APPArest, APPBrest

## Advanced rendering

To change advanced rendering settings, expand the default rendering tree and click **Advanced**.

**Alternate rendering:** In this section you can specify that default rendering should be used if nothing is recognized to render during transformation of a HATS component. This function is particularly useful when an application contains screens that have fields which may switch between unprotected and protected states. For example, if this function is not used and an input field component is being used to recognize the region, nothing will be recognized and rendered if the field switches to a protected attribute state.

**Use default rendering on component rendering failure**

Use this option to specify at the project level the action to be taken if a host component does not recognize a region in a screen event. This option is not selected by default. This causes nothing to be rendered if a region is not recognized by the specified component. If the option is selected, the region is transformed using default rendering.

**Rendering set**

From this drop-down, select the rendering set to use for the transformation from the list of currently defined rendering sets.

**Notes:**

1. If the selected rendering set is later deleted from the project and therefore does not exist at runtime, the default rendering set is used.
2. You can also specify this function at the individual component level in a screen transformation. For more information see “Insert Host Component” on page 178.
3. Dojo widgets are not supported in alternate rendering sets.

**HTML tables:** In this section you can select an option to create better-formed HTML in default rendering.

**Close default rendering table data and row tags**

Select this option to specify at the project level that all table data and row tags in default rendering be closed, that is, </td> and </tr>.

This option is not selected by default in order to reduce the amount of HTML included in default rendering. Reducing the amount of HTML is desirable for performance sensitive applications. However, selecting this option may be required for other applications, for example, screen reader applications used for accessibility.

## How to make default rendering customizations available in new HATS projects

If you want your default rendering customizations to appear the next time you create a new HATS project, follow these steps:

1. Create a project and customize your default rendering options.

2. Open the application.hap file for this project. For Web projects, it is located in *workspace_directory*\project name\Web Content\WEB-INF\profiles\. For rich client projects, it is located in *workspace_directory*\project name\profiles\.
3. Copy the source between the <defaultRendering> and </defaultRendering> tags in the application.hap file and replace the source between the <defaultRendering> and </defaultRendering> tags in the application.hap file located under the installed HATS offering.

   For Web projects, the complete file path is:

   <*shared_install_directory*>\plugins\com.ibm.hats_*nnn*\predefined\projects\new\Web Content\WEB-INF\profiles\application.hap

   For rich client projects, the complete file path is:

   <*shared_install_directory*>\plugins\com.ibm.hats_*nnn*\predefined\projects\rcp\new\profiles\application.hap

   In the file paths above, *shared_install_directory* is the shared resources directory where you installed the HATS offering using IBM Installation Manager, and *nnn* is the version and build level of HATS.

   **Note:** You must repeat this procedure after installing HATS maintenance. A new application.hap file will be located under a new `com.ibm.hats_nnn` directory structure.

## Global rules

Global rules enable pattern recognition and transformation of host input fields and work with customized and non-customized (default rendered) screens. They can be defined at both the project level and at the screen level. Use this section to specify project-level global rules. See “Global Rules” on page 165 for where to define screen-level global rules.

If a project-level and a screen-level global rule are both defined for the same input field, the screen-level rule has priority.

As an example, you can create a global rule that renders any input field with the word `Country` before it as a drop-down containing codes for the various worldwide countries. Project-level global rules are truly global to your HATS application; they work in screens both transformed by the rendering set, and also in custom transformations. This allows you to enable your entire application to recognize a particular host screen pattern without modifying any of your transformations.

A global rule consists of a configured pattern type and a transformation fragment. The pattern type configuration specifies what sort of content to look for on the host screen. It lets you transform specific fields or all fields of a certain size, or those nearest to a string you specify. For example, the pattern to recognize might be an input field preceded by the word `Date`. The transformation fragment contains the content to use to replace all occurrences of the pattern in all transformations. The transformation fragment might contain a HATS input field component and a calendar widget. This would result in all input fields preceded by the word `Date`, across the entire application, displaying as a calendar control.

All configured global rules are displayed in the global rules table. Global rules are processed in the order they appear in this list at the beginning of every **Apply transformation** action. Once an input field is recognized by one global rule in the list, it will not be recognized by a subsequent global rule. The check box next to each item indicates whether the item is enabled or disabled.

1. Click the **Add** button to add a global rule definition to the list of global rules for this project. If you do not have a screen capture, you will be prompted to open the host terminal to capture a screen. Creating a new global rule involves entering a **Name**, **Package** (for rich client projects), and **Description**, and associating it with a **Create a new transformation fragment**. If you are editing a global rule, you can **Use an existing transformation fragment**.
2. Next, select a pattern type to use. The pattern types to select from are as follows:
   - The **Find input fields by surrounding text** pattern type recognizes any input field which has a designated string bordering it on the left, right, top, or bottom. This pattern type requires you to select specific **Pattern Settings**. The designated string can contain one or more asterisks (*) used as wildcards. Spaces are significant. You can either **Transform** the nearest input field only or all input fields. Specify in the **Located** drop-down list where the text pattern must be in relation to the input field for the global rule to be applied. Insert the text for which you want this global rule applied in the **A protected field containing** field. If the **Case sensitive** box is used, it will search for the exact text specified. This pattern type is used by the `Country` and `Date` examples above.

     When you select **All input fields** for the **Transform** option, the first input field that matches the criteria determines the scope of the search region for subsequent fields. The example below shows the fields that are considered matches for all input fields below a protected field containing the string `Enter.`. The Global Rule Settings wizard highlights all of the matching fields. The first match is the field with the caption `Alpha field (blue)`. The start and end columns of this field determine which fields below it are also considered matches. Notice the fields with captions `Numeric field (green)` and `Column separators` are not considered matches because they are either longer or shorter than the first matching field. The field with the caption `Date field (pink)`, and the three fields immediately below it, are considered matches because they have the same start and end columns as the first matching field.

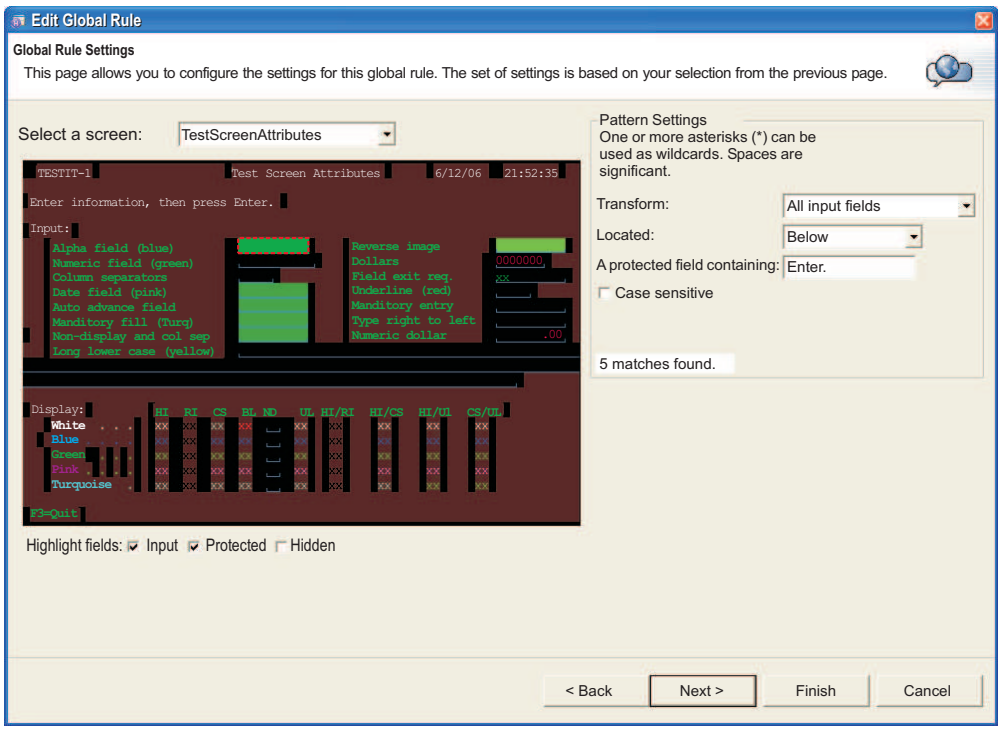



*Figure 11. HATS global rule example*

- The **Find all input fields** pattern type successfully recognizes all host screen input fields. This pattern type has no pattern settings. A global rule using this pattern type should be placed at the bottom of the global rule list to give it the lowest priority.
- The **Find input fields by field size** pattern type recognizes any input field which has a designated length. For example, this can be useful for transforming input fields of length one into check box controls.
- The **Find input fields within a specified region** pattern type recognizes any input field located within a host screen region as specified by start row, start column, end row, and end column. Optionally, a field length can be specified to further define the input fields to locate within the region. A host field must be entirely contained within the specified region to be recognized. You can use this pattern type in conjunction with screen-level global rules to pinpoint an input field to customize on a screen, for example data within a table, while still using default rendering for the screen.

Each pattern type component can contain a set of customizable settings as described above. As you change settings, the screen on the left is updated to reflect which input fields would match (note that multiple fields will be highlighted if multiple fields match). You can also highlight certain fields on your host screen by selecting the different options beside **Highlight fields**. If you want to see where the input fields are defined on the screen, select the **Input** check box. If you want to see what fields are protected, select the **Protected** check box. If you want to highlight any hidden fields, select the **Hidden** check box. To modify the colors of the input, protected or hidden fields highlighting, see “Using HATS preferences” on page 126. If you are using an existing fragment, click **Finish** to update the global rule set and skip the section below.

3. If you are creating a new fragment, you must now configure the HATS component to be inserted into the transformation fragment. The purpose of

global rules is to find input fields; therefore, only input components are displayed. The host screen region that is being used for configuration of this page is denoted by the selection box surrounding the first matched input field from the previous page, but remember that a project-level global rule will act on all matched input fields, on this host screen and all others. Select a component and widget, and configure their settings. Refer to the "Insert Host Component" on page 178 or "Edit Host Component Web-only " on page 179 for more details. Clicking **Finish** will update the global rules set and create a new transformation fragment in the **Transformation Fragments** folder. Make certain to save your project settings after finishing the wizard.

**Notes:**

1. Dojo widgets are not supported in global rule transformation fragments, and global rules are not used within Dojo widgets.
2. For rich client projects, input field components with SWT table widgets cannot be rendered using global rules.

If you need to modify the component and widget settings of an existing global rule, you can go to the **Rendering** tab, select **Global Rules** and select the transformation fragment from the list then click **Edit**.

The transformation fragment is initially created with a special HATS component which corresponds to the settings chosen on the third page of the global rule wizard. This component is special because it does not have region attributes. Instead, it dynamically obtains the region to render at runtime. You can edit this component as you would any other HATS component, but on the first page of the wizard, you must select a global rule instead of a screen region. You can insert other valid HATS transformation content into this transformation fragment as well.

**Note:** When you create a global rule, and then edit the transformation fragment, you are not able to see the output on either the design or preview tabs. To view the output, you must use the full page preview from the Edit Host Component function while editing either a transformation fragment or a full transformation that makes use of the global rule.

If you do elect to manually modify the transformation fragment associated with a global rule, it should not be done after using the editor from the **Rendering** tab. Doing this results in the wrong tag being read and the removal of any modifications made.

Project-level and applicable screen-level global rules are both applied in a screen event transformation. You can disable all global rule processing at the screen event level. To disable:

1. In the **HATS Projects** view, open the **Screen Customizations**, or the **Screen Combinations**, folder that contains the screen event.
2. Double click the screen event.
3. Click the **Actions** tab.
4. Select the **Apply a transformation** action.
5. Click **Edit**.
6. Clear the **Apply project-level and screen-level global rules** check box.

## Text replacement

HATS applications can convert text strings on host screens into different text strings, HTML content Web-only , or images.

**Text Replacement** displays a table that shows the original text you want to replace, together with the text, HTML content Web-only , or image to use as the replacement. Also shown is whether the text search is case-sensitive and whether regular expression support is being used. You can add, modify, or remove any text replacement specifications by using the buttons to the right of the table of values. For information about how to specify these settings, see "Text Replacement" on page 166.

## Components and widgets

To change settings for individual components and widgets, you must expand them in the tree to see the individual components and widgets. Some of the components and widgets do not have any customizable settings. For information about the settings that can be customized with the Insert Host Component wizard, see "Component and widget settings" on page 187.

You can override the default project settings for components and widgets when you insert them into transformations. Those modified settings only apply to the individual instances of those components or widgets in the transformation. All the other instances of the component or widget for any transformation in the project still use the default project settings, unless you modify them. For example, you have default settings for the VisualTable component. In a single transformation, you can have two VisualTable components; one that uses the default settings from the project settings, and another that uses modified settings.

### Components

**Components** displays a list of host components which can be used to convert elements of a host screen to objects that can be displayed in a GUI. The settings for a host component specify how that component is to be recognized on the host screen. Some of these components can be modified. For information about host components, see "Host component settings" on page 187.

### Widgets

If you select **Widgets**, you will get a list of all HATS widgets which your project can use to display host components in your GUI. Some of these widgets can be modified. When you customize a widget, you specify how the widget will appear on the Web page. For information about widgets and the settings you can modify, see "Widget settings" on page 218.

## Toolbar RCP-only

The Toolbar settings panel enables you to change settings related to the toolbar in the Transformation view of your rich client application. The toolbar contains application keypad actions, such as Refresh and Default, as well as actions contributed by the Toolbar widget. For more information about the Toolbar widget see "Toolbar RCP-only " on page 294.

In HATS Web applications, the application keypad is normally placed at the bottom of or in the side bar of the template. For HATS rich client applications, the functions provided by the application keypad exist as buttons on the Transformation view toolbar. This helps the HATS rich client application fit in with other applications in the rich client environment. See "Toolbar" on page 80 for more information about the Transformation view toolbar.

On this panel you can select or clear the **Show toolbar** check box. If you select it, you can set how the toolbar items are displayed, as **Text**, **Image**, or **Both text and image**. This setting applies to all items on the toolbar (including those contributed by the Toolbar widget).

## Application keypad

The default application keypad is displayed in the default template. You can customize the following settings for the application keypad:

**Show default application keypad**
Select the check box if you want a default application keypad defined in templates or transformations to appear in the GUI when users interact with your application.

**Select keys to display**
If the **Show default application keypad** check box is selected, you can select the check boxes next to each of the keys that you want to include on the default application keypad in the GUI.

**Display as** `Web-only`
Select the value to determine whether the selected keys appear as buttons, links, icons, or in a drop-down list.

## Host keypad

You can customize the following settings for the host keypad:

**Show default host keypad**
Select the check box if you want a default host keypad defined in templates or transformations to appear in the GUI when users interact with your application.

**Select keys to display**
If the **Show default host keypad** check box is selected, you can select the check boxes next to each of the keys that you want to include on the default host keypad in the GUI. If you want to include all the available keys, you can select **Select All**. You can select **Deselect All** to clear all of the boxes that are selected.

**Display as**
Select the value from the drop-down list to determine whether the selected keys appear as buttons, links, or in a drop-down list.

You can also add a custom host key by clicking the **Add** button and then entering the **Caption** and **Mnemonic**. Click **OK** when finished.

## Operator information area

Operator information area (OIA) allows you to easily determine the state of the host application. You can customize the following settings for the OIA:

**Show OIA**
Selecting this box will make the OIA visible. Selecting any of the following check boxes allows you to specify what to display:

**Secure host connection**
Select this box if you want to display whether the HATS connection is SSL secured

**Input inhibited indicator**
Select this box if you want to display whether the keyboard is locked, preventing input from the keyboard.

**System locked indicator**
Select this box if you want to display whether the system is locked while waiting for data to be returned. Refreshing the screen can clear the lock.

**Message waiting indicator**
Select this box if you want to display the message waiting indicator. This is a 5250-only feature where the OIA displays an indicator when a new message is waiting for the user. Once the user visits the Display Messages panel on the 5250 host, the indicator is cleared (until another new message is received).

**Notes:**

1. The user must have their message queue delivery type set to *NOTIFY to receive a message waiting indicator.
2. Selecting this box has no effect for 3270 or 3270E connections.

**Asynchronous update**
Select this box if you want to display whether asynchronous update is enabled.

**Input mode**
Select this box if you want to display whether overwrite mode is enabled (if supported by the browser).

**Absolute cursor position**
Select this box to display the host cursor position.

**Cursor row and column**
Select this box if you want to display the host cursor row and column position.

**Automatic field advance**
Select this box if you want to display whether automatic field advance is enabled (if supported by the browser).

**Field data**
Select this box if you want to display external field data, such as numeric only or field exit required.

**Type-ahead field** `RCP-only`
Select this box to display the type-ahead field. This field displays the type-ahead data as the user enters it, but the field cannot be directly edited. This setting is configurable only when the **Enable type-ahead support** setting is selected. See "Enable type-ahead support" on page 123.

**Display area layout** `Web-only`

**Horizontal**
Select this button if you want to display the OIA horizontally (across the bottom of the page).

**Vertical**
Select this button of you want to display the OIA vertically (as a side frame on the page).

**Style class** `Web-only`

Enter a cascading stylesheet (CSS) class name that controls the appearance of the OIA.

## DBCS

You can configure DBCS options for your project if your default connection specifies a DBCS code page. For more information about DBCS settings on the Rendering tab see “Rendering tab” on page 460.

# Events

The **Events** tab displays a list of prioritized screen events (screen customizations and screen combinations) contained in your project as well as application events.

If the check box next to the name of a screen event is selected, that event is enabled for the project. When a screen event is enabled, and the screen recognition criteria match the host screen, HATS performs the actions specified for that event, and no more screen events are checked for matches. When a screen event is disabled, HATS ignores the event. If you want to test certain screen events, you can disable other screen events. To disable a screen event, clear the check box while you are testing.

HATS applications check each incoming host screen against the list of enabled screen events. If there are multiple screen events that match a given screen, the first screen event that matches the screen is applied. The higher priority screen events should be near the top of the list. For example, you might have one screen event that recognizes a few specific screens, and a second one that recognizes a more general set of screens. If the second screen event is higher in the list than the first, a screen might be recognized by the more general screen recognition criteria and perform the associated actions, rather than recognizing the screen by the more specific criteria and performing the associated actions of the first screen event.

If you want to change the priority of any of the screen events, highlight the event by clicking it. Then click either **Up** or **Down** to move the event higher or lower in the list. For more information, see “Screen event priority.”

**Note:** If you have a screen event that specifies an identified next screen, the identified screens will take priority. For more information, see “Next Screen” on page 167.

If you want to edit an application event, click the link of the event below **Application Events** that you want to edit. For more information see “Application events” on page 102.

## Screen event priority

You can modify the order of the screen events using the **Events** tab of the project editor. See “Events” for more information.

When a HATS application is running and a new host screen is encountered, HATS checks the first enabled screen event in the event priority list to determine if the screen recognition criteria match the host screen. If so, no more screen events are checked for matches, and the actions for the first screen event are performed. If not, the next screen event in the list is checked to determine whether the screen recognition criteria match the host screen. This continues until the last screen event in the list is checked.

If there are no screen recognition criteria in the screen events that match the current host screen, HATS processes the Blank screen event if it has actions and the screen is blank. Otherwise, HATS processes the unmatched screen event. The HATS unmatched screen event is a special screen event that occurs only when no defined screen events match the host screen. The default action of this event is to display the host screen (default transformation) that applies the default template.

You can modify the actions to be taken if a host screen does not match any of your screen events. For example, you can create a Web page that tells the user that the page was not found and gathers information about how the user reached that screen. You can use the **Show URL** action to present the Web page.

If you want to modify the action of the unmatched screen event, go to the **Events** tab and click **Unmatched screen** under the **Application Events** section to open its editor. Click the **Actions** tab at the bottom of the editor to display the actions. Click **Add**, **Edit**, or **Remove** to customize the unmatchedScreen.evnt file.

## Application events

The rules-based approach used by HATS enables you to customize screens by applying certain actions upon a screen recognition event. You can also associate actions with other HATS events, known as application events. Application events include application occurrences such as connecting and disconnecting from the host server, starting or stopping your HATS application, or when your application encounters an error or an unrecognized host screen. You can configure these application events to perform certain actions when they occur.

To access the application events, go to the **HATS Projects** view and double click the **Project Settings** of your HATS project and select the **Events** tab. On the **Events** tab in the Application Events section, click the link for an event to open its editor.

Each application event has various actions associated with it.

The following list provides descriptions of the HATS application events and when they might occur along with their associated actions

**Start** A start event occurs when starting your HATS application. There is no default action for the start event.

You can specify the following actions for the start event:

- Execute business logic
- Set global variable
- Remove global variable
- Show URL `Web-only`
- Show URL or SWT composite `RCP-only`
- Forward to URL `Web-only`
- Perform macro transaction
- Pause

**Connect**

A connect event occurs when your HATS application connects to the host server. For example, upon connecting you might want to configure your project to run a macro to bypass certain screens to display a logon screen. The default action for the connect event is to obtain a connection. The actions of the connect event always run after the start event.

You can specify the following actions for the connect event:

- Obtain default connection
- Execute business logic
- Set global variable
- Remove global variable
- Show URL `Web-only`
- Show URL or SWT composite `RCP-only`
- Forward to URL `Web-only`
- Play macro
- Perform macro transaction
- Disconnect
- Pause

**Blank screen**

A blank screen event allows you to specify what actions you would like to perform on the default connection whenever the host screen is blank. This optional event contains no actions by default, and is ignored. If your application often displays a blank screen and requires the user to press the Refresh button, you might add a Pause action to this event to allow the host more time to complete drawing its host screens. If you do specify actions for this event, it is checked after exhausting the list of screen events you've provided in the screen event priority list, and just before running the unmatched screen event.

You can specify the following actions for the blank screen event:

- Execute business logic
- Set global variable
- Remove global variable
- Show URL `Web-only`
- Show URL or SWT composite `RCP-only`
- Forward to URL `Web-only`
- Play macro
- Perform macro transaction
- Disconnect
- Pause

There are also special settings you can use to specify how to handle a blank screen received when initially connecting to the host. For more information, see “Screen Handling” on page 137.

**Unmatched screen**

An unmatched screen event occurs when HATS receives a screen from the host application that does not match any screen events in the HATS application. If you have configured your HATS application to use the default rendering for some screens, then this is a normal occurrence. If you have tried to recognize all possible host screens in screen events, you can configure the unmatched screen event to show a URL that tells the user he has reached an unexpected screen, asks how he got to it, and offers paths back into the proper screens. The default action for the unmatched screen event is to show the default transformation.

You can specify the following actions for the unmatched screen event:

- Apply transformation

- Execute business logic
- Extract global variable
- Insert data
- Set global variable
- Remove global variable
- Show URL **Web-only**
- Show URL or SWT composite **RCP-only**
- Forward to URL **Web-only**
- Play macro
- Perform macro transaction
- Send key
- Disconnect
- Pause

**Error** An error event occurs when HATS encounters an application or host error. You might want to add an action that displays an error in the GUI when an error event occurs. The default action for the error event is to show a URL (error.jsp for Web applications) or to show a composite (CustomErrorComposite for rich client applications).

You can specify the following actions for the error event:

- Execute business logic
- Set global variable
- Remove global variable
- Show URL **Web-only**
- Show URL or SWT composite **RCP-only**
- Forward to URL **Web-only**
- Play macro
- Perform macro transaction
- Pause

**Disconnect**

A disconnect event occurs when your HATS application disconnects from the host server. The default action for the disconnect event is to release the default connection.

You can specify the following actions for the disconnect event:

- Release the default connection
- Execute business logic
- Set global variable
- Remove global variable
- Show URL **Web-only**
- Show URL or SWT composite **RCP-only**
- Forward to URL **Web-only**
- Play macro

  **Note:** For steps to take if a macro times out while playing as part of the disconnect event, see the description of the Stop event below.
- Perform macro transaction
- Pause

**Stop** A stop event occurs after stopping your HATS application. The default action for the stop event is to show a URL (stop.jsp for Web applications) or to show a composite (CustomDisconnectComposite for rich client applications).

You can specify the following actions for the stop event:

- Execute business logic
- Set global variable
- Remove global variable
- Show URL **Web-only**
- Show URL or SWT composite **RCP-only**
- Forward to URL **Web-only**
- Play macro
- Perform macro transaction
- Pause

**Note:** If a macro times out and fails to complete while playing as part of the disconnect event, the disconnect event fails to complete, and the HATS connection is not disconnected. Macro time outs are typically caused when the macro encounters an unknown host screen. To configure the Stop event to successfully terminate the host connection when this occurs, perform the following steps:

1. Click the link for the **Stop** event to open its editor.
2. In the editor click the **Source** tab.
3. Add the `<release enabled="true"/>` action prior to the `<show/>` action.
4. The Stop event source should look like the following example:

```
<?xml version="1.0" encoding="UTF-8"?>
<event description="" type="stop">
    <actions>
        <release enabled="true"/>
        <show enabled="true" page="stop.jsp" url="/stop.jsp"/>
    </actions>
</event>
```

5. Save the project settings file and restart the application.

# Other

The **Other** tab displays the settings for keyboard support, client locale, connection parameter overrides, asynchronous update, global variable overrides, and client settings.

You can customize the default project settings for each of these by clicking on the node in the customization settings tree.

## Keyboard support

You can customize the following settings for keyboard support:

**Enable keyboard support**
Select the check box if you want your users to be able to use the physical keyboard keys to interact with the host. This enables users to press certain physical keys that have been mapped to host AID keys, such as the F1,

SYSREQ, RESET, or ATTN keys. Users can toggle this feature off if they want to use a mapped physical keyboard key to interact with the browser instead.

To use keyboard support in HATS Web projects, you must have a supported Web browser. For the list of supported Web browsers and limitations, see "System Requirements for Host Access Transformation Services" at http://www.ibm.com/support/docview.wss?uid=swg27011794 and "Host Access Transformation Services 9.7 - Known issues and workarounds" at http://www.ibm.com/support/docview.wss?uid=ibm10876092.

**Turn initial keyboard support state on**
Select this check box to make the keyboard enabled when the GUI is initially displayed to the user.

Select from the following options to specify if keyboard support will be limited to function keys displayed on the browser page:

**Support only host functions displayed on host keypad** `Web-only`
Selecting this option will scan the browser page for recognized function buttons or links. If any functions are found, only those are supported. If none are found, all functions are supported. This was the default behavior in some earlier versions of HATS, and applications migrated from these versions will continue to function as they did previously.

**Support all mapped host functions** `Web-only`
Select this option to have keyboard support work with all mapped functions, regardless of which host key buttons or links may exist on the browser page. This is the default behavior for new HATS applications.

## Client locale

Button captions and messages can be displayed in different languages. To acquire the language to display, customize the following client locale settings:

**From the client's default locale**
For Web applications the language used to display button captions and messages is determined by the language specified by the user's browser. For rich client applications, the language is determined by either the default operating system value or a value specified when the rich client environment was started.

**From the server's primary locale** `Web-only`
The language used to display button captions and messages is determined by the locale of the server where the application is deployed.

**Always use the following language**
You can select the language to use for button captions and messages from the drop-down list.

## Connection parameter overrides

HATS allows Host On-Demand session parameters to be used as connection parameters for HATS host connections. For more information about these parameters see the Host On-Demand Knowledge Center at http://publib.boulder.ibm.com/infocenter/hodhelp/v11r0/index.jsp and search on `session parameters`. Information about these parameters is also available in the document at http://www-01.ibm.com/support/docview.wss?uid=swg21251884.

For HATS Web applications, these connection parameters can be overridden on the URL used to access the HATS application. For example, a URL like the following can be entered:

```
http://servername:port/appname/index.jsp?host=9.42.141.26&sessionType=1&port=523&
LUName=LU00001&TNEnhanced=true
```

After the server name, port, and HATS application name, you specify the welcome page index.jsp, followed by a question mark (?) and then the name=value pairs for the parameters, separated by ampersands (&). The values are the text values defined by Host On-Demand, not the values that are used elsewhere in the application.hap file. For example, sessionType=1 for 3270, and sessionType=2 for 5250.

In addition to the Host On-Demand session parameters, the SSLP12FilePath and SSLP12Password parameters can also be overridden. The SSLP12FilePath and SSLP12Password parameters specify the location and password of the PKCS12 keystore file used for SSL connections. For more information, see "Security" on page 139.

For information about how a rich client application user can override connection parameters using the **Properties** item on the pop-up menu of the Applications view, see "Connection Parameters" on page 78. For information about programmatically starting a rich client application instance using connection parameter overrides, see the *HATS Rich Client Platform Programmer's Guide*.

To control which connection parameters, if any, can be overridden, select one of the following radio buttons:

**Allow all default connection parameters to be overridden by client requests**
: Selecting this button allows all connection parameters to be overridden except for those listed and selected in the table.

**Do not allow any default connection parameters to be overridden by client requests**
: Selecting this button allows no connection parameters to be overridden except for those listed and selected in the table.

You can provide exceptions to the choice you select by adding or removing connection parameters. To get a list of available connection parameters, click the **Add** button.

## Selecting the type of connection to use when overriding parameters

When connection parameters are overridden at run time, HATS uses the overrides to create a transformation connection that differs from the one defined when the application was created. The choice of which connection HATS uses as a template for creating this new transformation connection is based on whether you specify the **connectionName** connection parameter:

- If you do not specify the **connectionName** parameter, HATS creates a basic transformation connection based on the HATS application's default connection.
- If you specify the **connectionName** parameter, but the value is not a valid connection name, HATS creates a basic transformation connection based on the HATS application's default connection.
- If you specify the name of a background connection as the value of the **connectionName** parameter, HATS creates a complete transformation connection based on the named background connection.

- If you specify the name of the HATS application's default connection as the value of the **connectionName** parameter (usually `main`), HATS creates a complete transformation connection based on the HATS application's default connection.

When HATS creates a basic transformation connection, certain functions are disabled even if they were enabled for the connection chosen as the template connection. With the basic connection, pooling is always disabled, and the connect and disconnect time-outs are ignored, as are the connect and disconnect macros. The HOD connection parameters for the template connection, such as the host name, and port are copied from the template connection, and then any supplied connection parameter overrides are applied to create the new basic transformation connection. Notice that basic connections are always based on the application's default transformation connection.

When HATS creates a complete transformation connection, all connection settings associated with the connection named by the **connectionName** parameter are used, including pooling settings, connect and disconnect macros, and connect and disconnect time-outs. Your connection parameter overrides, if any, are then applied to create the pool specification used to create the new transformation connection.

If you specify the name of a background connection as the value of the **connectionName** parameter, and specify no other overrides, you effectively switch which connection will be used as the transformation connection for that application instance.

All HATS connections are based on a pool specification. A pool specification is an internal object that describes the properties of all connections created from it. The HATS toolkit actually creates pool specifications, and the runtime uses these connection files, like main.hco, to create actual Telnet connections at run time. When you specify connection parameter overrides, the HATS runtime will dynamically create a new pool specification object, if needed, to describe the properties of the new connection. Pool specifications are visible in the HATS administration panel for Web applications. Pool specifications dynamically created when parameter overrides are specified, regardless of whether the **connectionName** parameter is specified, are automatically destroyed when the last active connection is terminated, provided that pooling is not enabled.

Connection parameter overrides can be different for each user accessing a HATS application. For example, if each user specifies a different LUName override, HATS runtime creates a different pool specification for each user. Ensure that pooling is disabled for the original connection so the pool specification is automatically destroyed. If multiple users specify exactly the same overrides, only one pool specification will be created, and reused to create many identically-configured Telnet connections. For example, if an entire department of users specifies a particular host name override, only one new pool specification will be created and destroyed when no more connections are using that specification.

To be able to specify the **connectionName** parameter as a connection parameter override, the parameter must be enabled for overrides on the Connection Parameter Overrides page on the **Other** tab in the project settings editor.

## The disconnectOnClose parameter

HATS provides automatic disconnect functions to automatically disconnect the host if the browser is closed before the session disconnects. For more information see "Automatic Disconnect and Refresh **Web-only** " on page 109. For applications not

using the automatic disconnect functions, the disconnectOnClose parameter performs a similar function. It enables the HATS server to reset the host Telnet session immediately if the browser is closed before the session is manually disconnected from the host by the user. Disconnection normally occurs when the user clicks Disconnect. Resetting the Telnet session is accomplished by having a hidden frame in the browser that launches a new browser when the original browser is closed. The new browser window sends a disconnect signal and then closes itself.

The functionality is only enabled when you specify `disconnectOnClose=true` and you are using a supported version of Internet Explorer. Specification of the disconnectOnClose parameter is case sensitive. A sample URL request follows:

```
http://hostname/appname/index.jsp?disconnectOnClose=true
```

where appname is the name of your HATS application.

The following conditions are true when using the disconnectOnClose parameter:

- The disconnectOnClose parameter works if all navigation is done through the HATS session frame (including clicking on links within the page).
- The disconnectOnClose parameter does not work if the user browses away using the address bar or a Favorites entry. If the user browses away by these means, and does not navigate back to the HATS application before closing the browser, the automatic disconnect does not occur.
- The disconnectOnClose parameter does not work if frame support is disabled or if pop-up prevention software is installed.
- The disconnectOnClose parameter does not work in a portlet environment because of its use of HTML frames.

## Asynchronous Update RCP-only

For HATS rich client applications the asynchronous update function is enabled by default. You can disable the function by clearing the **Enable asynchronous updates** box. The primary purpose of the asynchronous updating mechanism is to automatically update the GUI view if the underlying host screen changes because the host sent a new screen. This provides the following advantages:

- Improvement in response time for the user of the rich client application.
- The rich client interface is more like an emulator, in that screen refreshes occur in the rich client environment as they would in an emulator.

## Automatic Disconnect and Refresh Web-only

For HATS Web applications, the automatic disconnect and refresh functions can be performed by either a client pull method using an AJAX implementation or by a server push method using an applet implementation. To enable the AJAX implementation, select the **Enable client pull (AJAX)** option. To enable the applet implementation, select the **Enable server push (applet)** option. To disable both of these functions, select **Disabled**. The default is **Disabled**.

### Using the client pull (AJAX) method

For HATS Web applications, you can configure the client pull (AJAX) implementation method, which includes the auto-disconnect and auto-refresh functions.

The auto-disconnect function provides the ability to detect when the browser has been out of contact with your HATS application for a predefined period of time.

When detected, disconnect processing is performed to clean up resources associated with the browser session and thereby reduce the load on the server. To do this, the disconnect event is run, and the disconnect macro (if configured) is run.

The purpose of the auto-refresh function is to automatically refresh the Web page view of your HATS application if the underlying host screen changes because the host sent a new screen. This eliminates the requirement for the user to click the Refresh button to see asynchronously changed screens from the host.

With auto-refresh enabled, a Web page view is not automatically updated in the following cases:

- The user has updated the Web page view but not submitted the change.
- Auto-refresh has been disabled for the specific screen. Auto-refresh may be disabled for a specific screen by adding the following lines just after the </HATS:Form> tag in the screen transformation file.

```
<script>
disableBrowserRefresh(formID);
</script>
```

Unlike the server push (applet) implementation method, the client pull (AJAX) method does not require the use of certain browsers, but instead supports all browsers supported by HATS. Also unlike the server push (applet) method, an additional connection to the server is not required, thus simplifying configuration for the HATS Administrator. The client pull (AJAX) method is mutually exclusive of the server push (applet) method. If either of the client pull (AJAX) functions are enabled, then the server push (applet) configuration is ignored.

**Notes:**

1. The client pull (AJAX) method, including the auto-disconnect and auto-refresh functions, is supported only for HATS Web, including JSR 286 portlet, screen transformation applications.
2. This method is not supported in conjunction with the following HATS functions:
   - JSR 168 portlet projects
   - WebFacing projects
   - Linked HATS/WebFacing projects
   - Integration Objects
   - EJB projects
   - Web services
   - Rich client projects
3. HATS Web applications optimized for mobile devices do not provide keyboard host key support. As a result, auto-refresh support overwrites any data that the user has entered into the GUI view of the HATS application if a new host screen is asynchronously received from the host application.
4. AJAX polling from a browser running on an iPhone or iPod touch device stops when you switch from the browser to another application. As a result, when you switch from a browser accessing HATS to another application on the device, HATS disconnects the browser session after the **Time to wait for disconnect (seconds)** interval has passed. You may wish to increase this interval or disable the auto-disconnect function for HATS applications accessed from iPhone or iPod touch devices. For more information, see “Client pull (AJAX) settings” on page 111.

**Client pull (AJAX) settings:**

To configure the auto-disconnect and auto-refresh functions, use the following settings:

**Polling interval (seconds)**
The interval in seconds at which the browser will poll the HATS application to restart the disconnect timer, if enabled, and check for host screen updates. The value is specified in seconds to one decimal place. The minimum value is 1.0 second. The default is **5.0** seconds. If **Disconnect** is selected, then the polling interval value must be less than the value in the **Time to wait for disconnect (seconds)** setting by at least 1 second.

**Note:** If an HTTP session idle timeout is configured by selecting the **Set timeout** option in the WebSphere Application Server **Session management** settings for the application server, browser polling of the HATS application effectively disables the HTTP session idle timeout functionality. Because of this, the HATS runtime takes responsibility for monitoring the HTTP session idle timeout period and initiates a disconnect of the HATS session, including running the disconnect event, when no user activity is seen before the idle time is exceeded.

If auto-disconnect and auto-refresh are both disabled for the current page, when an HTTP session idle timeout is triggered, HATS disconnects the session and does not run the disconnect event.

**Disconnect**
Select this option to enable the auto-disconnect function. If selected, the HATS application initiates disconnect processing if the client has not polled the HATS application within the time specified in the **Time to wait for disconnect (seconds)** field.

**Note:** Disconnect processing is performed to clean up resources associated with the browser session and thereby reduce the load on the server. To do this, the disconnect event is run, and the disconnect macro (if configured) is run

**Time to wait for disconnect (seconds)**
The time in seconds to wait before performing the auto-disconnect function. The value is specified in seconds to one decimal place. The minimum value is 2.0 seconds. The default is **15.0** seconds.

**Refresh**
Select this option to enable the auto-refresh function. If selected, the browser initiates a refresh if there has been no user input and the poll response indicates that the host screen has changed.

## Using the server push (applet) method

**Note:** The server push (applet) method is deprecated. While this method is currently supported, IBM reserves the right to remove this function in a future release. You are advised to use the client pull (AJAX) method instead.

There are two functions performed by the server push (applet) implementation method, also known as the asynchronous update applet:

- Pushing asynchronous host updates to the browser:

  - This enables the user to see host updates automatically without the user clicking the **Refresh** button
- Notification of browser close to HATS server:
  - This allows the HATS server to clean up resources associated with the browser session and reduces the load on the server

When using the asynchronous update applet, in addition to the connection used to send transformed terminal screens from the HATS server to the user's browser, a separate socket connection is established between the applet running on the client and the HATS server. This separate socket connection is used to pass small messages, between the client and server, relating to asynchronous screen refreshes from the host and notification to the HATS server that the browser has been closed. Typically, the amount of traffic over this separate socket connection is very small. The sequence of events which occurs to set up this separate socket connection is as follows:

1. The user enters the HATS application URL in the browser (using a link or typing in the address bar).
2. The HATS server receives the request and sends a response to the browser, which includes the asynchronous update applet Java code.
3. Once the applet is downloaded to the client, it attempts to communicate directly back to the HATS server by opening a separate socket connection. The local port used on the client for making this socket connection is chosen at random from the available unused ports on the client, and is not configurable. The remote HATS server IP name and port information that the applet attempts to connect to are supplied by the server when the applet is downloaded to the client. If a value of 0 is specified for the server's port number in the HATS application configuration on the server, then one random, unused port number above 1024 is chosen on the HATS server and supplied to the applet as a parameter. The port number that has been chosen by the server can be seen by viewing the page source of the HATS transformation in the browser. Search the page source for the value specified for `PARAM NAME="port"`. For more information about the applet parameter settings, see "Server push (applet) settings."

In order to run the applet, a Java-enabled browser is needed. See "System Requirements for Host Access Transformation Services" at http://www.ibm.com/support/docview.wss?uid=swg27011794 for the current list of supported browsers.

**Note:** The applet does not work with HATS portlets. For more information, see Chapter 19, "WebSphere Portal and HATS," on page 387.

**Server push (applet) settings:**

To configure the server push (applet) implementation method use the following settings:

**Host name**

This is the host name or IP address of the server where the HATS application will run. If the server is in a horizontally clustered environment, the host name should be changed by editing the application.hap file after the applications file is deployed to the server.

The browser-side applet uses a TCP/IP socket to communicate with the HATS server. The host name of the server hosting the HATS application is needed to create the socket. If HATS is deployed on the same Web server specified by the browser's URL, then the host name parameter doesn't

need be specified and the applet will use the Web server's IP address to establish the socket connection. If the HATS application is on a different server, the host name parameter must be specified with either a host name or an IP address.

**Port for server socket**
The port on the server that the client should use to communicate with the server. A zero (0) specifies that a random unused port above 1024 is used. You can also specify a range of ports, such as 2043–4544, or a list of ports delimited by commas (,), such as 3045, 1345, 9596.

**Note:** If HATS is vertically clustered (multiple HATS instances on the same physical server) a different port must be used by each of the HATS instances. If you choose to use a value other than zero (0), you must specify at least as many ports as you have HATS instances. As each HATS instance starts, it tries the specified ports until it finds one that is free.

**Time to wait for disconnect (seconds)**
The time, in seconds, that the application waits to disconnect the user after the browser is closed. This can occur if you are away from the browser for some time and you want to ensure the application disconnects. The value is specified in seconds to one decimal place. The default is **300.0** seconds (5 minutes). Valid values are numbers greater than or equal to 300.0 seconds.

Additional browser specific settings as well as non-Windows mask settings can be specified for the applet. These settings can be changed by editing the application.hap file.

When you create a new project, a tag will be generated in the file Web Content/WEB-INF/profiles/application.hap within the <classSettings> tag. Following are the default settings for the applet:

```
<class name="com.ibm.hats.common.AppletSettings">
   <setting name="disconnectDelay" value="300000"/>
   <setting name="enable" value="false"/>
   <setting name="hostname" value=""/>
   <setting name="ie" value="disconnect|refresh"/>
   <setting name="nonWindows" value="disconnect|refresh"/>
   <setting name="other" value="disconnect|refresh"/>
   <setting name="port" value="0"/>
</class>
```

A brief explanation of each parameter:

**disconnectDelay**
This parameter is set from the **Time to wait for disconnect (seconds)** setting in the GUI for the **Enable server push (applet)** settings. Whenever the browser closes or loads a new page (that is, the user browses away from the HATS application), the applet will wait this amount of time before disconnecting the user from the host. If it is set too low, users may be disconnected when going from one page in the application to the next. There is no real danger in setting it too high, but resources on the server and host will be held for a user who has closed the browser until the time period has passed. The default value is 300000 milliseconds (5 minutes).

**enable**
Use this setting to configure which automatic disconnect and refresh update method to use. Specify `AJAX` to configure the client pull (AJAX) method, specify `true` to configure the server push (applet) method, and

specify `false` to disable both methods. The default is **false**. If `true` is specified, a listening socket is established on the given port. Each browser that connects to this application will have the applet tag inserted in the page.

**hostname**
The host name or the IP address of the HATS server must be configured in the application.hap file to allow the applet to connect back to the HATS server.

**Note:** For horizontal clustering, the host name should be changed by editing the application.hap file. Hand editing the .hap file needs to occur after the .ear has been deployed using the WebSphere Application Server Admin (on the server).

**port** Select the port on the server that the client should use for communicating with the server. Zero (0) specifies a random port above 1024; you can also specify a range of ports (for example, 2043-4544) or a comma-delimited list of ports (for example, 3045,1345,9596). The default value is random.

**Notes:**

1. If HATS is behind a firewall, a zero (0) value should not be specified because it can be difficult to let arbitrary ports go through the firewall.
2. For vertical clustering, if HATS is behind a firewall, a port range can be specified. If HATS is on the Web server, zero (0) or a range can be specified instead of a port number. In both cases, it is highly recommended that port numbers are specified individually on the Web servers.

The following browser-related settings determine the features to be used for a given browser or environment:

**nonWindows**
Specify settings for browsers that run under operating systems other than Windows. The possible options are disconnect, refresh, local. The algorithm looks for l d and r and doesn't care about separators; in order for any options to apply, they must be listed in both the operating system setting and the browser setting; (example: nonWindows=dr other=dr). The default value is **disconnect | refresh**.

**ie** Specify settings for all versions of Internet Explorer. The default value is **disconnect | refresh | local**

**other** Specify settings for other browsers (for example, Opera, Mozilla). The default value is **disconnect | refresh**.

The following values may be used for the browser-related settings listed above:

**disconnect(d)**
When the browser is closed, this setting causes the host-side connection from the HATS server to the host to be cleaned up after expiration of the disconnectDelay timer (specified in the application.hap file). This works in any Java-enabled browser.

**Note:** For all browsers except Internet Explorer, when the user browses away from the HATS application, this setting will cause the host-side connection to be cleaned up after expiration of the disconnectDelay timer. For more on Internet Explorer settings see disconnectAlways below.

**disconnectAlways**

This feature enables Internet Explorer to have the same asynchronous update applet-disconnect behavior as all other browsers (non-Internet Explorer). To illustrate:

- Internet Explorer default behavior (disconnect specified in the application.hap file): Close of the browser causes the HATS server to clean up the connection after expiration of the disconnectDelay (specified in the application.hap file) timer. User browse-away allows you to browse back and reclaim the session regardless of the value of the disconnectDelay timer.
- All other browsers default behavior: Both close and browse-away causes the HATS server to clean up the session after expiration of the disconnectDelay timer (specified in the application.hap file).

This feature enables you to specify that for Internet Explorer, the session should be cleaned up after the disconnectDelay timer has expired when either the browser is closed or the user browses away from the HATS application.

To change the Internet Explorer behavior, replace `disconnect` with `disconnectAlways`. The parameter list for Internet Explorer then appears as follows:

```
<setting name="ie" value="disconnectAlways|refresh|local"/>
```

.

**refresh(r)**

Allow the server to automatically refresh the page when new content becomes available; this should work in any Java-enabled browser.

**local(l)**

Refresh only the presentation space of the page; this only works reliably in Internet Explorer V7 and later releases of Internet Explorer V6; it may cause JavaScript errors and browser failures in other environments.

To enable asynchronous update applet client-side tracing, you must specify an HTML parameter, `appletJavaConsoleTrace=true`, on the HATS entry servlet Web address. For example,

```
http://myhost/myhats/entry?appletJavaConsoleTrace=true
```

**Notes:**

1. This is client-side tracing as opposed to the HATS server-side tracing and complements the server-side tracing for debugging applet issues.
2. Enable this trace only when requested by IBM support personnel.

## Global variable overrides

You can use this feature to pass data from the user into the application. For example, you can allow the user to supply a user ID and password, and use them to run a macro.

For HATS rich client applications, the user overrides global variables using the **Properties** item on the pop-up menu of the Applications view. For more information, see “Variables” on page 79.For information about programmatically starting a rich client application instance using global variable overrides, see the *HATS Rich Client Platform Programmer's Guide*.

For HATS Web applications, to override a local global variable, you add an HTML parameter **hatsgv**_*varName=newValue* to the client's HTTP request. To override a shared global variable, the parameter name is **hatssharedgv**_*varName*.

To allow the user to create and set the value of a shared global variable named UserID, you can supply an input field by adding the following statement to the body of the <HATS:Form> tag of a transformation JSP:

```
<INPUT name="hatssharedgv_UserID" type="TEXT" size="10">
```

Any value supplied by the user becomes the new value of the shared variable UserID. If you create the value by JavaScript, instead, and do not want the user to be able to enter the new value, you can use a hidden input field and supply the value yourself:

```
<INPUT name="hatssharedgv_UserID" size="10" type="HIDDEN" value="alice">
```

To create a new global variable or change the global variable's value in the URL, something like the following URL can be used in a GET request:

```
http://hostname/appname/entry?hatsgv_UserID=bob
```

To control which HATS global variables, if any, can be created or changed by a user's client request, click one of the following radio buttons:

**Allow all global variables to be overridden by client requests**
: This button allows you to override all global variables except for any listed and selected in the table.

**Do not allow any global variables to be overridden by client requests**
: If you click this button, you will not be able to override any global variables except for those listed and selected in the table.

You can add a global variable exception by clicking the **Add** button. There you can either type a global variable name in the text box or select it from the drop-down list. Select the **Shared global variable** check box to indicate a global variable that can be shared. For more information, see Chapter 12, “Interacting with global variables,” on page 337.

If you want to edit a table entry, click **Edit**. The **Remove** button will delete the highlighted global variable from the list of exceptions.

**Note:** The default for applications migrated from HATS V4 is to allow all global variables to be overridden by client requests. You can change that here after importing the application. The default for new HATS V5, or later, applications is to disallow any global variables to be overridden by client requests.

## Client settings

You can customize the following client settings:

**Enable HTTP compression** `Web-only`
: Select this box to use HTTP compression to reduce the number of bytes being transferred between the HATS runtime, which is running on the WebSphere Application Server, and the user's browser. This reduces the transfer time between the HATS runtime and the browser (which improves response time) and reduces the number of bytes flowing in the network (which improves network utilization).

**Notes:**

1. To determine if compression is working properly and to see before and after page sizes in bytes, you can enable tracing by updating the trace.UTIL line in the runtime.properties file (or runtime-debug.properties file if running in debug mode in the HATS Toolkit). For example: `trace.UTIL=7`. For more information, see Appendix A, "Runtime properties files," on page 469. In the trace file, search for the runtime.filters.CompressionFilter trace entries. For example:

```
+-------------------------------------+
Text UTIL runtime.filters.CompressionFilter.doFilter()
 17.50.10.140 11/27/06 Servlet.Engine.Transports : 1
enable compression: true 0000CBAx-81SRWWVmQfQ8-_47oK:-1

+-------------------------------------+
Text UTIL runtime.filters.CompressionFilter.doFilter()
 17.50.10.140 11/27/06 Servlet.Engine.Transports : 1
size before: 25315 0000CBAx-81SRWWVmQfQ8-_47oK:-1

+-------------------------------------+
Text UTIL runtime.filters.CompressionFilter.doFilter()
 17.50.10.140 11/27/06 Servlet.Engine.Transports : 1
size after: 4264 0000CBAx-81SRWWVmQfQ8-_47oK:-1
```

2. This feature is not applicable when running the HATS application on WebSphere Portal Server.
3. HATS ensures the "Accept-Encoding" HTTP header contains "gzip" before compressing a page.
4. You must restart the application server if you want changes made to compression-related settings picked up while the application is running on the server.
5. JavaScript (.js) and cascading stylesheet (.css) files are not compressed by this new function. You must configure HTTP compression within your HTTP server to provide compression for these types of files. See the documentation of your HTTP server for more information.
6. This setting can only be specified at the project level. It cannot be specified for individual transformation JSPs
7. If you are migrating projects from HATS 5.0.x, 6.0, or 6.0.1 you must manually update the web.xml file with appropriate tags. See "Special migration considerations" on page 15 for more information.

**Enable Minify JavaScript Feature** `Web-only`

Select the check box to enable the minify JavaScript file feature on the HATS project. The minify JavaScript feature will remove the unnecessary code contents like code comments and extra space (formatting), convert variables into shorter variables and so on, without affecting the processing of the resources by the browser in HATS.

**Notes:**

1. If **'Compress default javascript'** option is selected, then it will compress the HATS default JavaScript file, which is under the below HATS project folder :

```
\Web Content\common
\Web Content\common\hatsdojo
\Web Content\common\scripts
\Web Content\hatsadmin\scripts
```

2. If **'Compress all javascript'** option is selected, then it will compress all the 'HATS JavaScript files which are located at the below 'HATS project' folder:

```
\Web Content\
```

3. Users who do not want to get compressed javascript file, can set the property, **'avoidCompressJS'**, in the Source tab of Project, to manually mention the names of the JavaScript files with relative path in value (separated by comma (,)). (Already compressed or minified **JS** file names (abc.min.js) must be given in values to avoid being compressed again, as they are already compressed). For example:

   ```
   <setting name="avoidCompressJS" value="\bootstrap.min.js,\common\bidishape.js"/>
   ```

   After selecting these features, while doing an 'Export project', a dialog box showing the 'Compress JavaScript progress bar' will be displayed and the process of compression for the javascript file will begin as per the given options.
4. In a workspace, if a system crash, IDE crash or an abnormal condition occurs during a JavaScript compress process, the javascript files which already got compressed are reverted back to their original stage in the same project with the same workspace by default, when the IDE is restarted. system will take care to revert back original file in same project. (The crash recovery of the project to the original stage will occur only if at least one file related to that project is open at the time of system crash, for which, an 'export project' operation is being carried out by enabling the compress JavaScript option).
5. If an error is displayed on the console for a javascript definition or declaration during the Compress operation, then all the errors should be fixed for the file before re-running the export project operation. An Example of Error during Compress operation is shown below:

   ```
   INVALID_OCTAL_LITERAL. This style of octal literal is not supported in strict mode. at lx
   ```

   The Compress operation of the javascript file has failed due to an error in one of the javascript files. Correct those **JS** and re-run the export project.
6. This feature is applicable when running the HATS application on a WebSphere Portal Server.

**Enable same origin policy protection** `Web-only`

Select this box to prevent CSRF attack on the HATS, which is running on a HATS supported application Server, and the user's browser. This will discard the request that is originated from a URL that has no protocol, or request originated from an unauthorized origin/website.

**Notes:**

1. To protect against CSRF attack, declare the HATS running URL as the 'param-value', in web.xml under the 'HatsCSRFValidationFilter' section, for param-name= target.origin. For example:

   ```
   <param-value>http://localhost:9080/Test/entry</param-value>
   ```
2. To determine if the CSRF validation is working properly, uncheck the box in client setting and attempt a CSRF attack. This will now allow the modification of HATS data by a different source. The same operation will not be allowed from a different origin, if the client setting check-box is selected.
3. This feature is not applicable when running the HATS application on a WebSphere Portal Server.
4. The application server must be restarted for the changes made to origin policy-related settings to be picked up while the application is running on the server.

5. After enabling origin policy, the default URL, */entry, /hatsadmin/admin , /index.jsp* and the default project context path ( http://localhost:9080/Test/ ) will be secure from CSRF attack if the origin policy is enabled.
6. Users can customize to secure more custom URLs. If a user has added a new servlet, then it has to be configured in web.xml file, as shown below to secure it from a CSRF attack.For example:

```
http://localhost:9080/Test/transfer
   <filter-mapping>
   <filter-name>HatsCSRFValidationFilter</filter-name>
    <url-pattern>/transfer</url-pattern>
   <filter-mapping>
```

7. Users can enable multiple origin sources by adding the source list as the 'param-value' while configuring, under the filter 'HatsCSRFValidationFilter' section, against param-name = source.origin. For example:

```
<param-value>http://hats:9081/index, http://hatsapp.com/, http://citi.com</param-value
```

8. For users migrating projects from HATS version V8.0.x, V9.0.x, manual update of the web.xml file is required with appropriate tags. See "Special migration considerations" on page 15 for more information.

**Enable token based protection** `Web-only`

Select this box to prevent CSRF attack on the HATS, which is running on a HATS supported application Server, and the user's browser. This will discard the request even if the attacker is able to bypass the same origin policy.

**Notes:**

1. To determine if the CSRF validation is working properly, uncheck the box in client setting and try a CSRF attack. This will permit the modification of HATS data by a different source, if the 'Enable same origin policy' option is unchecked. The same operation will not be allowed from a different origin, if this check-box is selected, and if an attacker is able to bypass the origin policy, then a token based protection will not allow to modify HATS data by CSRF attack.
2. This feature is not applicable when running the HATS application on a WebSphere Portal Server.
3. The application server must be restarted for the changes made to token based settings to be picked up while the application is running on the server.
4. Token based check, when enabled, will secure the default URL /entry, /index.jsp and the default project context path from CSRF attack, and if the origin policy is enabled then */entry, /hatsadmin/admin , /index.jsp* and the default project context path ( http://localhost:9080/Test/ ) will be secured from CSRF attack.
5. Users can customize to secure more custom URLs. If a user has added a new servlet, then it has to be configured in *web.xml* file, as shown below to secure it from a CSRF attack, and has to add the INPUT HIDDEN FORM field name *HatsCSRF* in the respective **jsp**page and the value can be appended by the *HatsCSRFValidationFilter* token..For example:

```
http://localhost:9080/Test/transfer
   <filter-mapping>
   <filter-name>HatsCSRFValidationFilter</filter-name>
    <url-pattern>/transfer</url-pattern>
   <filter-mapping>
<INPUT TYPE="HIDDEN" NAME="HatsCSRF" VALUE="">
```

6. For users migrating projects from HATS version V8.0.x, V9.0.x, manual update of the *web.xml* file is required with appropriate tags. See "Special migration considerations" on page 15 for more information.
7. If both of the protection features have been enabled, then unknown sources cannot modify HATS data, and if an attacker is able to bypass the same origin policy, then the token based protection will restrict the modification of HATS user data.

**Suppress sending unmodified fields** `Web-only`

Select this box to specify that HATS should not send modified input field data to the host when the contents of the field are identical to the data supplied by the host.

Clear this box to specify that HATS should send modified input field data even when the contents of the field are identical to the data supplied by the host. For example, if the host filled a field with `ABC` and the user typed `ABC` into the field, the typed data will be returned to the host.

**Note:** This setting can only be specified at the project level. It cannot be specified for a single transformation JSP.

**Enable XSS Policy protection** `Web-only`

In the *web.xml* file, provide the below instructed value to protect against XSS attack on the HATS, which is running on a HATS supported application Server, and the user's browser. This will discard XSS attack.

**Notes:**

1. To protect against XSS attack, in the web.xml file, under the filter 'HatsHeaderSecurityFilter' section, update the 'param-value' from "NO" to "YES". Listed three policies can be enabled and disabled independently by giving respective value "YES" or "NO".
   - "Content-Security-Policy"
   - "X-Content-Type-Options"
   - "X-XSS-Protection"

   The policies are disabled by default. For example:

   ```
   <param-value>YES</param-value>
   ```
2. To determine if the XSS validation is working properly, set the *'param-value'* to "NO" in *web.xml* 'HatsHeaderSecurityFilter' section
3. This feature is not applicable when running the HATS application on a WebSphere Portal Server.
4. The application server must be restarted for the changes made to XSS policy protection-related settings to be picked up while the application is running on the server.
5. Users can customize to secure more custom URLs. If a user has added a new servlet, then it has to be configured in *web.xml* file, as shown below to secure it from an XSS attack.For example:

   ```
   http://localhost:9080/Test/transfer
      <filter-mapping>
      <filter-name>HatsHeaderSecurityFilter</filter-name>
       <url-pattern>/transfer</url-pattern>
      <filter-mapping>
   ```
6. For users migrating projects from HATS version V8.0.x, V9.0.x manual update of the *web.xml* file is required with appropriate tags. See "Special migration considerations" on page 15 for more information.

**Include protected, read-only labels in default tab order** `RCP-only`

Select this box to specify that the default tabbing order of fields on the transformation view should include protected, read-only fields.

Clear this check box to specify that the default tabbing order should not include protected, read-only fields.

**Enable automatic field advance**

Select this box to specify that when a user completely fills an input field with data, focus automatically advances to the next input field.

For Web applications, you can selectively disable this function for an individual transformation by adding the following lines just after the `</HATS:Form>` tag in the transformation file. To selectively enable the function specify `true` instead of `false`.

```
<script>
autoAdvance = false;
</script>
```

For rich client applications, you can selectively disable this function for an individual transformation by overriding the needsAutoAdvanceSupport method on the individual transformation as shown below. To selectively enable the function return `true` instead of `false`.

```
public boolean needsAutoAdvanceSupport(boolean preferredValue) {
return false;
}
```

**Note:**

For DBCS considerations when using this setting see "Enable automatic field advance" on page 461.

**Include host and non-host input fields**

Select this box to specify that when auto advance is enabled, focus automatically advances to the next input field in the order of the input fields on the transformed screen without regard to the order of the input fields on the host screen.

Clear this box to specify that when auto advance is enabled, focus automatically advances to the next input field in the order of the input fields on the host screen.

**Initial cursor position**

The position of the host application's cursor normally controls the initial input focus location for your screen transformation. In some limited cases, it might be desirable to have the host application's cursor location ignored, and instead give the initial focus to the first item on the transformation. This is intended for highly-customized transformations where the order of host input fields has been changed. With this feature enabled, the initial focus placement will be the first item on the screen transformation rather than the input field containing the host cursor.

**Notes:**

1. The **Initial cursor position** setting does not appear in the GUI as a configurable setting. It must be set in the source of the transformation file.

   For Web applications, you can enable this feature for a transformation by adding the following lines just after the `</HATS:Form>` tag in your transformation:

```
<script>
initialInputFocusFromCursor = false;
</script>
```

For rich client applications, you can enable this feature for a transformation by overriding the isInitialFocusAtCursorPosition method on the individual transformation as shown below.

```
public boolean isInitialFocusAtCursorPosition() {
return false;
}
```

2. Be certain to test your transformation carefully. Many host applications rely on the cursor position for proper behavior. For example, a macro button placed on a transformation may cause the invoked macro to begin entering data in the wrong location on the host application, since the cursor may be in a different location than expected.

**nextFieldForDropDown** `Web-only`

Use this setting to specify that the cursor position be moved to the next input field when a selection is made from a drop-down list. The default for new projects created in HATS V7.5.0.2, or later, is **true**. The default for projects created before HATS V7.5.0.2 is **false**.

**Notes:**

1. This setting does not appear in the GUI as a configurable setting. It must be set in the source of the application.hap file as shown below.

```
<class name="com.ibm.hats.common.RuntimeSettings">
        <setting name="nextFieldForDropDown" value="true"/>
</class>
```

2. This setting is effective only when **Enable automatic field advance** is selected.

**Overwrite mode (initial)**

Select this box to initially enable overwrite mode (if it is supported by the browser). If enabled, text entered into an input field overwrites text at the cursor position one character at a time. If not enabled, text entered into an input field is inserted at the cursor position pushing existing text ahead. The user can toggle from this initial setting using the Insert key.

**Note:**

For DBCS considerations when using this setting see "Overwrite mode (initial)" on page 461.

**Select all text on focus**

Select this box if you want all text in a field to be selected when the field receives focus, which is typical behavior for a Web application. Clear this box if you want no text selected when the field receives focus which is typical behavior for a terminal emulator.

**Notes:**

1. For Web applications:
   - The default is selected.
   - This setting does not affect the **Overwrite mode (initial)** setting behavior.
   - This setting is only valid when Internet Explorer is used as the browser for the application.

2. For rich client applications:
   - The default is cleared.
   - When selected, this setting functions like the **Overwrite mode (initial)** setting in that characters are overwritten as a user types into the field.
   - Text is selected only when the keyboard is used to tab into the field. Text is not selected when clicking the mouse in the field.
3. For DBCS considerations when using this setting, see "Select all text on focus" on page 461.

**Enable busy page** `Web-only`

Select this box to display a busy-page message when multiple requests are submitted by the user before processing has completed on the initial request. Clear this box if you do not want a busy-page message displayed. If cleared, you cannot submit any more requests until the server returns a response.

**Enable arrow key navigation** `RCP-only`

Select this box to allow use of the up, down, right, and left arrow keys to navigate to non-protected host fields on the rendered screen. If the Rendering tab, Field widget, **Allow cursor positioning on protected fields** setting is also selected, the arrow keys can be used to navigate to both non-protected and protected host fields. This is typical behavior for a terminal emulator application. Clear this box if you do not want to allow use of the arrow keys for navigation.

You can selectively disable this function for an individual transformation by overriding the needsArrowKeyNavigationSupport method on the individual transformation as shown below.

```
public boolean needsArrowKeyNavigationSupport(boolean preferredValue) {
return false;
}
```

To selectively enable the function return `true` instead of `false`.

**Notes:**

1. Navigation is always based on host field positions, not on the UI position of a field.
2. When this function is used in conjunction with screen combinations, the algorithm to calculate the next input field will first attempt to navigate to an appropriate field from the same screen as the current field. If an appropriate field does not exist, the next screen (based on the ID of the screen) will be searched to find an appropriate input field.

**Enable type-ahead support** `RCP-only`

Select this box to enable type-ahead support. If enabled, the user can begin typing data intended for input fields on the next screen (or screens) sent by the host, before they are received and processed by HATS. As the next screen (or screens) are received, HATS sends the previously typed data (type-ahead data) including any keys that submit the input to the host.

If screen customizations exist for these screens, all actions defined in the customization are executed, including applying transformations. However, the type-ahead data is not actually placed into the rendered transformation before being submitted to the host. Instead, it is sent directly to the host as previously typed, bypassing the transformation completely. In this manner, this functions much like a terminal emulator's type-ahead support. What

this means to the user is that type-ahead data must conform to what each native host screen expects, not what the applied transformation expects.

If you clear this box, the **Type-ahead field** setting is cleared in **Operator Information Area** on the **Rendering** tab. For information about displaying the type-ahead data in the operator information area, see "Type-ahead field" on page 100.

**Notes:**

1. Only host aid keys along with their host data, for example, `userid[tab]password[enter]`, and host navigation keys, for example, **tab** and **home**, can be typed ahead.
2. Application keys and commands, for example, the Disconnect command, which is the default command for the key sequence **CTRL+D**, cannot be typed ahead.
3. Key remapping at the RCP Runtime Extension plugin level is supported for host keys and commands in type-ahead mode.

## Macro Content Assistance

Use these settings to select which macros to enable for content assistance. While editing a macro on the **Source** tab of either the Macro Editor or the Visual Macro Editor, press Ctrl+Space to invoke content assistance.

**All the macros**
Select this radio button to enable content assistance for all of the macros in the project.

**Selected macros**
Select this radio button to enable content assistance for only those macros selected in the list of macros.

**Macros**
Select which macros to enable for content assistance.

**Select All**
Enabled only when the **Selected macros** radio button is selected. Click this button to select all macros in the list.

**Deselect All**
Enabled only when the **Selected macros** radio button is selected. Click this button to clear the selection of all macros in the list.

## Portlet settings

These settings apply only to HATS standard (JSR 168 or JSR 286) portlet projects. They can be used to help develop a portlet communication solution. For more information, see "Portlet communication" on page 391. For considerations when using bidirectional language support, see "Portlet support" on page 449.

### JSR 168 portlets

Use these settings to define properties received through WebSphere Portal from other JSR 168 portlets (HATS or non-HATS) on the portal server. Click **Add** to add a new property. In the Property field enter the name of the property that is being received. The name must match the property name used by the sending portlet, for example, the same property name used by the **Send global variable** action in a HATS JSR 168 sending portlet. In the Global variable field enter the name of the global variable to use to store the value of the received property. If the global variable is indexed, enter the index to use. Select the **Shared** check box if the global variable is a shared global variable.

When you click **Finish**, the property is added to the project's application.hap file. In addition, a Web Services Description Language (WSDL) file is created, if one does not already exist, and a receive property is defined in the WSDL file. The name of the WSDL file is <project_name>.wsdl. The WSDL file is required to wire the portlets using the WebSphere Portal wiring tool. The portlet.xml file is updated to reference the location of WSDL file.

If you click **Remove**, the property is removed from the project's application.hap file, but it is not removed from the WSDL file. You can edit the WSDL file and use the WSDL editor to remove the property.

**Note:** If you do not remove the property from the WSDL file, the HATS portlet runs properly as-is.

### JSR 286 portlets

Use these settings to define events received through WebSphere Portal from other JSR 286 portlets (HATS or non-HATS) on the portal server. Click **Add** to add a new event. In the Event name field enter the name of the event that is being received. The name must match the event name used by the sending portlet, for example, the same event name used by the **Send global variable** action in a HATS JSR 286 sending portlet. In the Global variable field enter the name of the global variable to use to store the value of the received event.

**Note:** Event names within a single portlet must be unique.

Click **Advanced** to set more options if necessary.

On the Advanced Options page, specify either **Overwrite the global variable with the new value**, or **Overwrite a specific index of the global variable with the new value**, including the **Index** to overwrite and the **Event payload type** of the object. Use the **Browse** button to assist in entering a valid value type. The type you select must be Serializable. If not, an error message is displayed. The type also must match the type used by the sending portlet, for example, the same type used by the **Send global variable** action in a HATS JSR 286 sending portlet. Select the **Shared** check box if the global variable is a shared global variable.

**Note:** The **Event payload type** specification is supported only for JSR 286 portlets. If you manually add the type parameter to the application.hap file source in a JSR 168 project, it is ignored.

When you click **Finish**, the event is added to the project's application.hap file, and the portlet.xml file is updated. No WSDL files are involved with JSR 286 portlet communication.

## Source

The **Source** tab displays the tags and values in the application.hap file for all the settings you selected, or for which you took the defaults, in your project. As you make changes on other tabs in the project editor, the tags and values displayed under the **Source** tab change to match.

You can also make changes to the tags and values in the application.hap file directly by editing the source under the **Source** tab, and they will be reflected on the appropriate tabs of the project editor.

## Using HATS preferences

You can use HATS preferences to control the appearance and behavior of HATS Toolkit. These preferences make HATS Toolkit more accessible or simplify programming tasks. To work with HATS preferences, click **Window > Preferences** and select **HATS** in the tree view. For quick help on any preference, place your cursor on it and press **F1**.

You can modify these preferences:

**Hide file extensions for known file types**
By default, the **HATS Projects** view does not display common file extensions such as .jsp. For example, all the files in the **Templates** folder have the .jsp extension. Clear this check box if you want the **HATS Projects** view to display all file extensions.

**Show HATS tips**
HATS tips provide helpful suggestions for new users. If you do not want to see HATS tips as you use HATS Toolkit, clear this check box.

**Show migration warning**
You can decide whether HATS should display a migration warning when it finds projects which have yet to be migrated to HATS V9.7 projects.

**Prompt to display terminal window**
You can decide whether HATS should display a dialog asking you if you want to turn on the display terminal when you test your project using Debug on Server (for Web projects) or Debug (for rich client projects). This can also be turned off by selecting the check box on the dialog when it pops up.

**Show extract hidden fields warning**
You can decide whether HATS should display a dialog warning you when it encounters a hidden field on a captured screen when one of the following events occurs:

- Extracting data during the recording of a macro using a live terminal
- Creating screen customization criteria using a captured screen
- Adding a new Extract global variable action using a captured screen

**Show fully qualified Java class name for rich client artifacts**
Select this box to show the fully qualified package name for HATS Java source artifacts (for example rich client templates and rich client transformations) in the **HATS Projects** view. By default this box is not selected and the **HATS Projects** view will show only the Java class name for the artifact. For example, assume a rich client project contains a transformation with the fully qualified name `myProj.transformations.SignOn`. By default, only `SignOn` will be displayed under the **Transformations** folder of the **HATS Projects** view. If this box is selected, `SignOn (myProj.transformations)` is be displayed.

This preference has no affect on what is displayed under the **Java Source** folder of the **HATS Projects** view.

**Show incompatible JRE warning**
If HATS detects an incompatible JRE, it can warn you with a message. Select this box to display a warning dialog that the current JRE is incompatible with HATS. You must change the JRE manually. Clear this box to suppress the warning dialog.

**Perform license settings check on workbench startup**

HATS checks the license settings of projects in a workspace on startup and prompts the user to update the license settings if any of the projects don't match the master license settings. Clear this check box to disable checking the license settings at startup.

**Selection filters**

This list of check boxes enables you to select which types of files you want displayed in the **HATS Projects** view. For example, if you never work with common images or style sheets, you can clear that check box and they will not appear in the **HATS Projects** view. If you have novice users and you do not want them to work with macros or Java source files, you can clear those check boxes and those files will not be displayed.

**Host terminal Preferences**

**Prompt for all macro prompts before debugging a macro**

If selected, a dialog will appear when you first debug your macro, and will allow you to enter values for all prompts at once. If not selected, you will be prompted with a dialog that requests only the value for the current prompt that your macro is stepping through. You will have to enter a value for every prompt, one-by-one. This preference is only used when debugging a macro, it has no effect when playing your macro in HATS Toolkit or when running of your macro or HATS application.

**Show macro prompt/extract name warning**

Select this box if you want HATS to warn you when you have entered a macro prompt or extract name that is not valid for use with Integration Objects.

**Fixed font size**

This preference is designed for users with impaired vision. You can specify that the size of the text on the host terminal should not change when the window is expanded or contracted. For example, if you require large type, you can set a large fixed font size and use the scroll bars to view different parts of the host terminal screen.

**Screen capture highlighting colors**

Colors are used to identify input fields, protected fields, hidden fields, and screen regions that match patterns. Use these preferences to change the colors that are used for these purposes.

There are more preferences which govern other behaviors related to HATS. To use these preferences, expand **HATS** in the preferences tree and click one of the sub categories:

**Accessibility**

**Alert recognition errors with audible alarm**

Select this box to play an audible alert (beep) when there is a recognition error, for example, on the Subfile component settings page. This is useful if the developer must be alerted to use screen reader software to read the message area.

**BMS Import**

These settings are only used when BMS source files are added to the **Maps** folder within the HATS project without using the BMS Import wizard.

**Host code page**

The code page used for the BMS maps being imported.

**BMS file code page**
The code page used for the BMS file when it was transferred to the workstation.

**Default Java Packages**
Use this panel to configure default Java package names for HATS artifacts that are Java source files (rich client transformations, rich client templates, rich client macro handlers, business logic, custom widgets, custom components, and RESTful services). A package name makes up the first part of a fully qualified Java class name (for example, for the Java class name `com.myCompany.MyClass`, the package name is `com.myCompany`)

When a package name is created for a new artifact, the name of the project is substituted for `{project name}`. Formatting on this substitution occurs to ensure a valid package name is created (for example, spaces are converted to underscores).

**Host Simulation**

**TCP/IP Port Range**
During trace recording and playback, HATS host simulation function uses a range of ports to record or playback host communication with the host terminal or your HATS application. A range of ports is used to allow trace recording and playback on multiple sessions concurrently. Use these fields to set the range of ports. The default starting port in the range is **7021**, and the default ending port is **7050**.

**Time Delay Settings**

**Delay** Use the options in this drop-down to set the time delay for host simulation to wait during playback before responding to requests from the host terminal or your HATS application. Options are **No Delay** (host simulation responds with the next screen immediately), **Random** (host simulation waits a random delay, within defined minimum and maximum times, before responding) , and **Actual** (host simulation waits the amount of time it actually took the screen to appear during recording). The default is **No Delay**. If you select **Random**, use the **Minimum** and **Maximum** fields to set delay value ranges.

**Minimum (ms)**
If the **Delay** setting is **Random**, use this field to set the minimum delay in milliseconds that host simulation will wait before responding with the next screen during playback.

**Maximum**
If the **Delay** setting is **Random**, use this field to set the maximum delay in milliseconds that host simulation will wait before responding with the next screen during playback.

**Integration Object**

**Automatically generate Integration Object when saving macro**
Select this check box if you want an Integration Object to be generated each time you save a macro. Clear the check box if you prefer to create Integration Objects manually. See Chapter 13,

"Using Integration Objects," on page 341 for information about creating Integration Objects from macros.

**Automatically generate EJB Access Bean in HATS EJB Project when an Integration Object is created**
Select this check box if you want an EJB Access Bean to be generated in your HATS EJB Project whenever you create an Integration Object. If you have selected the **Automatically generate Integration Object when saving macro** preference, an EJB Access Bean will be created along with the Integration Object when you create or modify a macro.

**Template to be used for Integration Object generation**
If you create new Integration Object templates, as described in the *HATS Web Application Programmer's Guide*, use this preference to specify the template to be used when you create Integration Object file.

**Template to be used for Integration Object BeanInfo generation**
If you create new Integration Object templates, as described in the *HATS Web Application Programmer's Guide*, use this preference to specify the template to be used when defining an Integration Object BeanInfo file.

**Target Platform** `RCP-only`

**Prompt to install HATS runtime plug-ins in target platform (rich client only)** The HATS runtime plug-ins must be installed in a target platform in order to test a HATS rich client plug-in in the local test environment for the target platform. When you create a new project and select a target platform, or later change the target platform, the HATS Toolkit will detect if the HATS runtime plug-ins are installed on the selected target platform. Select this box to have the toolkit prompt you to have the runtime plug-ins installed if they are not detected in the target platform.

**Transformation**
These preferences are used when you create a new blank transformation.

**Include a Free Layout Table (Web only)**
When you create a blank transformation, the default height and width of the table is retrieved from this setting. By default, this setting is initialized to 100% unless you specify something different. You can also specify measurements in pixels by selecting **Pixels** from the drop-down menu.

**Note:** If a host component is inserted after the table in the transformation, the rendered component will not be displayed when run on server because the table uses 100% of the screen

**Except when the project is optimized for mobile devices**
Select this preference to prevent including a free layout table in new blank transformations in projects that are optimized for mobile devices.

**Open the Properties view after inserting a host component**
Select this box to open the Properties view after inserting a host component in a transformation.

**Visual Macro Editor**

**Show screen actions by default**
Select this preference to show actions on macro screen objects by default. Clear this preference to hide the actions by default. You can click the action toggle on the macro screen to show/hide the actions for the specific screen. This preference only affects macros opened in the Visual Macro Editor for the first time. The default is selected.

**Restrict number of screen actions shown**
Specifies the maximum number of actions displayed for a macro screen object. This is useful for complex macros where the interface is too complex if all actions are shown. The default is **5**.

When you modify your HATS preferences, you can click **Apply** to see the results immediately in HATS Toolkit. Click **OK** to save your new preferences and close the Preferences window, or **Cancel** to exit without saving. If you click **Cancel**, only those changes you have made since you clicked **Apply** are cancelled.

You can return your HATS preferences to their initial values by clicking **Restore defaults**.

If you configure your HATS preferences on one HATS Toolkit, and you want to copy them to other development workstations, from the Rational SDP menu bar select **File > Export > General > Preferences** to save your preferences as an .epf file. Transfer the file to each workstation where you want to use it, or make it available over your network. Select **File > Import > General > Preferences** to import the preferences file.

## Use of other Rational SDP preferences

HATS Toolkit appearance and behavior can also be affected by using other preferences available with Rational SDP.

For example, you can use the Label Decorations preference to enable other toolkits such as Java Compile and CVS to decorate HATS labels and images displayed in the HATS Projects view. When there is an error, a red-mark can be displayed next to the file. When checking out files from a CVS repository, version numbers can be displayed next to the file names. To access the Label Decorations preference in Rational Application Developer:

1. From the menu bar select **Window > Preferences**.
2. Expand the **General > Appearance** tree view.
3. Select **Label Decorations**.

   **Notes:**

   a. Decoration preferences are similarly located in other versions of Rational SDP.

   b. For more information see the Rational SDP Help system and search on `Label decorations`.

# Chapter 6. Managing connections

A connection is a set of parameters, stored in an .hco file, used by HATS to connect to host applications. There are two types of connections in HATS, default (also referred to as transformation) and background. Each HATS application has one default connection for the host application whose screens HATS will transform. Background connections are any connections in a HATS application other than the default connection, and they can be used by Integration Objects or a Perform macro transaction screen-event action. HATS does not transform screens from background connections. It is possible, however, to dynamically choose which connection will be treated as the default connection. For more information, see "Selecting the type of connection to use when overriding parameters" on page 107.

Background connections can be different connection instances to the same host as the default connection, or they can be connections to entirely different backend hosts. Background connections allow you to interact with other hosts. You can gather data, enter data, or exchange data between hosts using prerecorded macros. This data can be combined with the default connection as well. In this way, you can automate transactions on other hosts as well as transform the default connection for your users. Background connections can be pooled.

VT hosts are limited to background connections. If you use the VT connection as one of the background connections, it can be used for recording macros and running Integration Objects. For more information, see "Perform macro transaction" on page 164.

## Creating a connection

The Create a Connection wizard enables you to create more connection configurations. To access this wizard, you can use the pop-up menu in the HATS Projects view, either the **HATS > New > Connection** or the **File > New > HATS Connection** selection on the menu bar, or the **Create a HATS Connection** icon on the toolbar.

The Create a Connection wizard appears and enables you to select the target project from a drop-down list, name the connection, give it a description, and see where the connection definition is saved. Click **Next** when you have specified these items.

The **Connection Settings** page appears next. The wizard requires basic connection information such as host name, port, terminal type, code page and screen size. For details about these settings see "Basic" on page 132.

You can also specify some advanced connection settings using this wizard. The advanced connection settings that are shown depend on your connection type. If your connection is 5250, you can configure the workstation ID. If you use a 3270E connection, you can configure the LU or pool name. There is no workstation ID or LU setting support for 5250W, 3270, or VT connections. For details about these settings see "Advanced" on page 133.

# Connection editor

The connection editor enables you to perform custom configurations to your connections and has the following tabs: Overview, Basic, Advanced, Printing, Screen Handling, Security, Pooling, Macros, User List, and Source.

## Overview

The **Overview** tab of the connection editor summarizes most of the connection settings you specified when you created your connection. The only item you can modify on this tab is the description of your connection.

Each of the section headings on the **Overview** tab is a link to the other tabs of the connection editor.

## Basic

On the **Basic** tab are the basic settings for the connection including those initially set up with the Create a Connection wizard. The following settings can be modified.

**Host name**
The name of the host to which your connection connects. This is either the name or IP address of the Telnet server. For 5250W connections (see the **Type** field below), this is either the name or IP address of the WebFacing server.

Host On-Demand version 8 included Internet Protocol version 6 (IPv6) support. WebSphere Application Server version 6 also provides IPv6 support. HATS allows you to configure an IPv6 address to take advantage of this functionality on the platforms where it is supported.

**Note:** HATS can only support IPv6 on Linux, Solaris and AIX®. There are limitations in the Windows HATS Toolkit environment with IPv6 addresses. Therefore, you will not be able to use the host terminal feature or test on a local test server if you use this type of address.

**Type** The type of host session for the connection, such as 3270, 3270E, 5250, 5250W, VT hosts (VT52, VT100, VT420_7 and VT420_8).

**Notes:**

1. HATS supports only 3270, 3270E, 5250, and 5250W connection types for the transformation, or default, connection.
2. A background connection can be of any type.
3. For more information about 5250W connections see “HATS connection to a WebFacing server” on page 399.
4. VT220 and VT320 are subsets of VT420. To support VT220 or VT320 hosts in HATS, specify either VT420_7 (if your VT host sends 7 bit commands) or VT420_8 (if your VT host sends 8 bit commands).

**Port** The port number through which the connection connects. Typically, port 23 is used for Telnet servers and port 4004 is used for WebFacing servers. If a different port number is used, you must specify it here.

**Code page**
The code page used for the connection. Select the code page for your backend host.

**Notes:**

1. If you select a bidirectional (bidi) code page, see "Enabling the user to reverse the screen direction" on page 444.
2. If you select the Arabic Speaking code page 420, see "Arabic selective shaping" on page 453 and "Disable entry of Arabic-Western digits" on page 453.
3. If you select the Traditional Chinese code page 937 for a 5250 connection and want to support the use of accented characters, see "Using accented characters for code page 937" on page 431.

**Screen size**
The screen size of the host connection. The screen size defines the number of rows and columns in the host screen.

**Note:** The valid values for screen size change depending on the code page and type of host session you select.

**Use host simulation instead of the live connection**
Select this box to specify that any connect action for this connection, such as **Open host terminal**, **Run**, or **Run on server**, will use host simulation trace playback instead of a live connection. If you select this box, then also select which trace file to use from the drop-down list. See Chapter 17, "Using host simulation," on page 371 for more information.

**Note:** This option is selectable only if a host simulation trace exists in the project.

## Advanced

On the **Advanced** tab are advanced settings for the connection, excluding those initially set up with the Create a Connection wizard. Use the Basic tab for modifying the basic settings: host, session type, port, code page, and screen size. The following advanced settings can be modified on the Advanced tab.

For 3270E connections you can specify an LU name or LU Pool name for your connection. You have four options for specifying the LU name or LU Pool name. Select one of the following options:

**None (server assigned)**
Click this if you want the Telnet server to assign the LU. This option can be used with pooling.

**Prompt user**
Click this if you want to prompt the user for the LU name. If pooling is enabled, this option should not be used.

**Use a specified value**
Use this to specify the LU name or LU Pool name to be used. If you plan to use connection pooling, you may specify the name of an LU Pool large enough to support the connection pool you define.

**Use an HTTP session variable** `Web-only`
Click this if you want to specify the contents of the HTTP session variable as the LU name or LU Pool name. If pooling is enabled, this option should not be used.

For 5250 connections you can specify a Workstation ID for your connection. You have four options for specifying the workstation ID. Select one of the following options:

**None (server assigned)**
Click this if you want the Telnet server to assign the workstation ID. This option can be used with pooling.

**Prompt user**
Click this if you want to prompt the user for the workstation ID. If pooling is enabled, this option should not be used.

**Use a specified value**
Click this if you want to supply a string from which the workstation ID is created. This option can be used with pooling if a wildcard workstation ID is specified.

Workstation ID parameter for 5250 connections can use wildcard characters. The workstation ID defines the name of the workstation. The first character must be A-Z, $ (dollar sign), @ (commercial at sign), or # (number sign). The remaining characters can be A-Z, 0-9, $, @, #, . (period), and _ (underscore). If you do not complete this field, a workstation ID is automatically assigned by the host. The HATS Server can generate a new and non-arbitrary workstation ID for a session. Keywords and special characters in the workstation ID field can cause some or all of the following information to be substituted into the workstation ID value that is sent to the Telnet server:

- short Session ID(*)
- Session Type ID(%)
- Collision Avoidance ID(=)

When specified, the Collision Avoidance ID enables the generation of a new workstation ID if the Telnet server rejects the previous name (which can occur when the old name is already in use on the IBM i server).

Wildcard combinations in the workstation ID field allows the HATS Server to automatically generate more ID variations for achieving connection acceptance from the host. This decreases the time required for session connection, as it decreases the number of times the host must reissue a request to the client for valid workstation ID.

Wildcard characters can be specified multiple times in the workstation ID field in any combination with other alphanumeric characters. (For example: N=A=M=E, NAME==, and so on.) With each use of the wildcard, you decrease the likelihood of generating a workstation ID already claimed by another session:

- Use of two [=] results in approximately 1 in 50 chances of duplicating an established ID.
- Use of three [=] results in approximately 1 in 300 chances of duplicating an established ID.
- The number of unique names generated is approximately $36^n$, where *n* is the number of equal signs [=].

For example: Given the input of NA=ME for workstation ID, the one [=] can trigger generation of 36 unique IDs that use the alphanumeric characters specified (N, A, M, E). Adding a second [=] can trigger generation of approximately 1296 unique IDs, and so on.

**Use an HTTP session variable** `Web-only`
Click this if you want to specify the contents of the HTTP session variable as the workstation ID. If pooling is enabled, this option should not be

used. You can also use wildcard characters in the specified string. For more information, see “Use a specified value” on page 134.

In the **Configure optional, advanced connection settings** section you can add, modify, or remove any additional IBM Host On-Demand session parameters using the buttons to the right of the table of parameters. If you click **Add**, you can select a parameter using the drop-down list next to the **Name** field and then type a value into the value field. Some parameters are set on other tabs of the connection editor. If you try to set them here, a warning message is displayed. For more information, go to the Host On-Demand Knowledge Center at http://www-01.ibm.com/support/knowledgecenter/SSS9FA_11.0.0/com.ibm.hod.doc/WebSphereHOD.htm and search on `session parameters`.

**Notes:**

1. Parameters that have their own fields on the Basic and Advanced tabs cannot be entered in this table. They must be modified in their own specific fields.
2. Print support parameters added here will be ignored. Add your print support parameters on the Printing tab. For more information see “Printing.”
3. For 3270E connections, the **negotiateCResolution** parameter is set to **true** by default. See “Contention resolution using z/OS Communications Server” on page 482 for more information and for when you might want to change this setting to **false**.

Other settings on the Advanced tab allow you to specify:

- Only allow single connection with each user ID.
- Abandon attempt to connect after a specified number of seconds.
- Abandon attempt to disconnect after a specified number of seconds.

For DBCS considerations about displaying and printing user-defined characters (UDCs), see “Working with user-defined characters” on page 464.

## Printing

Use the **Printing** tab to specify print support settings for the connection. Print support is available only for 3270E, 5250, and 5250W default connections.

**Note:** Print support is not available and the Printing tab is not displayed in projects that are optimized for mobile devices.

Select the **Enable print support** check box if you want print support.

### Print settings for 3270E connections

For 3270E connections, the default print settings are for Adobe Portable Document Format (PDF). If you are using PDF, in the **Adobe PDF File Properties** section you can select the paper size, page orientation, and font you want to use for printing jobs. The list of fonts from which you can select depends on the code page setting for the connection you are using. See Chapter 22, “Language support,” on page 427 for more information about code page.

To specify other, non-default, print settings you can use the **Initialize** section and the **Name/Value** table.

Use the **Initialize** button to add a starter set of print settings into the **Name/Value** table based on the printing scenario you select from the drop-down box. Three printing scenarios are supplied as starter sets. The scenarios and the print settings they generate are:

- Default printing (Adobe PDF format)

| Name | Value |
|---|---|
| printDestination | false |
| printMimeType | application/pdf |
| printSaveAsExtension | .pdf |
| separateFiles | true |
| useAdobePDF | true |
| usePDT | false |
| useWindowsPrinter | false |

- Basic text files (Plain text format)

| Name | Value |
|---|---|
| PDTFile | /pdfpdt/basic.hodpdt |
| printDestination | false |
| printMimeType | text/plain |
| printSaveAsExtension | .txt |
| separateFiles | true |
| useAdobePDF | false |
| usePDT | true |
| useWindowsPrinter | false |

- Direct to server-attached default Windows printer **Web-only**
  or
  Direct to default Windows printer **RCP-only**

| Name | Value |
|---|---|
| printDestination | true |
| useAdobePDF | false |
| usePDT | false |
| useWindowsDefautlPrinter | true |
| useWindowsPrinter | true |

Using the **Initialize** button is optional. If values exist in the table when the button is clicked, a warning message that the current settings will be replaced is issued. You can click to continue or cancel.

You can enter your own print settings, or modify settings already in the table using the **Add**, **Edit**, and **Remove** buttons. Clicking the **Add** button will present a combo box listing all of the supported settings from the list of WebSphere Host On-Demand printer settings.

For example to have more control over 3270E PDF print output you can add the following settings:

**charsPerInch**
Characters per inch. Specifies the number of characters printed per inch. On a Windows platform, three choices (10, 12, and 17) are available. The default value is 10.

**linesPerInch**
Lines per inch. Specifies the number of lines per inch. On a Windows platform, five choices (2, 3, 4, 6, and 8) are available. The default value is 6.

**maxCharsPerLine**
Maximum characters per line. Specifies the maximum number of characters per line, also called the Maximum Print Position or the Maximum Presentation Position (MPP). Enter a value from 1 to 255. The default value is 80.

**maxLinesPerPage**
Maximum lines per page. Specifies the maximum number of lines per page, including the top and bottom margins. This value is also called Maximum Page Length (MPL). Enter a value from 1 to 255. The default value is 66.

**RTLfile**
Printing right-to-left files for bidirectional language applications. Specify "true" to print a file as it appears on a RTL screen. The default is "false."

**Notes:**

1. Printing to a default Windows printer attached to a user's workstation is supported only by HATS rich client applications.
2. Printing to a default Windows printer attached to the WebSphere Application Server is supported only by HATS Web applications.
3. HATS 3270E printing does not support the Transparency command for sending unprocessed print data in the print job.
4. For more information about Japanese JIS2004 support, see "JIS2004 support for PDT printing and Print-to-File for 3270E sessions" on page 433.

See "Defining print support for your project" on page 353 for more information.

### Print settings for 5250 and 5250W connections

For 5250 and 5250W connections, if you select the **Enable print support** check box, you must specify the Web address of the System i® Access for Web Printer Output window. The default Web address is http://*hostname*/webaccess/iWASpool, where *hostname* is the name of the 5250 or 5250W host. The user of your application can set print options in the IWA Printer Output window. See "Defining print support for your project" on page 353 for more information.

## Screen Handling

The **Screen Handling** tab allows you to determine how a blank screen is handled at connection startup and how screen settling is handled. These settings only apply if your connection is used as the default connection.

When the connect application event processes the obtain default connection action, HATS attempts to open a connection to the host. If this connection attempt is not completed successfully, the application will end with an error page. If the connection attempt does complete successfully, the host screen is allowed to settle according to the screen timers specified on this page. These timers are also used to settle the host screen at the completion of a custom screen recognition event, the blank screen application event (if configured), and the unmatched screen event. For

a detailed description of screen-settling algorithms and settings, refer to Appendix B, “HATS screen-settling reference,” on page 477.

The **If blank screen is received at connection startup** settings allow you to select what to do if the host screen remains blank after successfully connecting to the host, and allowing the host time to settle the screen as explained above. Your options are:

**Wait until connect timeout before termination with an error**
This option causes HATS to terminate the application with an error page if the host screen remains blank after successfully connecting to the host.

**Show blank screen**
This option causes HATS to proceed with screen recognition processing even if the host screen remains blank after successfully connecting to the host.

If you wish to ensure that a blank screen is not shown to your users, you may specify a screen recognition event in the Event Priority list for processing a blank screen, or you may add actions to the Blank screen Application Event to handle a blank screen. By default, the Unmatched screen application event will present the user with the blank host screen. For more information about screen recognition events, refer to “Screen event priority” on page 101. For more information about application events, refer to “Application events” on page 102.

**Send the host key**
This option will send a host function key to the host if the connection remains blank after successfully connecting to the host. This key will be sent once, the screen will be allowed to settle, and then HATS will proceed with screen recognition processing. You can select which key to send from the drop-down list. This option is useful if your host always draws a blank initial screen, and users must submit a particular function key (SysReq, for example) before the host will present a new screen.

**Screen timers** allow you to configure the length of time to wait for the host to complete sending its screens to the HATS runtime. To understand when these timers are used, refer to Appendix B, “HATS screen-settling reference,” on page 477. You can specify the following settings:

**Minimum time to wait for initial host screen:**
The default is 2000 milliseconds. This is the minimum amount of time that the application waits for the arrival of initial screen updates after the host connection becomes ready. Increase this amount if the host is slow to send the first screen, even if the connection has been in the ready state for some time.

**Maximum time to wait for the screen to settle:**
The default is 1200 milliseconds. This is the maximum amount that the application waits for the arrival of screen updates after the initial screen update. Increase this value if the host is slow to send the contents and you receive frequent partial screens.

**Maximum time to wait for screen to settle (session using asynchronous update):**
This value is only used if your application is using asynchronous update (for rich client applications) or the asynchronous update applet (for Web applications). For more information see “Asynchronous Update RCP-only ” on page 109 and “Using the server push (applet) method” on page 111.

The initial default value is 400 milliseconds. You should increase this amount if the host is slow to send the screen and you receive frequent partial screens. This value can also be lower than the **Maximum time to wait for the screen to settle** if the browser specific settings in the com.ibm.common.AppletSettings class in the application.hap file are set to **refresh**.

## Security

The **Security** tab contains configuration settings for Secure Socket Layer (SSL) and Web Express Logon (WEL). For more information about security settings, see Chapter 21, "Security and Web Express Logon," on page 403.

**Note:** To enable SSL for a connection in a HATS EJB project, in the HATS EJB Project view open the EJB project and the Connections folder and double-click on the connection. Then follow the directions below. For information about how to create a HATS EJB project see the section, Creating and using a HATS EJB application, in the *HATS Web Application Programmer's Guide*.

**Enable SSL**
Select this check box to enable SSL.

**Note:** If the Telnet server uses a valid well-known personal certificate, then selecting this box is all that is required.

**Import PKCS12 keystore into project**
Select this option to import a PKCS12 keystore file into your project. Click the **Import** button to browse for and import the keystore file into the project. A pointer to the imported keystore file is set in the configuration for this connection. Click the **Remove** button to remove the pointer to the keystore file from the connection configuration. The keystore file, itself, is not removed from the project. After importing the file, the file name appears in the **Path to keystore file** edit box. For more information about when this is necessary and how to create a PKCS12 keystore file see "Enabling SSL security" on page 403.

**Note:** After importing a keystore file and saving the changes in the connection editor, refresh your project to ensure the keystore file is included in the project. To refresh your project, from the HATS Projects view right-click the project and select **Refresh**. To set your project to automatically refresh, from the Rational SDP menu bar select **Window > Preferences > General > Workspace > Refresh automatically**.

**Use PKCS12 keystore at a specific path**
Select this option to specify a keystore file that will not be contained within your project but will exist elsewhere on the target runtime system. In the **Path to keystore file** edit box, specify the complete path and file name for the keystore file on the target system. For more information about when this is necessary and how to create a PKCS12 keystore file see "Enabling SSL security" on page 403.

**Notes:**

1. To use this file during testing on your development system, it must reside at the same location on your development system as it does on the target runtime system.
2. For HATS Web applications, if you use a keystore file that is not contained within your project .ear file, and Java 2 security is enabled at the target WebSphere Application Server system, you must update the was.policy file on WebSphere Application Server before your HATS application tries to access it. The was.policy file is located in the Navigator view of the project .ear file in the META-INF directory. For example, to give read permissions for the keystore file, add the following statement to your was.policy file.

   ```
   permission java.io.FilePermission "c:\\myKeystores\\-", "read";
   ```

   Where `myKeystores` is the name of the folder containing the keystore file on the target WebSphere Application Server system. For more information see "Java 2 security" on page 419.

**Path to keystore file**

If you have imported a keystore file, this edit box contains the file name of the imported file. If you have selected the option to **Use PKCS12 keystore at a specific path**, then enter in this edit box the complete path and file name for the keystore file on the target runtime system.

**Password**

The password required to open the keystore file specified in the **Path to keystore file** edit box. Use the **Verify** button to test finding the keystore file and opening it with the password.

**Notes:**

1. This is the same password that was used when the keystore file was created. For more information about how to create a PKCS12 keystore file see "Enabling SSL security" on page 403.
2. To verify the location and password for a keystore file that is not contained within the project, the keystore file must reside at the same location on your development system as it does on target runtime system.
3. The password is not stored in the clear. However, if after deploying your HATS application, you want to change the password without having to redeploy the application, you can modify the password field in the .hco file that represents the connection on the runtime system. After editing the .hco file and making the modification, the password is stored in the clear until you redeploy the application.

**Enable JSSE**

Select this check box to enable JSSE.

**Use JSSE**

Selecting the 'Use JSSE' check-box enables the use of TLS v1.0, TLS v1.1, or TLS v1.2 using the Java Secure Socket Extension (JSSE) security library, instead of SSLite, for the connection between the HATS and the HOST system. The default option of using the

SSLite library, can be overridden by selecting this radio button to use TLS v1.1 or TLS v1.2 for a connection.

**Import Java keystore into project**
Select this option to import a **jks** keystore file into your project. Click the **Import** button to browse for and import the keystore file into the project. A pointer to the imported keystore file is set in the configuration for this connection. Click the **Remove** button to remove the pointer to the keystore file from the connection configuration. The keystore file, itself, is not removed from the project. After importing the file, the file name appears in the **Path to keystore file** edit box.

**Use jks keystore at a specific path**
Select this option to specify a keystore file that will not be contained within your project but will exist elsewhere on the target runtime system. In the **Path to keystore file** edit box, specify the complete path and file name for the keystore file on the target system.

**Path to keystore file**
If you have imported a keystore file, this edit box contains the file name of the imported file. If you have selected the option to **Use jks keystore at a specific path**, then enter the complete path and file name for the keystore file on the target runtime system, in this edit box .

**Password**
The password required to open the keystore file specified in the **Path to keystore file** edit box. Use the **Verify** button to test if the keystore file can be found and opened using the password.

**Add MSIE browser's keyring**
This check-box can be used only when JSSE is enabled.

Enable this checkbox to support **MSCAPI/Microsoft Cryptography API** for HATS. When this option is selected, the HATS client accepts certificate authorities trusted by the Microsoft Internet Explorer browser.

When this option is enabled, the 'SSLBrowserKeyringAdded' parameter will be set to true in the 'Advanced' tab of connection file.

**MSCAPI can be used only for HATS toolkit and in HATS RCP application**.

**Notes:**

1. MSCAPI is not supported for SSL.
2. As MSCAPI is supported only for toolkit and RCP application, users must add the jks file for HATS web based application. Otherwise, when deployed in runtime, the connection to host fails while validating the certificate.

**Use Web Express Logon** `Web-only`
For HATS Web applications, select this box and click the **Configure** button to enable and configure WEL. For more information see "Using Web Express Logon (WEL)" on page 406.

**Use Kerberos services ticket to automate sign-on (Windows domain clients only)** `RCP-only`

For HATS rich client applications, select this box to use a Kerberos services ticket to automate sign-on. This support uses underlying IBM Host On-Demand connection-based automation in conjunction with IBM i Kerberos-based network authentication. For more information, see the section on configuring connection-based automation in the Host On-Demand Knowledge Center at: http://publib.boulder.ibm.com/infocenter/hodhelp/v11r0/index.jsp?topic=/com.ibm.hod.doc/doc/logon/logon14.html.

**Notes:**

1. This setting is only available for 5250 Telnet connections.
2. This function is only supported when running on Windows domain clients.

## Pooling

A pool is a group of host connections that are maintained in an initialized state, ready to be used without having to create and initialize them. This reduces the response time when starting and running an application.

You can enable pooling by clicking the **Pooling** tab and selecting the **Enable pooling** check box. If pooling is enabled, after the connection is released and determined to be in an acceptable state as defined by the checkin screen definition, it is kept in a list of active connections. When a new request for a connection is received, one of these idle pooled connections is used. If there are no idle pooled connections, a new one may be created based on other pool settings such as maximum connections. If connection pooling is disabled, after a connection has been released, it will be disconnected.

Because there is no guarantee that a user navigating screens on the default connection will navigate back to the checkin screen, it is recommended that you do not use pooling on the HATS default connection. Pooling is efficient and effective for background connections used by Integration Objects or used in perform macro transaction actions, since these automated navigations can be programmed to navigate back to the checkin screen.

You also have the option of setting your **Connection Timeouts** and **Connection Limits**.

- For **Connection Timeouts**, select the **Terminate connection after a maximum idle time** check box, the **Terminate connection after a maximum busy time** check box, or both and enter the time in seconds.

  **Notes:**

  1. **Terminate connection after a maximum busy time** can also be used outside of pooling.
  2. Maximum busy time should be set to the same value for all connections within a HATS application.
- For **Connection Limits**, enter the minimum number of idle connections to retain in the pool, the maximum number of connections in the pool that can be active, or both in the text boxes.

  You can also decide what should be done after the maximum number of connections has been reached. You can either create a new (non-pooled) connection by selecting the radio button or wait for an available pooled connection and set the limit to the time to wait by selecting the **Limit the time**

**to wait for a pooled connection to a maximum number of seconds** check box and entering the time in seconds in the text box. Clear the check box to wait indefinitely for a connection.

## Macros

The **Macros** tab shows information about the Connect macro, the Disconnect macro, and the checkin screen. Both macros are optional. This information is on a separate tab from the pooling information because these macros can still be applied even if pooling is disabled.

Select the connect and disconnect macros from the **Connect macro** and **Disconnect macro** drop-down lists.

A **Connect macro** is run automatically when the connection is initially created. The only prompts allowed in a connect macro are those that can be satisfied by a user list or by using Web Express Logon. If pooling is enabled, the connect macro will be run when the HATS server initializes, otherwise the connect macro will be run when a user makes the first request to the HATS application (typically in the Connect event).

**Note:** If you need to have an automatic sign-on macro run using the user's credentials, you can add a play macro action to the connect event instead of using a connect macro here. Macros run using the play macro action can satisfy prompts interactively or with global variables initialized during the start event, for example.

A **Disconnect macro** is run automatically when the connection is destroyed. No prompts are allowed in a disconnect macro. If pooling is enabled, the disconnect macro will be run when the HATS server is shut down, otherwise the disconnect macro will be run when the HATS application releases the connection back to the system (typically in the Disconnect event).

A **Checkin screen** describes the screen a connection must be on to be placed into the pool. It is only enabled if pooling is enabled. Typically, the checkin screen is the same as the exit screen of the connect macro, and this also matches the entry screen of the disconnect macro.

The checkin screen section will be available only if you have enabled pooling on the **Pooling** tab. If you have enabled pooling, the checkin screen section has three options:

- **Use first exit screen of Connect macro**. This is the recommended option.
- **Use a specific screen of Connect macro** where you have the option of selecting a screen macro from the drop-down list.
- **Use different screen recognition criteria based on a previously captured screen** allows you to use different screen recognition criteria for the checkin screen based on a previously captured screen which you select from the drop-down list.

You can also specify recognition criteria for the checkin screen. For instructions on how to define screen recognition criteria, see “Screen Recognition Criteria or Begin Screen” on page 148.

**Note:** A poorly-defined checkin screen will render your pool useless. Adding definitive recognition criteria is highly recommended.

For more information about macros, see Chapter 11, “Macros and host terminal,” on page 323. If you use a connect macro, you can set up a user list to specify the list of users who can use the connection.

## User List

The user list is a list of user profiles that each include a user ID and password. The user list can be used by a connect macro associated with the host connection. When the connect macro runs, a user ID and password is pulled from the list and placed into the user ID and password fields in the host screen. This allows a predefined list of user IDs and passwords to be used, and the user does not have to know or enter a user ID and password on the host screen.

User lists are useful for cases where many end users may concurrently access a HATS Web application, EJB, or Web service that requires an automated login using a centralized list of generic user IDs.

Because HATS rich client applications and their user lists are deployed to individual end user systems rather than to a central server, user lists are typically less useful since each deployment will attempt to use the same user ID at the same time, possibly resulting in a failure to log in. Rich client applications might use a user list successfully for those host systems where a given user ID can be used by many users concurrently. In this case, a user list might be provided with only a single generic user ID which will be used concurrently by all end user systems. You might choose to do this if you want to supply a generic user ID and password, and want the password to be encrypted.

Be sure to properly set the option, **Only allow single connection with each user ID**, on the Advanced tab of the Connection editor to reflect the type of host system associated with the user list. If this option is not selected, the system will reuse one user profile for all connections to the host.

The **User List** tab displays a table of user profiles with the capability to add, edit, or remove any one of them.

If you click **Add** , you are prompted to provide the user ID, a description, and password. All of these fields are required and any of these fields can be modified by clicking **Edit**. You can also delete the entry by selecting it and clicking **Remove**.

Select **Encrypt user list properties** to store the passwords for all of the user list entries in encrypted form. The passwords are displayed as encrypted when the Source tab is selected. Clear this box to store all of the passwords in unencrypted form. The passwords always display as asterisks (*), and the User ID and Description fields remain unencrypted, regardless of this setting.

**Note:** For information about encrypting Host Publisher user lists, see “Migrating from Host Publisher Version 4” on page 23.

To specify the use of profiles from the user list, as you record a connect macro, when you reach the user ID field, click the **Add Prompt Action** icon on the host terminal toolbar. On the Add Prompt Action page, select **Set prompt to property from User List**. For User Profile, select a profile from the dropdown, and for User List property, select **_userid**. For the password field, use the same process, select the same profile, and for User List property, select **_password**. The profile you select while recording the macro is the one used as you test the macro. All profiles in the list are used, as necessary, during runtime.

To associate a connect macro with a host connection, edit the connection. On the Macros tab, for Connect macro, select your macro from the dropdown.

When using user lists, you must also record a disconnect macro to log off the user ID so it can be reused. To associate a disconnect macro with a host connection, edit the connection. On the Macros tab, for Disconnect macro, select your disconnect macro from the dropdown.

### Clustering and user lists

When you create multiple instances of WebSphere application servers running HATS applications and sharing the same application files, you can use user lists on multiple-logon hosts without any special considerations. However, there are special considerations for the use of user lists with single-logon hosts:

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

## Source

The **Source** tab displays the tags and values in the .hco file for many of the settings you selected, or for which you took the defaults, in the connection editor. As you make changes on other tabs in the connection editor, the tags and values displayed under the **Source** tab change to match.

You can also make changes to the tags and values in the .hco file directly by editing the source under the **Source** tab, and they will be reflected on the appropriate tabs of the connection editor.

# Chapter 7. Working with screen events

A HATS event is a resource that performs a set of actions based on a certain state being reached. There are two types of HATS events, application events and screen events. For more information about application events see “Application events” on page 102. A screen event is a HATS event that is triggered when a host screen is recognized by matching specific screen recognition criteria. There are two types of screen events, screen customizations and screen combinations.

A screen customization is a HATS screen event designed to perform a set of actions when a host screen is recognized. Examples of screen customizations include recognizing a screen and transforming it into a GUI for the user or playing a macro to skip the screen. The screen customization definition includes a set of screen recognition criteria and a list of actions to be taken when a host screen matches the screen recognition criteria. Screen-level global rules and text replacement settings are also included. Use the Create a Screen Customization wizard to create a new screen customization.

A screen combination is a HATS screen event designed to gather output data from consecutive, similar host screens, combine it, and display it in a single output page. An example of a screen combination includes recognizing a screen that contains only partial output data and navigating through all subsequent screens to gather all of the remaining data to display for the user. The screen combination definition includes a set of screen recognition criteria for both the beginning and ending screens to be combined, how to navigate from screen to screen, and the component and widget to use to recognize and render the data gathered from each screen. Also, like the screen customization, it includes a list of actions to be taken, screen-level global rules, and text replacement settings. Use the Create a Screen Combination wizard to create a new screen combination.

## Create a Screen Customization wizard

In your HATS project, use the **Create a Screen Customization** wizard to define screen customizations. You must have the host terminal open or a screen capture before you use the screen customization wizard. BMS maps can also be used to create screen captures used for screen customizations. For more information about BMS map sets, see “Importing BMS map sets” on page 170.

You can do any of the following actions to access the wizard:

- In the **HATS Projects** view, right click on the **Screen Customizations** folder in your project, and select **New HATS > Screen Customization**.
- On the HATS toolbar click the **Create HATS Screen Customization** icon.
- On the host terminal tool bar click the **Create HATS Screen Customization** icon.
- In the **HATS Projects** view, open the **Screen Captures** folder in your project. Double click on a captured screen to open it in the editor. After the editor opens, click the **Create HATS Screen Customization** icon.

For details on settings available in the wizard see “Editing screen events” on page 148.

The settings you define for the screen customization are saved in a screen customization (.evnt) file. You can see this file by expanding the **Screen**

**Customizations** folder in the **HATS Projects** view. Use the screen event editor to view and modify the screen customization file.

## Create a Screen Combination wizard

In your HATS project, use the **Create a Screen Combination** wizard to define screen combinations. You must have the host terminal open or a screen capture before you use the screen combination wizard. BMS maps can also be used to create screen captures used for screen combinations. For more information about BMS map sets, see "Importing BMS map sets" on page 170.

You can do any of the following actions to access the wizard:

- In the **HATS Projects** view, right click on the **Screen Combinations** folder in your project, and select **New HATS > Screen Combination**.
- On the HATS toolbar click the **Create HATS Screen Combination** icon.
- On the host terminal tool bar click the **Create HATS Screen Combination** icon.
- In the **HATS Projects** view, expand the **Screen Captures** folder in your project. Double click on a captured screen to open it in the editor. After the editor opens, click the **Create HATS Screen Combination** icon.

For details on settings available in the wizard see "Editing screen events."

The settings you define for the screen combination are saved in a screen combination (.evnt) file. You can see this file by expanding the **Screen Combinations** folder in the **HATS Projects** view. Use the screen event editor to view and modify the screen combination file.

**Note:** HATS Dojo widgets are not supported in screen combinations.

## Editing screen events

You can invoke the screen event editor by double clicking on the name of either a screen customization or screen combination .evnt file. The following sections describe each tab of the screen event editor.

### Overview

The **Overview** tab of the screen event editor summarizes all of the information you specified when you created the screen event. It contains the name and description of the screen event, the name and an image of the screen that was used to create the screen recognition criteria, a summary of the actions to be taken, and a summary of the screen recognition criteria. For screen combinations, this tab also contains a summary of the navigation, component, and end screen recognition criteria settings. On this tab, you can modify the description of the screen event, and you can select a different screen to associate with the screen event. The selected screen is used whenever you make changes to the screen event, such as modifying screen recognition criteria, or adding actions.

### Screen Recognition Criteria or Begin Screen

The **Screen Recognition Criteria** tab displays when editing a screen customization event. The **Begin Screen** tab displays when editing a screen combination event. In both cases the tabs display the screen recognition criteria that you set to match host screens that will trigger the screen event. Host screens can be recognized by any combination of criteria including how many total fields or input fields are on

the screen, the coordinates of the cursor's position, and text strings on the screen within a defined rectangle or anywhere on the screen. You can add, edit, or remove criteria on this tab.

**Note:**

For considerations when using DBCS support see "Screen recognition criteria / Begin screen" on page 462.

## Associated screen capture

This option displays only on the **Begin Screen** tab for a screen combination event. Click the **Change** button to select a different screen capture to associate with the beginning screen of the screen combination.

## Fields criteria

You can use the **Total number of fields** on a screen, the **Number of input fields** on a screen, or both as screen recognition criteria. The **Total number of fields** setting includes input fields, protected text fields, and hidden fields. These are the first two criteria shown on the tab.

If you select the check box for these criteria, they are used to recognize the screen. The fields next to each field criterion show the total number of fields and input fields for the screen specified on the **Overview** tab. If you change the screen being used for this screen event, click **Refresh** to update the values for the screen you select.

**Note:** If you are using field criteria to recognize screens that have a certain number of fields, and another screen does not contain the same number of fields, that screen is not recognized. For example, one screen might have a list of ten files with ten fields. If the host displays a screen with only eight files in the list and eight fields, the second screen does not match the number of fields criterion of the screen event that matched the first screen.

For an explanation of the **Optional** and **Invert** check boxes, see "Optional versus non-optional screen recognition criteria" and "Inverted match of screen recognition criteria" on page 150.

## Cursor position criteria

To use the initial position of the cursor as a screen recognition criterion, either by itself or in conjunction with other criteria, select the **Cursor position** check box. The fields next to the cursor position criterion show the row and column of the cursor position for the screen specified on the **Overview** tab. If you change the screen being used for this screen event, click **Refresh** to update the values for the initial cursor position row and column on the screen you select.

For an explanation of the **Optional** and **Invert** check boxes, see "Optional versus non-optional screen recognition criteria" and "Inverted match of screen recognition criteria" on page 150.

## Optional versus non-optional screen recognition criteria

You can select whether the screen recognition criteria you set is optional or non-optional. If you do not select the **Optional** check box, the recognition criterion is considered non-optional. How you use the **Optional** check box corresponds to the Host On-Demand screen descriptor attribute, **optional**.

If you have both optional recognition criteria and non-optional recognition criteria, HATS checks the non-optional criteria first. If all the non-optional criteria match, the screen matches. If at least one of the non-optional criteria does not match, HATS checks the optional criteria. For a screen to match the criteria, HATS must find all non-optional criteria, or at least one optional criterion. Otherwise, the screen fails to match. The following example explains this concept in greater detail.

**Note:** Non-optional does not mean required.

Suppose you defined cursor position location and two text strings with the values shown in the following example:

```
Cursor position recognition  Optional
                             Row: 1   Column: 1

String recognition           Non-optional
                             String 1: Welcome
                             Start position: Row: 1   Column: 6
                             End position:   Row: 1   Column: 12

                             String 2: Username
                             Start position: Row 20   Column 10
                             End position:   Row 20   Column 17
```

In this example, HATS must find both text strings or the cursor position for the screen to match. Because HATS checks non-optional criteria first, HATS looks for the text strings first. If HATS cannot find both text strings in the specified regions of the host screen, then it checks to see whether the optional criterion (cursor position) can be found.

## Inverted match of screen recognition criteria

You can select whether the screen recognition criteria you set matches or does not match the host screen. If you select the **Invert** check box, the recognition criterion must not match the screen for the criterion to be considered true.

Conversely, if you clear the **Invert** check box, the recognition criterion must match the screen for the criterion to be considered true.

How you use the **Invert** check box corresponds to the screen descriptor attribute, **invertmatch**.

## Additional criteria

**String criterion:** Text string location criteria are shown in the Additional Criteria section of the tab. If you have set a string location as a screen recognition criterion, it is shown in the table. The table shows the type of screen location containing the text, some of the characters of the text selected, and whether the text is case-sensitive, optional or invert.

If you highlight a row of the table that defines a string criterion and click **Edit**, or if you click **Add**, the Screen Criterion dialog appears. In the dialog panel, you can either modify or specify text string information. You also can select attributes for the string (case sensitive, optional or invert). The panel shows the screen selected on the **Overview** tab.

You can select any text on the screen by drawing a rectangle around the text. Place your cursor at any point on the screen, click and hold the left mouse button, and move the cursor to another location on the screen to draw the rectangle. The fields on the right of the dialog show the text you selected and the starting and ending

row and column numbers of the rectangle. You can specify the part of the screen that should contain the text by clicking one of the radio buttons for **Anywhere on the screen**, **At a specified position**, or **Within a rectangular region**. If the text you selected must be case-sensitive to be recognized as matching the screen recognition criteria, select the **Case sensitive** check box.

For an explanation of the **Optional** and **Invert** check boxes, see "Optional versus non-optional screen recognition criteria" on page 149 and "Inverted match of screen recognition criteria" on page 150.

Click **OK** when you have finished your selections.

**Global variable criterion:** You can use global variables to define your screen recognition criteria by clicking the **down arrow** beside the **Add** button and selecting **Global Variable Criterion**. If you use global variable criterion, you must either specify a global variable from the **Global Variable** drop-down list or type in the value. Specify the **Verification logic** by selecting one of the five bullets:

- **This global variable exists**
- **This global variable does not exist**
- **Check the length of this global variable** where you can specify how the length will be compared from the drop-down list, as well as the length to compare.
- **Check the integer value of this global variable** where you can specify how the integer value will be compared from the drop-down list, as well as the integer value to compare.
- **Check the string value of this global variable** where you can specify how the string value will be compared from the drop-down list, as well as the string value to compare.

For an explanation of the **Optional** and **Invert** check boxes, in the **Attributes** section, see "Optional versus non-optional screen recognition criteria" on page 149 and "Inverted match of screen recognition criteria" on page 150.

For more information about advanced global variable programming, see the *HATS Web Application Programmer's Guide* or the *HATS Rich Client Platform Programmer's Guide*, depending on your application environment.

**Color criterion:** You can define your screen recognition criteria by color by clicking the down arrow beside the **Add** button and selecting **Color Criterion**. This will open a panel in which you can select the row and column position as well as the foreground and background color.

For an explanation of the **Optional** and **Invert** check boxes, see "Optional versus non-optional screen recognition criteria" on page 149 and "Inverted match of screen recognition criteria" on page 150.

**Compare region to value criterion:** The **Compare region to value criterion** allows the comparison of text from a selected region of the host screen and to a hard coded value.

Start by drawing a box around the region of the terminal or type in the coordinates for the recognition in the **Define Region** section. The **Current value on the terminal** will also be shown in the text box.

Next, you can specify what the region will be compared to and how it will be compared. In the **Define Comparison** section, select how your region will be

compared by selecting from the drop-down list next to **Region**, and enter what it will be compared to in the **Value** text box.

**Comparison Type** will either be **Numeric** or **Text** (with the a **Case sensitive** check box option).

Attributes can also be defined. For an explanation of the **Optional** and **Invert** check boxes, see "Optional versus non-optional screen recognition criteria" on page 149 and "Inverted match of screen recognition criteria" on page 150.

Click **OK** when you have finished your selections.

**Compare region to region criterion:** The **Compare region to region criterion** allows the comparison of one region on the host screen to another region of the same host screen.

First select the **First Region** tab and start by drawing a box around the region of the terminal or type in the coordinates for the recognition in the **Define Region** section. The **Current value on the terminal** will also be shown in the text box.

Next, you can specify how these regions will be compared in the **Define Comparison** section by selecting from the drop-down menu between **Region 1** and **Region 2**.

**Comparison Type** will either be **Numeric** or **Text** (with the a **Case sensitive** check box option).

Attributes can also be defined. For an explanation of the **Optional** and **Invert** check boxes, see "Optional versus non-optional screen recognition criteria" on page 149 and "Inverted match of screen recognition criteria" on page 150.

Repeat the same steps for the **Second Region** tab then click **OK** when you have finished.

**Condition criterion:** The **Condition criterion** allows you to specify a conditional expression that the macro runtime evaluates during screen recognition, such as `$intNumVisits$ == 0`.

During screen recognition the macro runtime evaluates the conditional expression and obtains a boolean result.

If the conditional expression evaluates to true then the macro runtime evaluates this descriptor as true. Otherwise the macro runtime evaluates this descriptor as false.

The **Condition criterion** increases the flexibility and power of screen recognition by allowing the macro runtime to determine the next macro screen to be processed based on the value of one or more variables or on the result of a call to a Java method.

In the **Condition** input field, enter a conditional expression that the macro runtime evaluates during screen recognition.

For an explanation of the **Optional** and **Invert** check boxes, see "Optional versus non-optional screen recognition criteria" on page 149 and "Inverted match of screen recognition criteria" on page 150.

**Custom criterion:** The **Custom criterion** allows you to call custom description code.

In the **Class name** input field, specify the name of the Java class containing the method you want to run, or click **Browse** to select the class name from the **Source** directory.

For an explanation of the **Optional** and **Invert** check boxes, see "Optional versus non-optional screen recognition criteria" on page 149 and "Inverted match of screen recognition criteria" on page 150.

## Rendering (screen combination only)

The **Rendering** tab is displayed only for a screen combination event. Use this tab to set which component to use to recognize the information to be gathered by the screen combination. Also, use this tab to set which widget to use to render the component. A subset of the HATS components and widgets are supported in screen combination events. See "Insert Host Component" on page 178 for more information about using HATS components and widgets.

You can click the **Change Region** button to change the screen capture and the region of the screen to be considered by the component.

You can also use the following setting in non-portlet projects to control when the results of the screen combination are first displayed to the user in relation to when the screen combination completes.

**use dynamic, cached content loading** `Web-only`
: Leave this box clear to specify that HATS should navigate through all of the screens of the screen combination and combine all of the results before displaying the results to the user. Select this box to specify that HATS should begin displaying the results to the user as soon as the results from the first screen are gathered and continue updating the display of the results as the screens are navigated.

## Navigation (screen combination only)

The **Navigation** tab is only displayed for a screen combination event. This tab displays the commands necessary to navigate from the begin screen, through intermediate screens, to the end screen of the screen combination. Use the **Add**, **Edit**, and **Remove** buttons to set commands for **Up Movement** and **Down Movement** navigation directions. In the **Add command** or **Edit command** dialog that appears, there are three command options. Multiple commands per navigation direction can be set by clicking the **Add** button multiple times. Below are the three command options that can be set.

**Send key**
: Use this option to select a host key to send to the host system.

**Insert text**
: Use this option to insert text at the current cursor location.

**Set cursor position**
: Use this option to set the current cursor position on the host screen.

## End Screen (screen combination only)

The **End Screen** tab is only displayed for a screen combination event. This tab displays the criteria you set that defines the end of the screen combination. You

can add, edit, or remove criteria on this tab. You can select one of three types of end criteria, **Iteration count only**, **Screen descriptor only**, or **First match of either the iteration count or the screen descriptor**.

### Select ending criteria

**Iteration count only**
Select this, along with the **Number of iterations** setting, to specify that this screen combination ends after a fixed number of screens.

**Screen descriptor only**
Select this, along with the other screen recognition settings, for example the **Total number of fields**, the **Number of input fields**, the **Cursor position**, or the **Additional Criteria**, to define the end screen for the screen combination.

**First match of either iteration count or the screen descriptor**
Select this, along with all of the other settings, to specify that the screen combination ends either after a fixed number of screens or when the end screen is recognized, whichever occurs first.

### Number of iterations

Select this box to set the number of times to perform the navigation defined on the navigation tab. When the number is reached the end screen is assumed to be reached.

### Associated screen capture

Click the **Change** button to select a different screen capture to associate with the ending screen of the screen combination.

### Screen descriptor settings

The screen descriptor settings for the end screen are the same as the recognition criteria settings for the begin screen. See “Screen Recognition Criteria or Begin Screen” on page 148 for an explanation of these settings.

## Actions

The **Actions** tab of the screen event editor displays the configured actions for the screen event.

If you want to change the order of the actions, select one and click **Up** or **Down** to move that action higher or lower in the list.

You can also **Edit** or **Remove** any actions which appear in the list as well as **Add** a new action.

These actions are performed by the HATS runtime in the order that they are listed. If you want to display the transformed host screen to the user with some prefilled fields, for example, be sure to modify the host screen (using an **Insert data** action, for example) before you display the transformed host screen (using an **Apply transformation** action). If you use a **Play macro** action or a **Send key** action in a screen event, that action must be the last action in the list.

When all the actions in the screen event have been performed, HATS may send a default command to the transformation host in order to drive the host application off of the currently-recognized screen. Otherwise, an infinite loop might occur through the actions in the screen event. Selecting **Send a host key when the list completes to cause screen to change** lets you determine which key to send after

completion of the action list. By default, the HATS runtime will send an [enter] to the transformation host at the end of the action list. An aid key is sent if:

- No command parameter is received by the servlet from an **Apply transformation** action and
- No macro has been run against the transformation host using a **Play macro** action and
- No **Send key** action has sent the transformation host a screen-changing aid key.

So, if the action list contains an Apply transformation action, and the user sends the F2 command in response to the transformation, the [pf2] command will be sent rather than any default command. Likewise, if the screen event ends with a **Play macro** action, no default command will be sent.

You can add, edit, or remove actions on this tab. These are the possible actions:

- Apply transformation
- Execute business logic
- Extract global variable
- Insert data
- Set global variable
- Remove global variable
- Send global variable (HATS standard portlet projects only)
- Show URL or SWT Composite `RCP-only`
- Show URL `Web-only`
- Forward to URL `Web-only`
- Play macro
- Perform macro transaction
- Send key
- Disconnect
- Pause

All the action types you have defined for the screen event and their descriptions are shown in the table on the **Actions** tab. If you highlight a row of the table and click **Edit**, the **Edit Action** dialog appears.

If you click **Add**, the Select an Action wizard appears. The first panel shows a list of all the actions available. You can select the action you want to occur by clicking the radio button of that action and click **Next**. Depending on the action you select, the rest of the panel displays information that you can specify for that action.

**Note:** You cannot change the action type in the **Edit Action** dialog. You can change only the information that applies to the action.

## Apply transformation action

If you decide to apply a transformation as an action of this screen event, for HATS Web projects you can select the transformation you want to apply from the drop-down list of transformations defined in the project.

For HATS rich client projects you can use the **Browse** button to display a list of screen transformations in the class path of the current project. This typically just includes transformations in the current project, but it can also include other transformations from other plug-ins referred to by the current project or transformations contained within .jar files referred to by the current project.

For more information, see Chapter 8, "Working with transformations," on page 173.

The **Template** field has the **(default template)** selected by default. Unless you select a different template to be applied with this particular transformation, the template that surrounds the transformation in the GUI is the template you specified as the default template for the project. The drop-down list contains all the templates defined in the project.

Click **Immediate host keys** if you want some host keys that the user presses to be sent to the host immediately instead of waiting until all actions have been performed. Select the check boxes of the keys that should be sent to the host immediately. If a host key is sent immediately, no other actions will occur. The immediate sending of these keys applies only to the current transformation, and not to all transformations in the project.

You can disable project-level and screen-level global rules processing by clearing the **Apply project-level and screen-level global rules** check box.

## Execute business logic action

If you decide to run some business logic as the action of this screen event, you must specify in the fields provided the fully qualified Java class name and the Java method for the business logic you want to perform. You can click **Browse** next to the **Class name** field to select a class in which the business logic method is defined. You can select any class defined under the **Source** folder in the **HATS Projects** view tab of the HATS Toolkit. If you have not created the Java code for this business logic, right click in the **HATS Projects** view tab, and click **New HATS > Business Logic** to invoke the Create Business Logic wizard.

For more information about business logic, see the *HATS Web Application Programmer's Guide* or the *HATS Rich Client Platform Programmer's Guide*, depending on your application environment.

## Extract global variable action

You can extract information from the host screen and define it as a global variable.

For the region of the host screen, you define the starting and ending rows and columns for the area of the screen you want to assign as a global variable.

In the **Advanced** section, use the **Plane to extract** drop-down list to select the data plane to extract. The options are:

- Text
- Color
- Field
- Extended field
- DBCS
- Grid

**Note:** For information about the format and contents of the different data planes in the Host Access Class Library (HACL) presentation space model, see Host Access Class Library Planes -- Format and Content at http://www-01.ibm.com/support/knowledgecenter/SSS9FA_11.0.0/com.ibm.hod.doc/doc/hacl/DWYL0M88.HTML.

When you extract data for the global variable, you can specify a name for a new global variable or select an existing global variable name from the drop-down list for the **Name** field.

To specify how text extracted from multiple rows of the host screen is defined in a global variable, click **Advanced**. You must select one of the radio buttons to specify if the extraction should be treated as one string or as a list of strings (indexed). When you select the **Extract this region as one string** option, the extracted data is saved in a global variable as a single object in the form of a 2-dimensional character array. When you select the **Extract this region as a list of strings** option, the data is extracted as a 2-dimensional character array and broken up into individual 1-dimensional arrays, each one representing a single extracted row and stored in an index in the global variable.

If you selected an existing global variable in the **Name** field before you clicked **Advanced**, you must click one of the following radio buttons specifying how HATS should handle the extracted data:

- **Overwrite the existing value with this new value**
- **Overwrite the existing value with this new value, starting at the specified index**
- **Append this new value after the last index of the existing value**
- **Insert this new value into the existing value, at the specified index**

For the two options that use a specified index, you must enter the number of the index in the **Index** field.

The following example illustrates how the variable value is modified based on the option you select. Start with an existing indexed variable named “sample”. The values of “sample” are “a b c d”. The “a” in the value has an index of 0, therefore the value of “sample[0]” is “a”, and the “b” in the value has an index of 1, therefore the value of “sample[1]” is “b”, and so on. Assume that you extract a new set of values “e f g”.

- If you click the **Overwrite the existing value with this new value** radio button, the value “a b c d” of “sample” is changed to “e f g”.
- If you click the **Overwrite the existing value with this new value, starting at the specified index** radio button, and assume an index of 2, the value “a b c d” of “sample” becomes “a b e f g” .
- If you click the **Append this new value after the last index of the existing value** radio button, the value “a b c d” of “sample” becomes “a b c d e f g”.
- If you click the **Insert this new value into the existing value, at the specified index** radio button, and assume an index of 2, the value “a b c d” of “sample” becomes “a b e f g c d”.

Selecting the **Shared** check box will allow the global variable to be shared.

For more information about global variables, see Chapter 12, “Interacting with global variables,” on page 337.

## Insert data action

Use the mouse or the **Start row** and **Start column** settings to set the region on the host screen where you want data inserted. You can highlight certain fields on the host screen by selecting the different options beside **Highlight fields**. If you want to see where the input fields are defined on the screen, select the **Input** check box. If you want to see what fields are protected, select the **Protected** check box. If you

want to highlight any hidden fields, select the **Hidden** check box. To modify the colors of the input, protected or hidden fields highlighting, see "Using HATS preferences" on page 126.

Click **Next** then select whether the data is a string or a global variable by clicking the appropriate radio button. To insert a string, type the text in the entry field provided. To insert a global variable, select the name of an existing global variable from the drop-down list or type in the name of the global variable. If you want to use a shared global variable from a different application, you will need to type the name in the field.

If the value of the global variable is indexed (contains a list of strings), click **Advanced**. You must click one of the radio buttons to specify whether all of the strings are inserted at the specified position one after the other or if the strings are inserted as separate lines into a rectangular region of the screen.

Selecting the **Shared** check box will allow the global variable to be shared.

For more information about global variables, see Chapter 12, "Interacting with global variables," on page 337.

**Note:** Inserting information onto a host screen must occur *before* any transformation occurs for the global variable to appear in the GUI. See "Actions" on page 154 for information about modifying the order of the actions.

## Set global variable action

You can set global variables to be used by other objects within your project and also by other projects in the .ear. When you set a global variable, you can specify a name or select an existing global variable name from the drop-down list for the **Name** field. If you select an existing indexed global variable, click **Advanced** to specify how to handle the setting of the value. The following options are available:

- **Overwrite the existing value with this new value**
- **Overwrite the existing value with this new value, starting at the specified index**
- **Append this new value after the last index of the existing value**
- **Insert this new value into the existing value, at the specified index**

For the two options that use a specified index, you must enter the number of the index in the **Index** field.

For an example of how a variable value is set based on the option you select, see "Extract global variable action" on page 156.

If you are setting the global variable to a fixed value, type the value in the entry field.

If you are setting the global variable to a calculated value, you specify the operands to be used and the operation of the calculation. The operands can be either fixed values that you enter into the field, or you can use the values of existing global variables for the calculation. If you use an existing indexed global value, and you want to specify an index of the variable to use as the operand, click **Advanced**. Enter the number of the index in the **Index** field.

Selecting the **Shared** check box will allow the global variable to be shared.

For more information about global variables, see Chapter 12, “Interacting with global variables,” on page 337.

## Remove global variable action

Use this action to remove one or more global variables. You can remove one local or one shared global variable, all local global variables, all shared global variables, or all local and shared global variables. For either the **One local** or the **One shared** option, enter the name of the global variable you want to remove or select the name from the corresponding drop-down list.

**Note:** Names of shared global variables, created in another HATS Web application that runs in same .ear file or created in another HATS rich client application that runs in the same rich client environment, may not appear in the drop-down list of shared global variables. To remove one of these shared global variables, enter its name in the entry field for the **One shared** option.

For more information about global variables, see Chapter 12, “Interacting with global variables,” on page 337.

## Send global variable action

This action applies only to HATS standard (JSR 168 or JSR 286) portlet projects. It can be used to help develop a portlet communication solution. For more information, see “Portlet communication” on page 391. For considerations when using bidirectional language support, see “Portlet support” on page 449.

**JSR 168 portlets:**

Use this action to send the value of a global variable as a java.lang.String value in a property from a HATS JSR 168 portlet to other JSR 168 portlets (HATS or non-HATS) on the portal server. In the Property field enter the name of the property that is being sent. The name must match the property name used by the receiving portlet, for example, the same property name defined in **Portlet** settings on the **Other** tab in a HATS JSR 168 receiving portlet. In the Global variable field enter the name of the global variable. If the value of the global variable is indexed, click **Advanced**. Then, specify whether to **Send all indexes** of the variable, optionally separated by a string delimiter, or to **Send a single index** including which **Index** to send. Select the **Shared** check box if the global variable is a shared global variable.

When you click **Finish**, the action is added to the screen's event (.evnt) file. In addition, a Web Services Description Language (WSDL) file is created, if one does not already exist, and a send property is defined in the WSDL file. The name of the WSDL file is <project_name>.wsdl. The WSDL file is required to wire the portlets using the WebSphere Portal wiring tool. The portlet.xml file is updated to reference the location of WSDL file.

If you click **Remove**, the action is removed from the screen's event file, but the property is not removed from the WSDL file. You can edit the WSDL file and use the WSDL editor to remove the property.

**Note:** If you do not remove the property from the WSDL file, the HATS portlet runs properly as-is.

**JSR 286 portlets:**

Use this action to send the value of a global variable in an event from a HATS JSR 286 portlet to other JSR 286 portlets (HATS or non-HATS) on the portal server. In the Event name field enter the name of the event that is being sent. The name must match the event name used by the receiving portlet, for example, the same event name defined in **Portlet** settings on the **Other** tab in a HATS JSR 286 receiving portlet. In the Global variable field enter the name of the global variable.

**Note:** Event names within a single portlet must be unique.

Click **Advanced** to set more options if necessary.

On the Advanced Options page, if the value of the global variable is indexed, select **Variable is indexed**. Then, specify whether to **Send all indexes** of the variable, or to **Send a specific index**, including which **Index** to send, and the **Type** of the object contained in the global variable. Use the **Browse** button to assist in entering a valid value type. The type you select must be Serializable. If not, an error message is displayed. The type also must match the type used by the receiving portlet, for example, the same type defined in **Portlet** settings on the **Other** tab in a HATS JSR 286 receiving portlet. Select the **Shared** check box if the global variable is a shared global variable.

When you click **Finish**, the action is added to the screen's event file, and the portlet.xml file is updated. No WSDL files are involved with JSR 286 portlet communication.

**Common considerations::**

- This action must follow the Apply transformation action for the event.
- The following actions are the only actions supported between the Apply transformation action and the Send global variable action:
  - Execute business logic
  - Extract global variable
  - Insert data
  - Set global variable
  - Remove global variable
  - Perform macro transaction
  - Pause
- The above actions are useful if you want to process the global variable before sending it to other portlets. For example, you want to send the user ID entered on the login screen to other portlets. In this example, you add an Apply transformation action to transform the login screen and an Extract global variable action to extract the region of the user ID input field and save it in a global variable. Then you add a Send global variable action to send the user ID stored in the global variable to other portlets.
- The following actions are not supported between the Apply transformation action and the Send global variable action. If any of these actions are placed between the Apply transformation and Send global variable actions, they are executed as normal, but the global variable is not sent.
  - Apply transformation
  - Show URL
  - Forward to URL
  - Play macro
  - Send key

  - Disconnect
  - Portlet Prepresentation (Block Delimiter)
- The **Type** specification for the object contained in the global variable is supported only for JSR 286 portlets. If you manually add the type parameter to the screen event source in a JSR 168 project, it is ignored.

## Show URL or SWT composite action RCP-only

Use this action to show a Web page or an SWT composite in the transformation area of the transformation view. To show a Web page specify the URL (uniform resource locator) address of the Web page in the **URL** field. To show an SWT composite, enter its class name in the **Composite class** field, or use the **Browse** button to select an SWT Composite from within the rich client project. The Web page or SWT composite is shown surrounded by the template you select from the list of templates. After the Web page or SWT composite loads, the user must click a **Continue** button to return to the HATS application.

**Note:** For information about how to implement the SWT composite including a **Continue** button, see the *HATS Rich Client Platform Programmer's Guide*.

## Show URL action Web-only

If you want to show a Web page as the action of this screen event, you must specify the URL (uniform resource locator) address of the Web page in the **URL** field. The Web page is shown surrounded by the default template, similar to the way a transformation is shown. After the Web page loads, the user returns to the HATS application by clicking the **Continue** button at the bottom of the Web page.

If embedded objects are not supported or not allowed by the browser, for example, objects with data type of html, this action shows a link to the Web page instead of showing the contents of the Web page. For a list of supported Web browsers and limitations, see "System Requirements for Host Access Transformation Services" at http://www.ibm.com/support/docview.wss?uid=swg27011794 and "Host Access Transformation Services 9.7 - Known issues and workarounds" at http://www.ibm.com/support/docview.wss?uid=ibm10876092.

When you specify the target of the Show URL action, the client's workstation must have access to the target page. Because of this, you may not be able to specify a target Web page that is protected by your enterprise firewall, for example, unless the client's workstation has direct access to the target page. Be sure to give the complete URL for the target, since the client may not otherwise be able to resolve the host name of the target page. If the target page is not on the application server hosting the HATS application (a "cross-domain" access), the browser settings may need to be adjusted to allow the off-server target page to be loaded.

For example, in Internet Explorer V6.0, or later, go to the **Tools** menu and select **Internet Options**, then the **Security** tab. The browser displays the Web content zone in the lower right corner of the browser window. If the page is in the Internet zone, for example, you may need to adjust the Internet zone settings to allow the target page to load. Choose the correct zone, and then press the **Custom Level** button. Scroll to the **Miscellaneous** settings, and select a setting for **Access data sources across domains**. Select **Enable** to allow the target page to load. Select **Prompt** to be prompted before loading the target page. If you select **Disable**, you may be unable to load the target Web page.

## Forward to URL action Web-only

The forward to URL action enables you to pass control from a transformation-oriented HATS Web application to a JSP that invokes one or more

chained Integration Objects. This enables you to use Integration Objects that you have already created. The Integration Objects can use the existing connection or a background connection.

This action is supported in HATS Web and HATS standard portlet projects.

To add a forward action to an event, you must specify the following parameters:

- The JSP to which control is passed. In addition to invoking the Integration Objects, this JSP can interact with the user if information is needed.

  **Note:** If the JSP to which control is passed is a Model 1 JSP generated in a HATS standard portlet project, then the following statements must be added to the JSP:

  ```
  <%@ taglib uri="http://java.sun.com/portlet" prefix="portletAPI" %>
  <portletAPI:defineObjects/>
  ```

- The startStateLabel of the first Integration Object to be run. This value is used by the HATS application to store the default connection to ensure that it will be available to the Integration Objects.

  **Note:** This parameter is only required if the default connection needs to be passed to the JSP/Integration Object logic.

**Note:** If you plan on running a large JSP, set the JVM system property **com.sun.tools.javac.main.largebranch=true** and then restart the server. You can set JVM properties by using the administrative console, by setting up arguments for the JVM, or by creating a Custom Property.

The forward to URL allows multiple ways to manage the connection that will be used by the JSP/Integration Object logic. For example:

- The HATS application has already established a default connection and you want the JSP/Integration Object logic to use this connection. In this case, the JSP that gains control should drive an Integration Object that is not first-in-chain. The Integration Object that is driven will locate the connection to use by looking up the connection in the HTTP session object using the start state label that was associated with the Integration Object when it was created. This label should match the label that was specified as part of the forward to URL definition. In this case, note that when control is returned to the transformation-oriented project, a check will be made to see if the host screen has been updated. If it has, all remaining actions in the action list associated with the current event will be ignored. If the host screen has not been modified, the remaining actions will be honored.
- The HATS application has already established a default connection, but you want the JSP/Integration Object logic to use a new background connection. In this case, the JSP that gains control should drive an Integration Object that is either first-in-chain or not chained at all. The Integration Object will cause a new background connection to be established and perform its designated task against that connection. Note that the startStateLabel does not need to be specified in the forward to URL definition in this situation.
- A default connection has not been started. This will be the case if you add the forward to URL action to the Start event or to the Connect event (before the obtain action is performed). In this scenario, many possibilities exist. For example, the JSP that gains control can drive an Integration Object that gets its own background connection, performs a designated task, and is programmed to return control to the HATS application. At that point, the HATS application can continue with the event processing and establish a default connection. Another

possibility would be that the JSP drives an Integration Object that is first-in-chain. The Integration Object establishes a connection, performs a task, and then passes the connection to the HATS application to be used as the default connection.

- When the connection is established by a HATS standard portlet, it is saved using a key comprised of a unique connection identifier and the forward to URL action start state label. This same key must be used by the first Integration Object that gets control. The key is saved as the attribute CommonConstants.HPUB_LINK_KEY in the request object. You must edit the JSP to retrieve the link key from the request object and call the setHpubLinkKey() method on the Integration Object, before calling the IntegrationObject processRequest(), method. For example:

  ```
  ExampleIO.setHPubLinkKey
    ((String)request.getAttribute(CommonConstants.HPUB_LINK_KEY));
  ```

  When the connection is established by the first in chain Integration Object, it is saved using a link created by the first in chain Integration Object.

  In both scenarios, subsequent Integration Objects must reuse the same link. The link can be retrieved from the first Integration Object calling the getHPubLinkKey() method. If necessary, the link can be passed to subsequent JSPs as a parameter on the request object. This can be accomplished by adding a hidden input parameter to a form, as shown below:

  ```
  <INPUT NAME="<%= CommonConstants.HPUB_LINK_KEY %>"
         VALUE="<%= ExampleIO.getHPubLinkKey() %> "TYPE="hidden">
  ```

  The subsequent JSP would use the following statement to retrieve the key and set it on the IO, before calling the processRequest() method:

  ```
  ExampleIO_2.setHPubLinkKey
   ((String)request.getParameter(CommonConstants.HPUB_LINK_KEY));
  ```

When you use the forward action, control does not return automatically to the HATS application after the Integration Objects have been run. The JSP must explicitly pass control back to the HATS application. If the default connection was not made before the forward action, and the Integration Objects used the default connection, you must pass the connection to the HATS application to be used as the default connection. In this case you must set a request parameter on the HttpServletRequest before forwarding the request to the HATS application. The parameter is CommonConstants.HATS_EXISTING_CONN. You can obtain the value needed for this parameter by calling the getHPubEndChainName method on the last Integration Object in the chain.

For example, in a Web project, the FORM might look like this:

```
<FORM NAME="exampleLink" METHOD="GET"
ACTION='<%= response.encodeURL(request.getContextPath()+"/entry")%>'>
<INPUT TYPE="HIDDEN" NAME="<%= CommonConstants.HATS_EXISTING_CONN %>"
VALUE="<%= ExampleIO.getHPubEndChainName()%>" />
<INPUT TYPE="submit" VALUE="Return to HATS application" />
</FORM>
```

You can also use the **HATS Tools** menu to **Insert Forward to HATS Application**. This option will automatically insert the code with one exception. It will not include the following line:

```
<INPUT TYPE="HIDDEN" NAME="<%= CommonConstants.HATS_EXISTING_CONN %>"
       VALUE="<%= IntegrationObjectName.getHPubEndChainName() %>" />
```

If you want to pass the connection to the HATS application, you must then edit the JSP and add the line of code right after the "FORM NAME=" line of code.

In all other cases (when the default connection was opened before the forward action was invoked, or when the Integration Objects do not use the default connection), you do not have to return the connection. You can omit the first INPUT statement from the example in those cases.

**Note:** It is possible for the JSP/Integration Object logic to be the first entry into your HATS application. In that case, you can use the code that is identified to drive your HATS application and pass the connection established by the JSP/Integration Object logic to the HATS application to be used as the default connection (if the connection was created from the default connection definition).

## Play macro

If you recorded a macro, you can select it from the drop down box to play it. The macro will begin playback whenever this event is selected. It should be noted that the **Play macro** action runs a macro on the current connection. The **Perform macro transaction**, however creates a new connection to run the macro on.

To play a macro, select the **Play macro** bullet from the **Add Action** list, then select the name of the macro to play from the drop-down list. If you define a macro to be played as an action of this screen event, it is the last action applied. Be sure to use very specific criteria for recognizing that screen, to ensure that the macro will not be played on other screens. For example, if you are using a string criterion, specify that it must be found at the exact location, rather than anywhere on the screen. Make sure that the screen on which the macro ends does not satisfy the criteria for starting the macro. When you record a macro, be sure that the final screen is not the screen that is recognized by the screen event. If the recognized screen is the final screen of the macro, a loop will be created and the option to send a key to force the screen to change will not run when play macro is performed.

Ordinarily you will not apply a transformation and play a macro as actions on the same screen. If both actions are specified, the transformation will be applied, and the macro will be run only after the user interacts with the transformed screen. The playing of a macro is always the last action to be taken. If you want a macro to be played automatically when a screen is recognized, do not apply a transformation to that screen. When you create the screen event for that screen, you should clear the **Apply a transformation** check box on the Actions page of the wizard. You can create another screen event for the last host screen to which your macro will navigate, and apply a transformation to govern the appearance of the Web page derived from that final screen.

You can insert a macro button into a transformation to enable the user to run a macro from the transformed screen. In this case, the user can either interact with the transformed screen or click the macro button to cause a macro to begin running at the current host screen.

You can record macros in the HATS Toolkit using the host terminal. For more information about importing macros, see Chapter 11, “Macros and host terminal,” on page 323.

## Perform macro transaction

Selecting this action enables you to play a recorded macro on a new instance of a designated connection, even a connection to a different host. After clicking the **Perform macro transaction** button, select which macro you want to play in the **Play a macro** drop-down list and what connection to use in the **Perform on connection** drop-down list.

**Note:** If there is a connect macro defined for the designated connection, the connect macro will be run before your selected macro is run.

### Send key

This action sends a specified key to the host screen. Once you have selected the location on the screen to apply your action, select which key to send to your host screen.

Select either **PF and PA keys** or **Other host keys** from the bulleted list. Then click the drop-down menu of the bullet you selected and select the key.

Keep in mind that the key you select must change the host screen in some way or an infinite loop of the action results.

The **Send key** action must always run last when considering the order of your action list.

### Disconnect

The **Disconnect** action immediately performs the disconnect event. By default, the disconnect event disconnects and releases the default connection.

### Pause

The **Pause** action allows you to specify the amount of time (in milliseconds) before continuing with normal processing.

### Portlet Prepresentation

When using **Portlet Prepresentation**, you can add **Prepresentation Begin** and **End** delimiters using the **Add Block Delimiter** button on the **Actions** tab.

This action is not supported for HATS standard portlets.

The **Prepresentation Begin** and **End** allow you to enclose actions that must be performed in the prepresentation phase of portlet processing. Use the **Add Block Delimiter** button when implementing portlet messaging or cooperative portlets. Click the **Add Block Delimiter** button once to insert the prepresentation begin delimiter and click **Add Block Delimiter** again to insert the prepresentation end delimiter. Use the **Up** and **Down** buttons to move the action needing execution in the block delimiter higher or lower in the list. When the action needs to be invoked will determine the location of the prepresentation block. When executing a screen event, you can trigger the prepresentation action first or at the top of the **Actions** list. If you want to run the prepresentation action after user interaction on the screen, you would trigger the event at the bottom of the **Actions** list. For more information about how to use the block delimiter with portlets, see the *HATS Web Application Programmer's Guide*.

## Global Rules

Global rules enable pattern recognition and transformation of host input fields and work with customized and non-customized (default rendered) screens. They can be defined at both the project level and at the screen level. Use this tab to specify screen-level global rules. See "Global rules" on page 94 for details on how to define a global rule.

Screen-level global rules allow you to use default rendering for a screen and in addition customize the input fields on the screen. For example, by using the **Find input fields within a specified region** pattern type in your screen-level global

rule, you can pinpoint an input field to customize on a screen, for example data within a table, while still using default rendering for the screen.

If a project-level and a screen-level global rule are both defined for the same input field, the screen-level rule takes priority.

## Text Replacement

The **Text Replacement** tab displays a table in which you can specify the original protected host screen text that you want to replace, together with the text, HTML content **Web-only**, or image to use as the replacement. Also shown is whether the text search is case-sensitive and whether regular expression support is being used. Click the **Add** button to create new text replacement parameters.

You can either select the text you want to replace from the selection window, or type in the text in the **Replace** text box.

If the **Case sensitive** check box is selected, it will only search for exact text that you entered in the **Replace** text box.

To replace with text, type in the text you want replaced in the window below.

If you enter HTML code **Web-only**, HATS will highlight it in red allowing you to know whether your code is valid.

Clicking the **Insert** icon lets you add a button or a link. The settings for **Insert Button** and **Insert Link** are:

**Caption**
The text that you want to appear on the rendered buttons or links

**Action key**
The host AID key to send to the host when the buttons or links are clicked.

**Style class** **Web-only**
The cascading stylesheet (CSS) class name that controls the appearance of the buttons or links. The default for buttons is 'HATSBUTTON'. The default for links is 'HATSLINK'

You can also replace text with images by clicking the **Image** radio button and selecting one from the drop-down list, or clicking the **Import** button.

Selecting the **Regular expression** check box allows you to use Java regular expression support as part of the text replacement algorithm. Regular expressions are patterns of characters that describe a set of strings. You can use regular expressions to find and modify occurrences of a pattern. For example, specifying:

```
Replace: ([\.\-|_\w]+)@([\-\_\.\w]+)
With: <a href="mailto:$1@$2" class="HATSLINK">$1@$2</a>
```

would result in the following replacement, which converts an e-mail address on a host screen to a link.

```
Original host screen text: user@company.com
Replacement text:
<a href="mailto:user@company.com" class="HATSLINK">user@company.com</a>
```

You can add, modify, or remove any text replacement specifications by using the buttons to the right of the table of values. The text can be replaced at the project level, rendering item level, transformation level as well as the component level.

If you have a list of text replacements, text or HTML created by one text replacement might be replaced by a subsequent text replacement. For example, if you replace a string on the host screen with `<img src="yourImage.jpg">`, and a subsequent text replacement replaces `sr` with another string, the `<img>` tag will no longer work properly, because the `src` parameter has changed. You can avoid this situation by reordering your text replacement items or by making them more specific, to ensure they do not accidentally replace previous strings.

**Notes:**

1. Care should be taken when using text replacement. Text replacement with a disparate number of characters in the strings can cause changes in the GUI representation of the screen. Depending on the widget used for presenting a region of a screen, text on a line of the screen can be contracted, expanded, or forced to a new line. Also, it is generally not efficient to replace strings consisting of a single character or text that has protected fields between it.
2. By default, text replacement truncates strings that are longer than the text being replaced. Sometimes the text is truncated to maintain the layout of the original host screen on the transformation. Truncation only occurs on text rendered using the field widget (or in some cases of the table widget).

   To enable text replacement of short strings with longer strings, you can add a source setting, **truncateToPreserveLength**. The values for this setting are `true` and `false`. The default value when the setting is not specified is `true`, which has no effect on your project. Specifying `false` enables the longer text replacement. To use this setting, you must not be using default rendering. If you use default rendering, the setting is ignored.

   The **truncateToPreserveLength** setting is a project level setting. It affects all text replacements in the project. If you use this setting, verify that all of your renderings appear as you want them and that any alignment issues that might arise are not problems.

   To enable the setting for the entire project, open the source view of the application.hap file. Locate the class for the transform, and add the highlighted setting as shown in the following example:

   ```
   <class name="com.ibm.hats.transform">
      <setting name="truncateToPreserveLength" value="false" />
   </class>
   ```
3. Using the selection text window can be very helpful but the actual text which appears in the **Replace** field is what will actually be replaced. Text replacement cannot be applied across multiple lines or non-protected fields. You cannot select and replace multiple lines of text by using the preview field.
4. To replace text with images when the HATS application is accessed through a proxy server, you must configure the HATS application to use the proxy server. For instructions see “Configuring HATS applications to use a proxy server” on page 35.
5. Limitations exist relating to the use of replacing text with images in combination with the use of certain SWT widgets in rich client applications. For more information see “HATS rich client considerations and limitations” on page 84.

## Next Screen

The **Next Screen** tab allows you to identify the next likely screens to occur after the screen you are on.

Click the **Add** button to add screens to the list. You can use the **Up** or **Down** buttons to modify the order in which your screens are compared or the **Remove** button to remove a screen.

If none of the next likely screens in the list match, your application can either **Search application event priority list** or **Execute application event**.

The **Search application event priority list** option searches the event priority list in your project settings for the next screen. This is the default. The **Execute application event** drop-down list allows you to select an error (or other) event if none of the next likely screens in the list match. The events listed are:

- Unmatched screen
- Error
- Disconnect
- Stop

For more information, see "Application events" on page 102.

Using the **Next Screen** can improve performance if you can predict what screen likely appears next.

### Source

The **Source** tab displays the tags and values in the *se-name*.evnt file for all the information supplied for the screen event, where *se-name* is the name you gave to the screen event when you created it. As you make changes on other tabs in the screen event editor, the tags and values displayed under the **Source** tab change to match.

You can also make changes to the tags and values in the source file, and they are reflected on the appropriate tabs of the screen event editor.

## Inhibited screens

The operator information area (OIA) status of a HATS connection is one of the screen matching criteria used by HATS to determine if a screen matches one of the configured screen events. If the OIA status is not a match, then HATS will not match the corresponding screen event, and you will see the default transformation instead of the custom transformation or action. This section provides methods that can be used to handle screens that are input-inhibited (keyboard locked) or that have a nonmatching OIA status.

### Recognition criteria

HATS, by default, adds the following recognition criteria for the OIA status to all screen events:

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

This means that the OIA status must be `NOTINHIBITED`, in addition to the other matching criteria, for HATS to evaluate the screen as a match. If the OIA status is anything else, the screen will not be considered a match even if other criteria are met. As a result, the default transformation will be displayed in the browser.

To have HATS ignore the OIA status for a specific screen event, simply change the recognition criteria from `NOTINHIBITED` to `DONTCARE`, by following these steps:

1. In the HATS Projects view, expand your project and the list of screen events (screen customizations or screen combinations).
2. Double-click on the screen event to start the screen event editor.
3. At the bottom of the editor, click the **Source** tab.
4. Find the line that contains the string `status="NOTINHIBITED"`. Change `NOTINHIBITED` to `DONTCARE`.
5. Save the change by clicking **File > Save** or pressing **Ctrl+S**.

If the host screen is in fact inhibited, or locked, you must press a specific key or key combination, such as Ctrl+R (host Reset) or Escape (host Clear), to unlock the screen or to move to the next screen.

## Automated handling of inhibited screens

You can enable HATS to automatically recognize and handle an inhibited, or locked, screen by creating a special screen customization for that purpose. To do that:

1. On the Screen Recognition Criteria tab for the screen customization, clear and remove all criteria.
2. On the Source tab for the screen customization change the OIA line to have `invertmatch="true"`. When you are done, it should look like this:

   ```
   <oia invertmatch="true" optional="false" status="NOTINHIBITED"/>
   ```
3. On the Actions tab of the screen customization, add an action that will clear the OIA status and restore it back to `NOTINHIBITED`. The correct action to be taken may be different in each case. For example, in some cases simply sending a host key of Enter (or some other host key) may be sufficient to clear the OIA status and restore it back to `NOTINHIBITED`. However, in other cases, it may be necessary to run a macro that will change the cursor position or send a sequence of several host keys to return the host connection to a state in which the OIA status is `NOTINHIBITED`.
4. In the HATS Project Settings editor, select the **Events** tab and then sort the Screen Event Priority list such that the special screen customization is at the top of the list. This ensures that HATS will detect and handle screens with an `INHIBITED` OIA status first, before attempting to match any other screen event.

## Handling multiple inhibited screens

In some cases there may be more than one inhibited, or locked, screen, each with a different action needed to clear that condition. It may therefore be necessary to create and use more than one special screen customization to handle such cases. In addition to changing `invertmatch` to `true` on the OIA line on the Source tab, it may be necessary to add other criteria on the Screen Recognition Criteria tab, so that each screen customization is unique and will only be triggered when appropriate.

For example, a host screen may become locked if the user attempts to type text into a protected field. In this case, perhaps a `Cursor in protected area of display` message appears. To clear this condition, the user would have to move the cursor to a non-protected area. A special screen customization can be created to automate this. In addition to having `invertmatch="true"` for the OIA area, it would also look for the message text `Cursor in protected area of display`. If a match exists, the customization can have the action of running a macro to move the cursor out of the protected area and thus clear the OIA status.

On the same system, perhaps there is a second condition in which the screen may become locked if, for example, the user presses an invalid function key. In this

case, the text on the error screen says `Invalid function key pressed`. To clear this condition, the user might have to press the Enter key. Another special screen customization could be created to automate this. It would also have `invertmatch="true"` for the OIA area, but it would match the text `Invalid function key pressed`. For the action, this screen customization would send the Enter key to the host.

Depending on the specific environment, it may be easiest to handle inhibited screens by simply setting the OIA criteria to `DONTCARE` and allowing the users to handle the situation just as they would with a terminal emulator. As shown above, however, you can also enable HATS to automatically recognize and handle inhibited screens if you create one or more special screen customizations.

## Importing BMS map sets

HATS enables you to create screen captures from Customer Information Control System (CICS®) Basic Mapping Support (BMS) map sets. BMS maps are screen definitions files for CICS. Each map defines all or part of a screen and a CICS application typically displays one or more maps to create a complete screen image. Maps contain both static and dynamic areas or fields.

The source for BMS maps is organized in groups called map sets. One map set contains one or more maps. Map sets exist in source form as one map set per source file. When HATS imports BMS Maps, the import takes place at the map-set level. It is not possible to import an individual map.

Maps initially exist on the host system and must be placed on your local file system (or appear to be on the local file system) and have an extension of .bms before being imported to HATS.

You must first copy the BMS map sets from the host to your local file system. This is a manual step outside of HATS that can be done by either physically copying the files to the file system or by connecting the file system to the host using networking software. The file should not be transferred as a binary file. It must be readable locally on your workstation.

1. Click **File > Import > HATS > BMS map sets into HATS** to open the Import wizard.
2. Select the BMS map set files you want to import from the list and the destination of the imported files.
3. Click on the pull down arrow to select the country **Host code page**. This is the code page that was used to create the BMS map on the host.
4. Click on the pull-down arrow to select the **BMS file code page**. This is the code page used for the BMS file when it was transferred to the workstation.
5. You have the option to **Overwrite existing maps without warning** and to **Generate one screen capture per map using the map name as the screen capture name**.

   **Note:** You may not want to automatically create screen captures if any of the maps need to be combined with other maps to form a screen.
6. You can also select **Overwrite existing screen captures without warning**.
7. Click **Finish**.

BMS capture files are stored in a separate **Maps** folder within the HATS project. By default, the **Maps** folder is not visible in the HATS Projects view unless there are

imported maps. Within the **Maps** folder is a separate folder for each map set, and within each folder, a separate file for each map (using extension .bmc). Additionally, the original source is stored in the file system (extension .bms) and is shown in the folder containing its associated maps. You can edit the source by double clicking the file. If the source is edited and saved, the maps are automatically regenerated and overwritten except for maps that have been modified through the **Properties** view.

You can also import BMS maps by dragging a BMS source file into the <*project*>/bmsmaps folder in the **Navigator** view. The file must be placed directly in the bmsmaps folder and not in any subfolders. If errors occur while importing maps this way, messages will be displayed in the **Problems** view.

**Note:** The source is saved as a reference only and has no function within the project other than to be displayed in the editor. When it is displayed, the entire map set is displayed and no attempt is made to highlight the definition of the particular map that is being displayed in the editor.

After BMS maps are imported, there are several actions that can be taken from the HATS Projects view.

- Hovering your mouse cursor over the map name displays a small image of the map.
- When you click on a map in the HATS Projects view, the **Properties** view is updated to show details about the selected map. If there are named fields in the map, you can use the **Properties** view to modify the contents of those fields. By making copies of maps to alter the contents of named fields within the copies, you can create different customized versions of maps that accurately reflect the way the screen will be shown by the application.
- Double clicking on a map brings up an editor page that shows how the map would display on a 3270 screen. Double clicking on the map set folder name will show the map source in an editor.
- Right clicking on a map name in the HATS Projects view adds a **Generate Screen Captures** selection to the pop-up menu. Selecting this launches the Generate screen captures wizard that can be used to select one or more BMS capture files to be used to create a screen capture. You can elect to create separate screen captures for each BMS map selected or merge selected BMS maps into a single screen capture. Maps cannot be merged if fields overlap. The **Overwrite existing resources without warning** check box can also be selected if needed.

  **Note:** If separate screen captures are generated and more than one file is selected, the names of the screen capture files are generated automatically.

Once your screen captures have been created, you can begin to create HATS screen events.

# Chapter 8. Working with transformations

A transformation is either a JSP file (for Web projects) or an SWT composite (for rich client projects) that defines how to customize specific host screens. When you create a transformation, you have the option of selecting the host screen on which the transformation will be based. The host screen can be based on a host terminal screen, or from a screen capture. A screen capture can be obtained from an imported BMS map (3270 only) or directly from the host terminal.

Applying a transformation is one of the possible actions of a screen event. Some common uses of transformations are:

- Rearranging the presentation of host screen information.
- Filtering host screen information that you do not want to show to users.
- Presenting host components as widgets in the Web presentation.

## Create a Transformation wizard

Use the Create a Transformation wizard to define screen transformations. You can access the wizard a number of ways, such as:

- Right click on any screen capture in the **Screen Captures** folder and select **New HATS > Transformation**.
- Click the **Create HATS Transformation** icon on the HATS toolbar.
- Select **HATS > New > Transformation** (or **File > New > HATS Transformation**) on the HATS menu bar.
- Any place you can create a screen customization, you can create a transformation. For more information, refer to Chapter 7, "Working with screen events," on page 147.

The Create a Transformation wizard appears and enables you to select the target project, name the transformation, override the package name (for HATS rich client projects), give it a description, and see where the transformation definition is saved. Click **Next** when you have specified these items.

On the **Select Screen** panel you have the option of selecting the host screen on which the transformation will be based. From the **Select a screen** section, you can select **Use the host terminal screen** or **Use a previously captured screen** from the pull-down menu. You can highlight certain fields on your host screen by selecting the different options beside **Highlight fields**. If you want to see where the input fields are defined on the screen, select the **Input** check box. If you want to see what fields are protected, select the **Protected** check box. If you want to highlight any hidden fields, select the **Hidden** check box. To modify the colors of the input, protected or hidden fields highlighting, see "Using HATS preferences" on page 126. When your selections have been made, click **Next**.

There are four rendering patterns from which you can select, **Blank**, **Default Rendering**, **Prepopulated using Default Rendering**, and **Prepopulated using Fields**.

**Blank** Creates a new blank transformation. You can later add individual components using the Insert Default Rendering or Insert Host Components wizards.

**Default Rendering**

Creates a new transformation that displays with default rendering at runtime. Use this pattern if you want to use a rendering set different from the default, change the region of the screen to be rendered, or add additional components using the Insert Host Components wizard. Click **Options** to configure the following setting on the Rendering Options panel:

**Use the default rendering set**

Select this box to use the default rendering set of the application when the transformation is rendered at runtime. Clear this box to select a specific rendering set to always use.

**Prepopulated using Default Rendering**

Creates a new transformation prefilled with component tags, generated using default rendering, which can be customized using the Edit Host Component wizard. Click **Options** to configure the following settings on the Rendering Options panel:

**Use the default rendering set**

Select this box to use the default rendering set of the application when the transformation is rendered at runtime. Clear this box to select a specific rendering set to always use.

**Include empty, protected fields**

Select this box to generate tags (for Web transformations) or component rendering composites (for RCP transformations) for empty, protected fields. In most cases, the default (not generating tags or composites for these fields) is sufficient, and provides more flexibility for moving other controls around the transformation.

**Note:**
In general, when creating a prefilled transformation, you should capture a screen that has as much information displayed as possible. For example, if you are creating a transformation for a screen containing a table, you should capture a screen that has the entire table of data filled in. This will cause tags (for Web transformations) and composites (for RCP transformations) to be generated for all fields in the table.

**Prepopulated using Fields**

Creates a new transformation prefilled with labels and input fields which can be customized using the Edit Host Component wizard.Click **Options** to configure the following setting on the Rendering Options panel:

**Include empty, protected fields**

Select this box to generate tags (for Web transformations) or component rendering composites (for RCP transformations) for empty, protected fields. In most cases, the default (not generating tags or composites for these fields) is sufficient, and provides more flexibility for moving other controls around the transformation.

**Note:**
In general, when creating a prefilled transformation, you should capture a screen that has as much information displayed as possible. For example, if you are creating a transformation for a screen containing a table, you should capture a screen that has the entire table of data filled in. This will cause tags (for Web transformations) and composites (for RCP transformations) to be generated for all fields in the table.

Use the **Preview** button to display the transformation using the selected pattern.

After the patterns are laid out you can arrange the components inside your transformation. If you want to launch the screen customization wizard, select the **Start the Create Screen Customization wizard when done** check box, then click **Finish**. For more information, see: "Default rendering" on page 90.

## Editing transformations

Transformations in HATS Web projects are JSP files and by default are edited using the Rich Page Editor. Transformations in HATS rich client projects are SWT composites and by default, are edited using the Window Builder.

### Editing transformations for Web projects

You can see the transformations you have created by expanding the **Web Content/Transformations** folder in the **HATS Projects** view. To edit a transformation using the Rich Page Editor that is built into Rational SDP, double-click on the name of the transformation. To see other available editors, right-click on the name of the transformation and select **Open With**. See Rational SDP documentation for the Rich Page Editor by selecting **Help > Help Contents** from the menu bar and then search on `Rich Page Editor`.

**Note:** HATS does not support the Window Builder for editing Web transformations.

The following limitations apply to editing transformations:

- Transformations must be UTF-8 encoded.
- Do not use any JSP variable, CSS class, HATSForm, or other object whose name starts with HATS, hats, or Hats. These names are reserved for use by HATS.
- If you drag an input field outside a <HATS:form> tag, it will not be processed and submitted to the server.

**Note:** When you copy a transformation from one HATS project to another, the screen capture file associated with the transformation is not copied. The copied transformation will work at runtime without the associated screen capture file. However, you cannot preview the copied transformation in the HATS Toolkit without the associated screen capture file.

In order to preview the copied transformation, you must copy the associated screen capture file by selecting it in the **Screen Captures** folder. Once you have copied the screen capture file, go back and select the copied transformation from the **Transformations** folder and open the transformation. Go to the **Design** tab and right-click anywhere in the window. Select **HATS Tools > Edit Host Component**. Once the wizard comes up, select the screen from the drop-down menu and click **Finish**.

When editing a transformation, you can change the properties of HATS components, or other controls (such as buttons, images, and links), by using the **Properties** view. The Properties view is located at the bottom area of the transformation editor. You can also access the **Properties** view by selecting **Window > Show View > Properties** from the HATS menu.

The **Palette** view can also be used to add HATS components to your transformation. You can select the component from the drop-down menu and drag it to your transformation. The **Palette** also contains different tags (such as HTML,

form tags, JSP) for easy editing. This view can also be launched by selecting **Window > Show View > Palette** from the menu bar.

**Note:** Custom components, ENPTUI, and light pen components do not appear in the HATS Components section in the **Palette** view of the Rich Page Editor. When enabled they are accessible from the Rational SDP menu bar by clicking **HATS Tools > Insert Host Component**.

The following sections describe each tab of the Rich Page Editor.

## Design

The **Design** tab of the Rich Page Editor displays the current WYSIWYG view of the transformation as you make changes to it.

When viewing your JSP file on the **Design** tab, JSP symbols appear in the top left of the file. To hide these symbols, go to **Window > Preferences** and find the **Web** preference. Go to **Web > Page Design > Appearance > Editing Symbols** and clear the **JSP** check box.

**Note:** For considerations when using GB18030 national language characters, see "Using code page 1388 (GB18030)" on page 431.

While on the **Design** tab, you can insert or edit items on the transformation using either the HATS Tools menu on the HATS Toolkit menu bar or the Palette.

The **HATS Tools** menu contains the following items, each of which launches a transformation wizard. For information about each wizard see "Transformation wizards" on page 178.

- **Insert Host Component**
- **Edit Host Component**
- **Transform for Dojo Editing** (no GUI wizard is associated with this tool)
- **Insert Default Rendering**
- **Edit Default Rendering**
- **Insert Tabbed Folder**
- **Insert Macro Key**
- **Insert Global Variable**
- **Insert Operator Information Area**
- **Insert Host Keypad**
  - **Default Keypad**
  - **Custom Keypad**
  - **Individual Key**
- **Insert Application Keypad**
  - **Default Keypad**
  - **Custom Keypad**
  - **Individual Key**
- **Insert All Host Components**
- **Insert Stored Screen**
- **Insert Integration Object Properties**
  - **Input**
  - **Output**

- **Insert Forward to HATS Application**

The last two menu items (**Insert Integration Object Properties**, and **Insert Forward to HATS Application**) are described in Chapter 13, "Using Integration Objects," on page 341.

The HATS Components drawer in the Palette contains the following components:

- **Command line**
- **Field**
- **Function key**
- **Input field**
- **Input field with hints**
- **Item selection**
- **Selection list**
- **Subfile**
- **Table**
- **Table (field)**
- **Table (visual)**
- **Text**
- **URL**

**Note:** If you edit a transformation and attempt to insert a HATS component between two consecutive text characters, sometimes the tag is not inserted directly at the cursor location.

If you want to add images to your project, it is recommended that you import them into the Web Content/Common/Images directory of your project. To import images, click **File > Import > General > File System** to open the Import wizard. Select the location of the image source files you want to import in the **From directory** field. Select the *project_name*/Web Content/Common/Images directory as the **Into folder**. When your image source files are imported, right-click the on the **Images** folder, and select **Show Thumbnails** on the **Thumbnail** tab in the lower right window to see the images. You can use the drag-and-drop method to copy images into the **Design** tab view of your transformation.

## Source

The **Source** tab displays the HTML and JSP tags in the *transformation_name*.jsp file necessary for extracting host components from the host screen, the widgets you selected to present those host components, and any other items you added to the transformation. As you make changes on other tabs in the Rich Page Editor, the tags and attributes displayed in the tags of the source file change to match.

You can also make changes directly to the tags and attributes in the source file, or you can insert items using the **HATS Tools** in the context menu by right click on source. The items you can insert on the **Source** tab are the same items listed on the **Design** tab. Place your cursor in the source file at the point you want to insert one of the menu items and then select an option from the **HATS Tools** drop-down menu.

When a host component and its rendering widget have been inserted, you can use the **Edit Host Component** option in the **HATS Tools** to modify the host component and widget. Before you click **Edit Host Component**, make sure your cursor is inside the <HATS:Component> tag.

When you make changes to the file displayed on the **Source** tab, they are reflected on the other tabs of the Rich Page Editor.

### Editing transformations for rich client projects

Refer to the chapter https://www.ibm.com/support/knowledgecenter/en/SSXKAY_9.7.0/com.ibm.hats.doc/rcppgd06.htm in the *HATS Rich Client Platform Programmer's Guide* for more information.

## Transformation wizards

### Insert Host Component

With the Insert Host Component wizard, you select the screen from which you want to extract a host component. You also select a region on the screen from which to extract a host component by drawing a rectangle around the text. Place your cursor at any point on the screen, click and hold the left mouse button, and move the cursor to another location on the screen to draw the rectangle. The fields at the bottom of the wizard show the starting and ending row and column numbers of the rectangle. You can also enter the row and column numbers by typing the numbers in the fields.

You can highlight certain fields on your host screen by selecting the different options beside **Highlight fields**. If you want to see where the input fields are defined on the screen, select the **Input** check box. If you want to see what fields are protected, select the **Protected** check box. If you want to highlight any hidden fields, select the **Hidden** check box. To modify the colors of the input, protected or hidden fields highlighting, see "Using HATS preferences" on page 126. When you have selected the starting and ending row and column numbers of the screen, click **Next** to display the rendering options for the host components found in the selected region.

HATS provides host components and widgets. You can select one of the host components and widgets provided or you can create your own custom host components and widgets. For more information about host components and widgets see Chapter 9, "Component and widget descriptions and settings," on page 187.

Click one of the components in the **Components** list. The Component Preview window displays the component if it is found in the screen region. If you want to use text replacement for the component, click the **Text Replacement** icon to the right of the component list to open and edit the settings. For more information about text replacement, see "Text Replacement" on page 166.

You select the widget to use to render the host component from the widgets in the corresponding **Widgets** list. When you select a widget, the Widget Preview window displays how the widget is displayed in the final Web page. A larger widget preview is available if you click the **Widget preview in large window** icon (the magnifying glass).

You can override the default project-level settings for individual instances of components and widgets by clicking the **Component Settings** and **Widget Settings** icons to the right of the **Components** and **Widgets** lists. The default project-level settings for components and widgets are configured using the **Rendering** tab of the project editor. See "Rendering" on page 90 for more information.

Click the **Advanced settings for rendering** icon to the right of the **Widgets** list to specify advanced rendering options. The project-level default advanced rendering settings are configured using the **Rendering** tab of the project editor. See "Rendering" on page 90 for more information.

For HATS Web projects, clicking **Full page preview** shows all the components on the page along with the associated template. This preview shows the page as it will appear to the user.

**Note:** When you insert a HATS component into a free layout table of a transformation, the **Full page preview** function in the **Insert Host Component** wizard always shows the new component at the top of the free layout table. The preview can be misleading, the component will be inserted at the right location inside the table once you click **Finish** in the wizard.

For HATS rich client projects full page previewing is not supported in the **Insert Host Component** wizard. To see a full page preview use the preview functions located in one of the following areas of the HATS Toolkit:

- Screen capture editor preview tab
- Host terminal preview tab
- Screen customization wizard / Select actions panel preview button
- New transformation wizard / Rendering options panel preview button.

For information about the settings that can be customized with the Insert Host Component wizard, see "Component and widget settings" on page 187. The widgets that are available depend on the selected host components. Table 2 on page 307 lists the existing HATS host components and their corresponding widgets.

If HATS does not find the component in the screen region, the Component Preview window displays the message `No` *`component_name`* `components where found in the specified region`, where *`component_name`* is the component selected in the **Components** list. If this message is displayed, you might have selected a region that does not contain the complete component, or you might need to use the **Component Settings** dialog to modify the settings of the component to match the way your host application displays the component. For example, you might have a Command Line component in the region, but your command line uses the token >>> instead of ==>. You can change the token attribute of the Command Line component to look for a command line with the correct token.

Click **Finish** when you have made your component and widget selections.

## Edit Host Component `Web-only`

With the Edit Host Component wizard, you may select a different screen to look for components from the **Select a screen** drop-down list. You may also select a different region on the screen from which to extract a host component by drawing a rectangle around the text.

**Edit Host Component** has the same features as **Insert Host Component**. It allows you to edit a tag that you have already inserted such as an already placed <HATS:Component> tag. For more information, see "Insert Host Component" on page 178.

**Note:** For rich client transformations, use the **Properties** view to edit host component composites.

## Transform for Dojo Editing Web-only

Use this HATS tool if you want to customize a HATS Dojo widget that you have previously added to render a host component. This tool modifies the tags used by the HATS Dojo widget to enable you to do your own customization. For examples of how to customize HATS Dojo widgets, see Customizing a HATS Dojo widget in the *HATS Web Application Programmer's Guide*

## Insert Default Rendering

With the Insert Default Rendering wizard you can select the screen you want to use from the **Screen** drop-down list and the rendering set from the **Select a rendering set** drop-down list. When inserting a default rendering, the area defined will render the items in the order of the rendering sets. For more information, see "Default rendering" on page 90.

The fields at the bottom of the wizard show the starting and ending row and column numbers of the rectangle. You can also enter the row and column numbers by typing the numbers in the fields. You can highlight certain fields on your host screen by selecting the different options beside **Highlight fields**. If you want to see where the input fields are defined on the screen, select the **Input** check box. If you want to see what fields are protected, select the **Protected** check box. If you want to highlight any hidden fields, select the **Hidden** check box. To modify the colors of the input, protected or hidden fields highlighting, see "Using HATS preferences" on page 126. When you have selected the starting and ending row and column numbers of the screen, click **Finish** and the section will be inserted in your screen on the **Design** tab.

## Edit Default Rendering Web-only

To edit the default rendering, click the section you want to edit on the **Design** tab, then select **Edit Default Rendering** from the **HATS Tools** menu. This will launch the Edit Default Rendering wizard where you can select the screen you want to edit from the **Screen** drop-down list and the rendering set from the **Select a rendering set** drop-down list. When editing a default rendering, the area defined will render the items in the order of the rendering sets. For more information, see "Default rendering" on page 90.

The fields at the bottom of the wizard show the starting and ending row and column numbers of the rectangle. You can also enter the row and column numbers by typing the numbers in the fields. You can highlight certain fields on your host screen by selecting the different options beside **Highlight fields**. If you want to see where the input fields are defined on the screen, select the **Input** check box. If you want to see what fields are protected, select the **Protected** check box. If you want to highlight any hidden fields, select the **Hidden** check box. To modify the colors of the input, protected or hidden fields highlighting, see "Using HATS preferences" on page 126. When you have selected the starting and ending row and column numbers of the screen, click **Finish** and the section will be inserted in your screen on the **Design** tab.

**Note:** For rich client transformations, use the **Properties** view to edit default rendering composites.

## Insert Tabbed Folder Web-only

**Note:** HATS tabbed folder support is deprecated in HATS V9.7. While support for tabbed folders continues for now, IBM reserves the right to remove this capability in a subsequent release of the product. Some alternatives are:

- Use the Dojo TabContainer widget to create tabs. For more information see Using the Dojo TabContainer widget in the *HATS Web Application Programmer's Guide*.
- Create custom DIV sections and JavaScript to control the visibility of the individual DIV sections. Tools and examples are widely available on the Internet.

Use **Insert Tabbed Folder** to insert a folder with tabs into your Web page. Tabbed folders are helpful in organizing your widgets and information to display on the Web page. With **Insert Tabbed Folder**, you specify how many tabs you want for your folder. Combining the tabbed folder and the stored screen function enables you to have a folder with a different host screen on each tab. For each tab in the folder, you also specify the following settings:

- The label text for the tab
- The host components you want to display on the tab
- The background color for the tab when it is not selected
- The background color for the tab when it is selected

Under **Tab advanced options**, if you clear the **Use default values** check box, you can specify the following settings:

- The color of the text on the tab page when the tab is selected
- The color of the text on the tab page when it is not selected
- The color of the tab page when you place your cursor over the tab

Under **Folder advanced options**, if you clear **Use default values** check box, you can specify the following settings:

- The height of the tab in pixels
- The width of the folder in pixels
- The height of the folder in pixels
- The color of the folder outline

The Preview window shows how the tabbed folder will appear, based on the selections you make. This preview does not show the components that have been inserted into the tabbed folder. It only displays changes to the folder such as size and color.

You can click **Full page preview** to show all the components on the page along with the associated template. This preview shows the page as it will appear to the user.

Click **OK** when you have defined all of the tabbed folder options for each tab in the folder.

Once you have created a tabbed folder, you can not come back to the Insert Tabbed Folder wizard to make changes. You need to edit the source because the output is in HTML (and HATS component tags).

**Notes:**

1. Because a tabbed folder is made up of several HTML tags, it is not recommended to try to move the individual sections. If you want to change the location of your tabbed folder, select the cell that surrounds it and move the cell to the location you want.
2. For rich client transformations, to insert a composite for a tabbed folder, use the SWT TabFolder widget located in the SWT Containers folder in the **Palette** view.

## Insert Macro Key

With Insert Macro Key, the user can run the macro on a transformation by clicking on a button or a link, or by selecting the macro from a drop-down list. For example, your transformation can present a logon screen that also has a button for a logon macro. When the user clicks the button, the macro plays to supply a user ID and a password, and navigates to the next screen that the user needs to see.

To add a macro to your transformation, select the macro you want to add. For Web projects you can select multiple macros within the same instance of the wizard. For rich client projects you can select only one macro for each instance of the wizard. You must also define how to display and initiate the macro from the transformation. Choose from one of the following options:

- Button
- Link
- Dropdown list `Web-only`

**Note:** HATS uses the description of the macro as the text inserted into the transformation for any of the rendering options but you can still change the name by editing the text in the **Source** view.

## Insert Global Variable

Insert Global Variable allows one of the following options:

- The insertion of a global variable into a transformation as text.
- A prompt for a global variable with input fields.

You select a defined global variable whose value you want to display from the drop-down **Name** list.

You then select how you want the global variable displayed at runtime. If you select **Display global variable value as static text**, you can click the **Advanced** button and the **Variable is indexed** check box. Then use the radio buttons to specify whether all indexes or only a single index is inserted. If you click the **Show all indices** radio button, then select the **Separator** from the drop-down list. If you click the **Show a single index** radio button, then specify the number of the index to insert. Selecting the **Shared** check box will take the global variable from the shared list.

If you want to prompt the user using an input box, select **Prompt for global variable with input box**. Select **Set initial value from global variable** to insert an initial value in the input field. Selecting **Mask as password field** will mask what the user inserts in the input field.

## Insert Operator Information Area Web-only

Select **Insert Operator Information Area** to display any available information you specified on the **Rendering** tab of your **Project Settings**. For more information, see "Operator information area" on page 99.

## Insert Host Keypad

Host keypads can be inserted in a number of ways. You can insert default or custom host keypads, or select an individual key for your transformations.

### Default Keypad Web-only

Select **Insert Host Keypad > Default Keypad** to add a default host keypad into a transformation. An inserted host keypad is only visible in the **Preview** and **Design** views if your project settings are configured to display the default host keypad in your HATS application. To configure your project to show the default host keypad and define which keys to include, go to the **HATS Projects** view, double click the **Project Settings** of your HATS project, select the **Rendering** tab and click **Host Keypad**.

### Custom Keypad Web-only

Select **Insert Host Keypad > Custom Keypad** to add a custom host keypad into a transformation. Custom host keypads use the same keys specified in the default host keypad settings, located in the **Project Settings** of your HATS project. With a custom host keypad, you can edit the attributes of individual keypad buttons or links in the **Design** view by highlighting the button or link, right-clicking, and selecting **Properties**.

### Individual Key

Use this option to add an individual host key to your transformation. From the **Insert Host Key** panel, select which host key to insert and whether you want it displayed as a button or link, then click **OK**. For Web projects you can select multiple host keys within the same instance of the wizard. For rich client projects you can select only one host key for each instance of the wizard. After inserting, you can edit the button or link using the **Properties** tab.

## Insert Application Keypad

Application keypads can be inserted in a number of ways. You can insert default or custom application keypads, or select an individual key for your transformations.

### Default Keypad Web-only

Select **Insert Application Keypad > Default Keypad** to add a default application keypad into a template. To configure default application keypad and define which keys to display, go to the **HATS Projects** view, double click the **Project Settings** of your HATS project, select the **Rendering** tab and click **Application Keypad**.

### Custom Keypad Web-only

Select **Insert Application Keypad > Custom Keypad** to add a custom application keypad into a template. Custom application keypads use the same keys specified in the default application keypad settings, located in the **Project Settings** of your HATS project. With a custom application keypad, you can edit the attributes of individual keypad buttons or links in the **Design** view by highlighting the button or link, right-clicking, and selecting **Properties**.

### Individual Key

Use this option to add an individual application key to your transformation. From the **Insert Application Key** panel, select which application key to insert and whether you want it displayed as a button or link, then click **OK**. For Web projects you can select multiple application keys within the same instance of the wizard. For rich client projects you can select only one application key for each instance of the wizard. After inserting, you can edit the button or link using the **Properties** tab.

## Insert All Host Components Web-only

Select **Insert All Host Components** when you want to add all possible host components from a particular screen capture or customization to a transformation. Choose the appropriate project screen from the **Select Screen** drop-down menu. You can highlight certain fields on your host screen by selecting the different options beside **Highlight fields**. If you want to see where the input fields are defined on the screen, select the **Input** check box. If you want to see what fields are protected, select the **Protected** check box. If you want to highlight any hidden fields, select the **Hidden** check box. To modify the colors of the input, protected or hidden fields highlighting, see "Using HATS preferences" on page 126. Click **OK** after making your selections.

The host components added will use the default rendering set specified within **Rendering Sets** in the **Rendering** tab of your HATS **Project Settings**.

## Insert Stored Screen Web-only

Select **Insert Stored Screen** when you want to add a stored screen to your transformation. Select the stored screen name and then select **All fields (non-protected and protected)** or **Shared**. For example, stored screens can be combined in a transformation to create a screen consisting of all the screens' fields.

For information about combining multiple screens, see Chapter 14, "Combining screens," on page 351.

# Previewing transformations

As previously mentioned, you can preview a widget as it displays on a transformation in the Insert Host Component wizard or the Insert Tabbed Folder wizard.

Another way to preview your transformation is to use screen captures. For every transformation you create in your project, there is an associated screen capture. You can see the screen captures by expanding the **Screen Captures** folder in the **HATS Projects** view. Double-clicking on the name of the screen capture displays a view of the screen capture with two tabs, **Host Screen** and **Preview**. The **Host Screen** tab displays the screen as it appears on the host. The **Preview** tab displays how the transformation is rendered, along with the template associated with the transformation.

The transformation and template used to generate the preview are based on screen customizations defined in your project. **Preview** scans the list of enabled screen customizations. When a screen customization is encountered that matches the screen capture, the first action that applies a transformation (along with the associated template) is used to render the preview. If no matching screen customization is found, the default template and transformation are used for the preview.

**Notes:**

1. The **Preview** tab will not process the **Next Screen** defined in the screen recognition event. For more information, see “Next Screen” on page 167.
2. The **Preview** tab will not correctly display a transformation in a portlet project. Limitations include not displaying images and CSS styles correctly. Depending on the content of the transformation, JavaScript errors may also occur.

## Host keypad

The host keypad includes buttons that represent host keys, such as F1, F2, and Clear, which are used to control functions on the host screen.

By default, HATS does not display the host keypad. To display the keypad, go to the **HATS Projects** view, double click the **Project Settings** of your HATS project, select the **Rendering** tab, and click on **Host Keypad**. You then have the option of selecting the check box to display the keypad as well as selecting which keys to display.

**Notes:**

1. Custom keypads and individual keys are defined using individual tags or composites, as a result they are displayed in the transformation's editor.
2. Text on the host screen that describes function keys are transformed into buttons or links using the default transformation. They are displayed in the transformed host screen area, separately from the host keypad.

Refer to Chapter 5, “Modifying a HATS project,” on page 87 for more information about the settings for keyboard support and keypads.

# Chapter 9. Component and widget descriptions and settings

HATS provides host components and widgets that are used to convert elements of a host screen to objects that can be displayed on a Web page. Some component and widget settings can be modified using the wizards and editors in the HATS Toolkit. This chapter describes HATS host components and widgets and the settings you can modify.

You can also create your own custom components and widgets by using the HATS wizard. To launch the wizard, select **HATS > New > Component** or **HATS > New > Widget** from the HATS menu. You can also launch the wizard by selecting **File > New > HATS Component** or **File > New > HATS Widget**.

For more information about using the wizard to create custom components and widgets, see the *HATS Web Application Programmer's Guide* or the *HATS Rich Client Platform Programmer's Guide*, depending on your application environment.

## Component and widget settings

The components and widgets supplied by HATS have default settings that you can modify, either for an entire project using the project editor, or for an individual transformation using the Insert Host Component wizard. Not all components and widgets have customizable settings.

**Note:** The vertical bar (|) character, which is valid in many component and widget settings, implies OR. For example, the function key component has a setting called start delimiter. You can specify PF|F for the start delimiter value. This means HATS will look for PFxxxx or Fxxxx on the host screen when looking for function keys.

## Host component settings

The settings for a host component specify how that component is to be recognized on the host screen.

Some host components have more complicated settings. For example, several settings are used to recognize a function key or a selection list. These settings will be described under each host component.

HATS provides the host components that are described in the following sections.

### Command line

The Command line component is responsible for recognizing input fields which have a specific string of characters (the token) preceding the input field.

A recognized command line can be rendered by the following widgets:

- Combo **RCP-only**
- Drop-down (data entry)
- List

- Popup
- Radio button (data entry)
- Text input
- Combo box (Dojo) `Web-only`
- Filtering select (Dojo) `Web-only`
- Text box (Dojo) `Web-only`

The following figure shows how a Command line component might appear on your host screen:


*Figure 12. Command line component example*

1. Field caption (extracted)

The following settings can be configured for this component:

**Token** The string of characters which must precede the input field.

**Extract field caption**
Select this check box if you want available captions extracted for the selected input fields.

**Trim spaces on caption**
See "Input field" on page 196 for more information.

**Clip to selected region**
See "Input field" on page 196 for more information.

---

## Dialog

The Dialog component is used to aid default rendering in recognizing modal (pop-up) frames on the host screen.

The concept of the Dialog component is to find the smallest area around the cursor that fits the description of the border in a rectangular form, assuming you are working with modal dialogs. A modal dialog retains the input focus while open. The user must close or otherwise satisfy the dialog before switching to another window.

To input data and complete a modal dialog, HATS assumes that the cursor must be within the dialog's border. Starting from the cursor character's coordinates on the screen, HATS searches to the left for a valid vertical border character as defined by the **Vertical border characters** setting. If a valid left vertical border character is located, HATS returns to the same cursor coordinate and searches to the right for a valid vertical border character. If valid vertical border characters are found on the left and right of the cursor coordinates, HATS measures the distance between them to ensure that a valid length exists.

HATS also searches for the top and the bottom horizontal border characters, as defined by the **Horizontal border characters** setting, using the same process as the search for the vertical border characters.

Because a dialog window is a rectangle, HATS uses the columns of the left and right borders and the rows of the top and bottom borders to determine whether the columns meet the vertical criteria and the rows meet the horizontal criteria of a modal dialog. If the outer rows and columns of the rectangle align, assuming the **Minimum row count** attribute has been satisfied, the Dialog component declares the window as a match and the recognized area is rendered.

To configure this component, go to the **Project Settings** of your HATS project, select the **Rendering** tab and click **Default Rendering**.

**Note:** The dialog component is only available in default rendering.

The dialog component is rendered using the Dialog widget.

The following figure shows how a Dialog component might appear on your host screen:

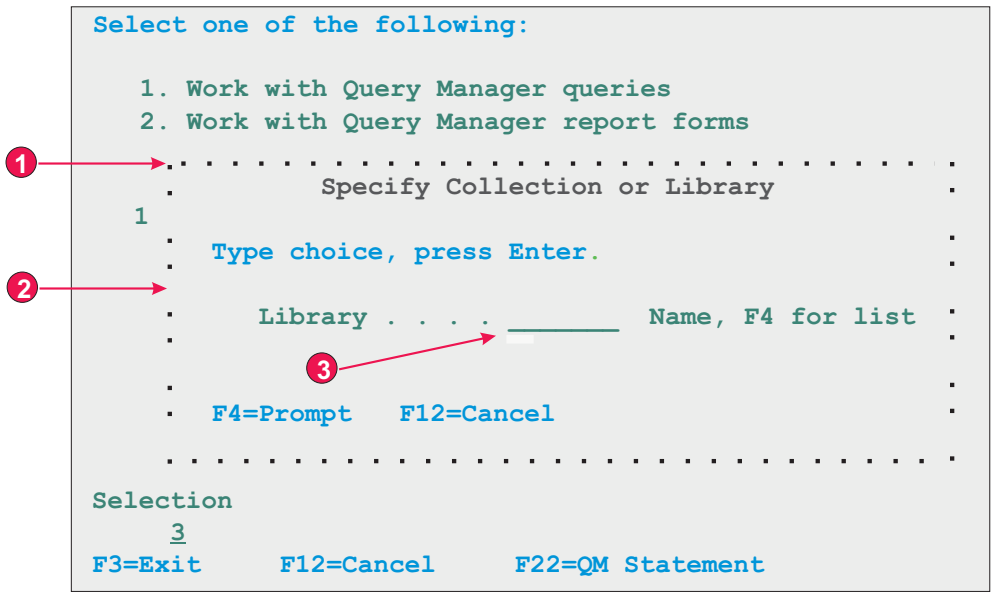



*Figure 13. Dialog component example*

1. Horizontal border characters
2. Vertical border characters
3. Cursor must be positioned within dialog

The following settings can be configured for this component:

**Horizontal border characters**
Specifies the characters that make up the top and bottom borders of the dialog. By default, horizontal border characters are any protected reverse-video character, a period (.), a hyphen (-), or an underscore (_). Multiple characters are separated by a vertical bar (|) and are evaluated left to right.

**Vertical border characters**
Specifies the characters that make up the side borders of the dialog. By default, vertical border characters are any protected reverse-video

character, a colon (:), or a vertical bar (|). Multiple characters are separated by a vertical bar (|) and are evaluated left to right.

**Note:** Because the vertical bar (|) is used as the separator when specifying multiple characters, use &vl. to specify a vertical bar (|) as a vertical border character.

In addition to characters, field attributes for border characters can also be specified in the **Horizontal border characters** and **Vertical border characters** settings for this component. The following indicators are used when specifying the attributes of a field. Each field specification is enclosed within a set of curly braces ( {...} ). Each indicator is separated by a space. To indicate a negative (NOT), add an exclamation point (!) as a prefix to the indicator, with no space between them.

- Protected field: `p`
- Hidden field: `h`
- Numeric only field: `no`
- High intensity field: `hi`
- Foreground color of a field, post-indicated by hex character, example `f_7`
- Reversed video field: `rv`
- Underlined field: `ul`
- Blinking field: `b`
- Column separator field: `c`

Example: `{!p hi no}` specifies an unprotected highlighted numeric only field.

**Notes:**

1. When specifying field attributes for border characters, the specification is intended to be a modifier to the data directly preceding it, if any. For example:
   - `{p rv}` matches any protected, reverse video character.
   - `:{p rv}` matches only a single, protected, reverse video colon character.
   - `{p rv}:` matches any protected, reverse video character because the colon is not considered since it follows the field specification.
2. The ordering of the indicators within the field attribute definition does not matter, because all indicators within a pair of curly braces must match for a border character to be recognized.

**Minimum row count**
The minimum number of rows the region must contain for it to be recognized as a dialog. The default is 3.

**Minimum column count**
The minimum number of columns the region must contain for it to be recognized as a dialog. The default is 3.

**Cursor must be positioned within dialog**
Specifies whether the cursor has to be within the region recognized as a dialog. If not selected, the region recognized as a dialog does not have to contain the cursor, and the user may get unexpected results if the dialogs are modal.

**Allow interaction outside dialog**
Specifies whether the area outside of the border should be disabled.

**Note:** Text replacement for content within the borders of a dialog cannot be defined using the Dialog component settings. The Dialog component itself is

only the border of the dialog window (the box around it). The content within the borders is rendered by other components of the default rendering. To perform text replacement for content within a dialog, you can specify text replacement for the default rendering item that renders the dialog content. For example, text replacement can be specified for the Field component. To get an accurate view of the combination of Dialog and other components, click **Full page preview**.

---

# ENPTUI

Enhanced Non-Programmable Terminal User Interface (ENPTUI) enables an enhanced interface on non-programmable terminals (NPT) and programmable work stations (PWS) over the 5250 full screen menu-driven interface, taking advantage of 5250 display data stream extensions. HATS supports a subset of the ENPTUI feature which includes graphical interface constructs. These constructs are represented by two HATS components: Scrollbar and Selection Field.

You can add ENPTUI support during the creation of a new HATS project from the HATS toolbar by selecting **HATS > New > Project** (or **File > New > HATS Project**) to launch the Create a Project wizard. When you get to the Connection Settings panel and select either **5250** or **5250W** for your connection type, select the **Add graphical interface DDS keywords (ENPTUI) rendering support** check box to enable the support.

**Note:** This option is not available in projects that are optimized for mobile devices.

If the check box is selected, the default rendering set in the application.hap file will include standard ENPTUI rendering items and the component and widget registry (ComponentWidget.xml) will include the ENPTUI components and widgets for inclusion in either default rendering or custom transformations. Also, the default connection object created will have ENPTUI=true added to the advanced connection properties, which is required for HATS to receive the ENPTUI data from the host.

If the check box remains cleared, the Create a Project wizard will edit the component and widget registry and the application.hap file and remove ENPTUI rendering items, components, and widgets from the files laid down by the wizard in the new project.

**Note:** This is the only chance to have HATS add ENPTUI components and widgets to the project. If you want to add ENPTUI to your project after creation, all edits to ComponentWidget.xml, application.hap, and main.hco must be done manually (by comparing to find the changes required). This is not recommended. Creating a new project is the recommended method to add ENPTUI support.

Following are the ENPTUI components and their settings. These components do not appear in the HATS Components section in the **Palette** view of the Rich Page Editor. If the ENPTUI support has been enabled as described above, these components are accessible from the Rational SDP menu bar by clicking **HATS Tools > Insert Host Component**.

## Scrollbar field (ENPTUI)

The Scrollbar field (ENPTUI) component is responsible for recognizing ENPTUI scrollbar fields from the 5250 data stream. This component will recognize any

selected host screen region and this component can be rendered using the Scrollbar (ENPTUI) widget. There are no customizable settings for this component.

## Selection field (ENPTUI)

A recognized ENPTUI selection field can be rendered by the following widgets:

- Button
- Check box
- Link
- Radio button (selection)

The following settings can be configured for this component:

**Selection Field Type Recognition**
The selection field types that you want to match for any given instance of the component

**Push Buttons**
Select this box if you want to recognize push button ENPTUI selection fields

**Single Choice Fields**
Select this box if you want to recognize single choice ENPTUI selection field

**Multiple Choice Fields**
Select this box if you want to recognize multiple choice ENPTUI selection fields

**Menu Bars**
Select this box if you want to recognize menu bar ENPTUI selection fields

**Selection Field Attribute Recognition**
The selection field attribute you want to match for any given instance of the component

**Recognize fields having auto-enter**
Select from the drop down list whether you want to recognize ENPTUI selection fields with auto-enter enabled, disabled or both

The following figure shows how the ENPTUI components might appear on your host screen:

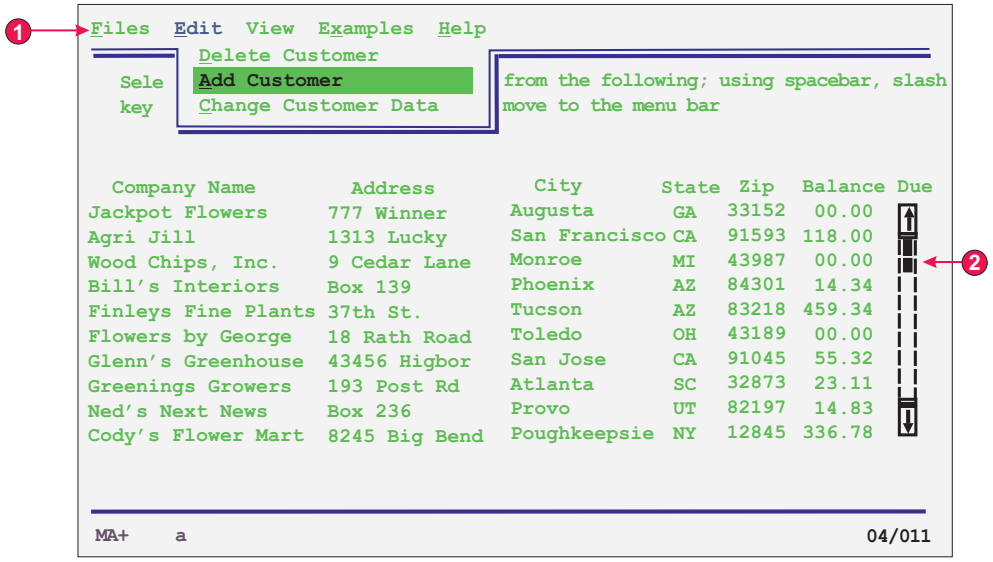



*Figure 14. ENPTUI components example*

1. Selection field component
2. Scrollbar field component

---

## Field

The Field component is responsible for recognizing protected text and input fields. This component will recognize any selected host screen region and this component can be rendered using the Field widget.

**Note:** The Field component includes only the region you specify. If the selected region cuts through an actual host screen field, only the part inside the selection will be rendered.

The following figure shows how a field component might appear on your host screen:


*Figure 15. Field component example*

The following setting can be configured for this component:

**Support character level attributes for protected fields**

This setting allows you to accurately render the color and extended attributes of all characters in a protected field, instead of rendering the entire field with the color and attributes of the first character in the field.

If not selected, which is the default, the first character of the field will determine the attributes for every character in the field even if an attribute, such as color, changes in the middle of the field.

If selected, different color and extended attribute (for example reverse video) information is supported within a field by rendering Field component elements as segments of sequential, similar characters. For

example, a field that is ten characters in length, where the first five characters are green and the last five characters are red, will be represented by two Field component elements.

**Note:** This setting only applies to protected fields, not input fields.

---

## Function key

The Function key component is responsible for recognizing function (PF) keys. This component is used to transform static text on the host screen representing a function key into a button or link which your user can click. Because function keys appear differently on different host systems, HATS gives you flexibility in configuring the different parts. For example, a host screen might contain the following function keys:

**F3: Exit** **F4: Back** **F5: Fwd**

A function key can be rendered by the following widgets:

- Button
- Button table
- Drop-down (selection)
- Link
- Radio button (selection)
- "Toolbar `RCP-only` " on page 294

The following figure shows how a Function key component might appear on your host screen:

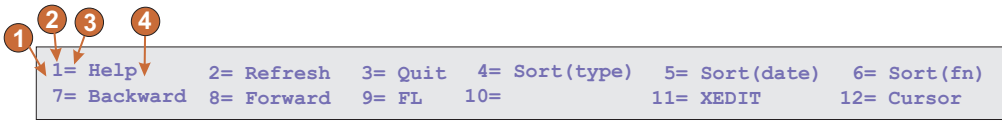



*Figure 16. Function key component example*

1. String before leading token (a space)
2. Start delimiter (#, signifying a number)
3. Delimiter (an equal sign (=))
4. String after description (a space)

The following settings can be configured for this component:

**String before the leading token**
Optional. The string that must precede the start delimiter. You can specify more than one value, separated by the vertical bar (|) character. For a function key candidate to be recognized, it must be preceded by this value.

**Start delimiter**
Required. The string that marks the beginning of the potential function key. You can specify more than one value, separated by the vertical bar (|) character. Any of the values will be recognized as beginning a function key. This value can contains a "#" (pound sign) – which will be evaluated as "a number" during recognition.

**Delimiter**

Optional. The string that separates the start delimiter from the description. You can specify more than one value, separated by the vertical bar (|) character.

**String after description**

Optional. The string that immediately follows the description. The default is " " (two spaces) to allow for detection of multiple word descriptions for function keys.

When using Function key component settings, consider the following example. Suppose a host screen has the following function key descriptions on it:

```
F1=Help F2=Open F3=Exit
```

where there is only one space between each description. HATS will recognize this as three individual function keys even though the default value for the **String after description** setting is " " (two spaces). This is because the Function key component searches for a **Start delimiter**, a **Delimiter**, and then the **String after description**. But, if it finds a new **Start delimiter** before finding the **String after description**, then the start of a new function key description is assumed.

**Notes:**

1. The HATS Function key component only recognizes and renders F# keys, PA keys, Enter, and Clear. To disable recognition of Enter and Clear, remove the last # from the start delimiter definition.
2. The HATS Function key component does not recognize other host function keys such as Help, PageUp (also called rollup), PageDown (roll down), System Request, Attention, and Print. These other functions are available on the host keypad. You can also add these host keys to a HATS transformation using the **HATS Tools** menu item.
3. If you use the Drop-down (selection) widget to render the Function key component in default rendering, multiple drop-down widgets may be rendered in the preview and during runtime. To correct this during runtime, in the source for the application.hap file, locate the `<renderingItem ...>` tag for the FunctionKeyComponent with the SLDropdownWidget. Then change the value for the setting, `keepOutputTogether`, from `false` to `true`. The preview will continue to display multiple drop-down widgets.
4. If you use the function key components in default rendering and want them to appear on different lines in your transformation, you must use the <BR> tag or map them to a table to force the move. This can be done by viewing and editing your transformation under the **Source** tab.

---

## HTML DDS keyword Web-only

Using the HTML DDS keyword in IBM i display files allows the sending of HTML fragments along with the 5250 data stream. This feature is not intended for display with standard emulation programs; it is only sent to 5250 Workstation Gateway devices, and host access products such as HATS, that have been enhanced to show this HTML data in a browser

HTML DDS is available as a component when the host type is specified as 5250. The use of this component is not part of the default rendering and must be added to either the default rendering or custom transformations, if needed.

The HTML DDS Keyword component is responsible for recognizing this HTML data if it is present in the 5250 data stream

This component can be rendered using the Label widget.

The following settings can be configured for this component:

**Ignore DDS data using filter**
Click this button if you want to recognize the DDS data as it was sent in the 5250 data stream, or as it was sent except for those tags which are specified by these "ignore" filters.

**Ignore these tags**
Enter any tags (without the < and > characters) that you want to ignore. The data between the tag and its end tag is kept. For example, specify a b if you wanted to keep some data surrounded by the bold tags, but not the tag itself.

**Ignore all DDS data in these tags**
Enter any tags (without the < and > characters) for which you want to ignore the data between the tag and its end tag.

**Accept DDS data using filter**
Click this button if you want to recognize, from the HTML DDS data in the 5250 data stream, only the HTML tags which pass these "accept" filters. HTML DDS data which doesn't pass these filters is ignored.

**Accept these tags**
Enter any tags (without the < and > characters) that you want to keep. The data between the tag and its end tag is ignored

**Accept all DDS data in these tags**
Enter any tags (without the < and > characters) for which you want to keep the data between the tag and its end tag.

You can include a keyword, HTML, which will allow you to embed raw HTML data into the data stream for interpretation only by browser-like recipients. This support in HATS will enable data that is sent from a host application in the HTML DDS keyword to be used in either default rendering or custom transformation.

To enable HTML DDS keyword support for a connection definition, take the following steps:

1. Open the **Connection Editor** by clicking your main HATS connection in the **HATS Projects** view.
2. Select the **Advanced** tab and click the **Add** button to configure an advanced connection setting.
3. Select **HTMLDDS** for **Name** from the drop-down list and type in **true** in the **Value** field.
4. Click **OK**.

---

## Input field

The Input field component is for recognizing input fields (non-protected fields). This component, besides recognizing input fields, can also extract the field's accompanying caption. To find the field's caption, this component looks for

protected text directly preceding the input field (on the same row) first. If the algorithm cannot find a caption, it will look directly above the field for protected text.

A recognized input field can be rendered by the following widgets:

- Calendar **Web-only**
- Check box
- Combo **RCP-only**
- Drop-down (data entry)
- List
- Popup
- Radio button (data entry)
- Text input
- Combo box (Dojo) **Web-only**
- Date text box (Dojo) **Web-only**
- Filtering select (Dojo) **Web-only**
- Text box (Dojo) **Web-only**
- Validation text box (Dojo) **Web-only**

The following figure shows how an input field component might appear on your host screen:

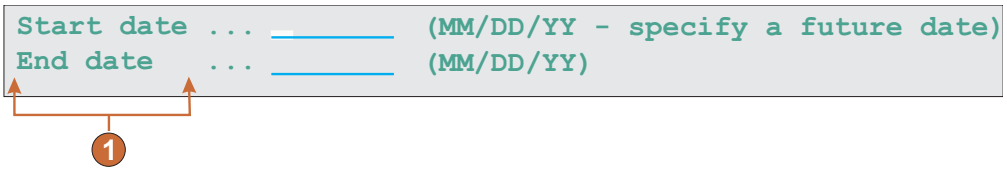



*Figure 17. Input field component example*

1. Field caption (extracted from component)

The following settings can be configured for this component:

**Extract field caption**
If selected, a caption for the recognized input field is extracted (see the description above for more information about this algorithm). This extracted caption can be used by the widget.

**Restrict caption to selected region**
If selected, the caption is limited to the text within the selected part of the screen.

**Strip end of caption**
If selected, the end of the caption is stripped after (and including) the first occurrence of any value specified in the **Strip after** setting. This setting is useful for cleaning up extracted captions.

**Strip after**
Required. The string (or set of strings) that is used to strip the extracted caption. For example, if the extracted caption is **Command ==>** and the value for this setting is **=**, everything after (and including) the first **=** is stripped off.

**Replace with**
Optional. The string to replace the stripped off portion (if applicable) of the caption with.

**Trim spaces on caption**
If selected, white space (for example spaces and tabs) is removed from both ends of the extracted (and stripped) caption.

**Clip input field to selected region**
If selected, only the part of the input field inside of the selected region is recognized and rendered. If cleared, if any part of an input field is inside the selected region, the entire input field is recognized and rendered. This setting is useful if you need to break a large host screen input field into multiple, smaller Web page input fields.

---

## Input field with hints

The Input field with hints component is for recognizing input fields (non-protected fields) which have accompanying hints (text immediately following the input field on the same row). This component is functionally similar to the Input field component, except that hints must be found next to the candidate input. For example, suppose your host screen had the following input field:

**Product code: [ ] (Valid codes: A, B, C, D)**

In this example, the start of the hints is **Valid codes:**, the end of the hints is a close parenthesis **[)]**, the separator is a comma **(,)**, and the leading token type is **None**.

A recognized input field can be rendered by the following widgets:

- Combo `RCP-only`
- Drop-down (data entry)
- List
- Popup
- Radio button (data entry)
- Combo box (Dojo) `Web-only`
- Filtering select (Dojo) `Web-only`

The input field also uses valid value hints. By default, the input field does not recognize hints, therefore you will have to modify the component settings in order to pass these along to the rendering widget.

The following figure shows how an input field with hints component might appear on your host screen:

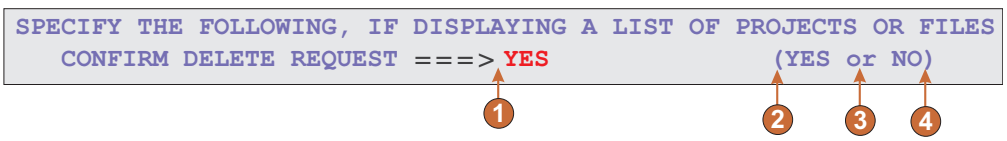



*Figure 18. Input field with hints component example*

1. Delimiter: position of delimiter, but leading token type was set to **None**.
2. Start of hints: open parenthesis (
3. Separator: or

4. End of hints: close parenthesis )

Leading token type is not marked. In this example, the leading token type is **None**. No letter or digit precedes the hints.

The following settings can be configured for this component:

**Extract field caption**
If selected, a caption for the recognized input field is extracted (see the description above for more information about this algorithm). This extracted caption can be used by the widget.

**Restrict caption to selected region**
If selected, the caption is limited to the text within the selected part of the screen.

**Strip end of caption**
If selected, the end of the caption is stripped after (and including) the first occurrence of any value specified in the **Strip after** setting. This setting is useful for cleaning up extracted captions.

**Strip after**
Required. The string (or set of strings) that is used to strip the extracted caption. For example, if the extracted caption is **Command ==>** and the value for this setting is **=**, everything after (and including) the first **=** is stripped off.

**Replace with**
Optional. The string with which to replace the stripped off portion (if applicable) of the caption.

**Trim spaces on caption**
If selected, white space (for example spaces and tabs) is removed from both ends of the extracted (and stripped) caption.

**Clip input field to selected region**
If selected, only the part of the input field inside of the selected region is recognized and rendered. If cleared, if any part of an input field is inside the selected region, the entire input field is recognized and rendered. This setting is useful if you need to break a large host screen input field into multiple, smaller Web page input fields.

**Start of hints**
Optional. The string of characters which identifies where the set of hints starts. Multiple values can be specified for this setting; separate with the vertical bar (|) character.

**End of hints**
Optional. The string of characters which identifies where the set of hint ends. Multiple values can be specified for this setting; separate with the vertical bar (|) character.

**Separator**
Required. The string of characters which separates each hint in the hint set. Multiple values can be specified for this setting; separate with the vertical bar (|) character.

**Leading token type**
Specifies what each hint is. For example, if the hint set was **A=Apple,O=Orange,G=Grape**, you would set this value to **Letter** (or **Letter or Digit**) because A, O, and G are all letters. You would set the delimiter to **=**.

**Maximum length of leading token**
Specifies the maximum length of the leading token. The default is **4**.

**Delimiter**
Required. The string of characters which separates the leading token from the hint description. This setting only applies when the leading token type is not **None**.

**Minimum required hints**
The minimum number of hints that must be found for this input field to be recognized by this component. This setting is useful for avoiding errant recognition.

---

## Item selection

The item selection component is responsible for recognizing screens with list of items, where the interaction is to select a character next to the item.

The item selection component can be rendered by the following widgets:

- Check box
- Combo `RCP-only`
- Drop-down (data entry)
- Link (item selection)
- List
- Popup
- Radio button (item selection)
- Text input
- Combo box (Dojo) `Web-only`
- Filtering select (Dojo) `Web-only`

The following figure shows how an item selection component might appear on your host screen:


*Figure 19. Item selection component example*

1. Input field
2. Caption

The following settings can be configured for this component:

**Extract field caption**
Select this box if you want available captions extracted for the selected input fields.

**Strip end of caption**
Enabled only if **Extract field caption** is selected. Select this box to delete the ending text of a caption. Specify where to begin the deletion in **Strip after**, and any characters with which to replace the deleted caption text in **Replace with**.

**Trim spaces on caption**
Enabled only if **Extract field caption** is selected. Select this box to remove any extra spaces on either end of the captions.

**Captions are after input fields**
Enabled only if **Extract field caption** is selected. Select this box if the captions to be extracted are on the right of the input fields. Clear this box if captions are on the left of the input fields.

**Input fields must have a caption**
Select this box of you only want input fields with captions to be recognized on the host screen.

**Clip input field to selected region**
Select this box if you want to render only the selected part of an input field. Clear this box if you want to render the entire field if any part of the field is selected.

---

## Light pen (attention)

The light pen (attention) component is responsible for recognizing a light pen (also known as selector pen) attention field on 3270 host screens.

**Note:** This component does not appear in the HATS Components section in the **Palette** view of the Rich Page Editor. If light pen support was enabled by selecting the **Add light pen rendering support** check box during the initial creation of a 3270 or 3270E project, this component is accessible from the Rational SDP menu bar by clicking **HATS Tools > Insert Host Component**.

A recognized light pen attention field can be rendered by the following widgets:

- Button
- Link
- Radio button (selection)

Depending on the designator character (the first character of the field), clicking the rendering widget (button, link, or radio button) causes either an [enter] or a [cursel] attention identifer (AID) to be sent to the host. If the designator is an ampersand character (&), an [enter] AID is sent. If the designator is a blank character, a [cursel] AID is sent.

The following figure shows how a light pen (attention) component might appear on your host screen:


*Figure 20. Light pen (attention) component example*

1. Designator character (the first character of the field)

The following settings can be configured for this component:

**Consume remainder of field**
Specifies whether the entire field or just the first character of the field is consumed and rendered by the widget. Select this check box to specify that the entire field is consumed and rendered. For example, the contents of the field are rendered as the caption of the rendering widget (button, link, or radio button). Clear this check box to specify that only the first character will be rendered by the widget. This is useful to preserve colors or extended attributes in the remainder of the field. When in Default Rendering, clearing this check box will cause the remainder of the field (everything except for the first character of the field) to be rendered by some other widget (in most cases by the Field widget). The default is **selected**.

---

## Light pen (selection)

The light pen (selection) component is responsible for recognizing a light pen (also known as selector pen) selection field on 3270 host screens.

**Note:** This component does not appear in the HATS Components section in the **Palette** view of the Rich Page Editor. If light pen support was enabled by selecting the **Add light pen rendering support** check box during the initial creation of a 3270 or 3270E project, this component is accessible from the Rational SDP menu bar by clicking **HATS Tools > Insert Host Component**.

A recognized light pen selection field can be rendered by the Check box widget.

If the designator character (the first character of the field) is a question mark (?), the rendered check box is not selected. If the designator character is a greater-than sign (>), the rendered check box is selected.

The following figure shows how a light pen (selection) component might appear on your host screen:


*Figure 21. Light pen (selection) component example*

1. Designator character (the first character of the field)

The following settings can be configured for this component:

**Consume remainder of field**
Specifies whether the entire field or just the first character of the field is consumed and rendered by the widget. Select this check box to specify that the entire field is consumed and rendered. For example, the contents of the field are rendered as the caption of the rendering widget (check box). Clear this check box to specify that only the first character will be rendered by the widget. This is useful to preserve colors or extended attributes in the remainder of the field. When in Default Rendering, clearing this check box will cause the remainder of the field (everything except for the first character of the field) to be rendered by some other widget (in most cases by the Field widget). The default is **selected**.

---

## Selection list

The Selection List component is responsible for recognizing selection lists. Selection lists are like menus, but they are presented as a list of options, each of which is preceded by a leading token and a delimiter, such as in the following example:

**Option 1. Prepare form**
**Option 2. Work with forms you submitted**
**Option 3. Work with forms requiring action**

In this example, **Option** is the string before the leading token, **1**, **2**, and **3** are the leading tokens, and the delimiter is a period followed by a space **(. )**.

Once a list item has been selected (either clicked or selected depending on the widget), its leading token is placed in the targeted host input field.

A recognized selection list can be rendered by the following widgets:

- Button
- Button table
- Drop-down (selection)
- Link
- Radio button (selection)

- Toolbar **RCP-only**
- Combo box (Dojo) **Web-only**
- Filtering select (Dojo) **Web-only**

The following figure shows how a selection list component might appear on your host screen:

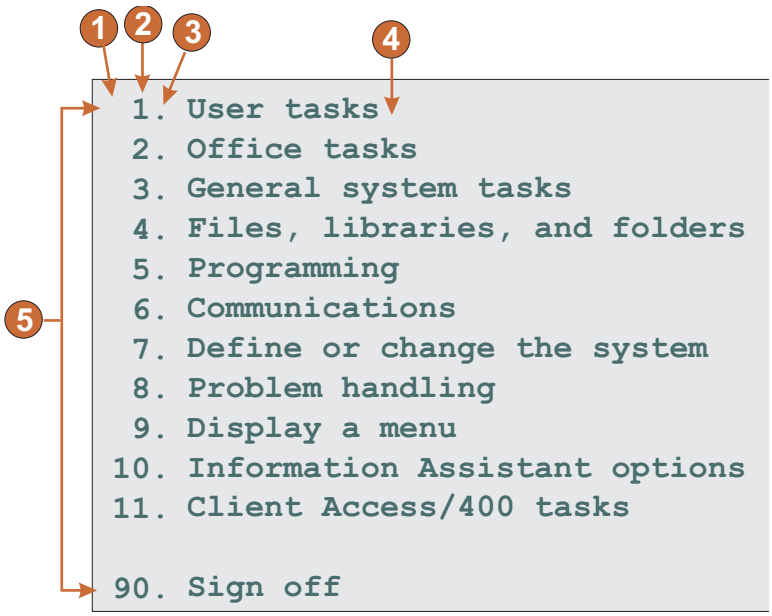



*Figure 22. Selection list component example*

1. String before the leading token (a space)
2. Leading token type (digit)
3. Delimiter (a period (.))
4. End delimiter (a space)
5. List options (or items)

The following settings can be configured for this component:

**String before the leading token**
Optional. The string which precedes the leading token of each list item.

**Leading token type**
Specifies whether each list item starts with letters, digits, or both letters and digits.

**Maximum length of leading token**
Specifies the maximum length of the leading token. The default is 4.

**Delimiter**
Required. The string of characters between the leading token and the description. You can specify more than one value, separated by the vertical bar (|) character.

**End delimiter**
Optional. The string of characters which must follow each list item.

**Minimum required options**
The minimum number of list items that must be found in order for the selected region to be recognized as a selection list.

**Group Items Individually**
Select this box if you want to group items individually.

**Selection target field**
Specifies which input field receives the selection. Options: **First on screen**, **Last on screen**, **Previous field**, **Next field**, **Cursor position**, or **User defined**.

**Row** Specifies the row of the input field which should receive the selection.

**Note:** This setting only applies if **User defined** is selected as the selection target.

**Column**
Specifies the column of the input field which should receive the selection.

**Note:** This setting only applies if **User defined** is selected as the selection target.

**Auto submit on select**
If selected, an AID key is automatically sent to the host even if the input field is blank. If cleared, an AID key is not automatically sent.

**Action key**
Optional. Specifies the host aid key that should be pressed after the target input field has been updated with the user's selection. The default value is **[enter]** meaning that the enter key will be pressed once the user makes a selection.

**Sort list options**
If selected, list items are sorted according to either their leading token or description (see below for more information about other settings).

**Sort by**
Specifies whether to sort the set of list options by their leading token or description.

**Sort order**
Specifies whether to sort the set of list options in ascending or descending order.

**Note:** The locale-sensitive sorting methods, built into Java, are used to perform this operation.

**Placeholder options**
Specifies whether to embed placeholder list items when gaps (such as empty space) exist between list items. Options: **Actual**, **None**, or **Single**.

**Note:** For information about using this component with DBCS support see "Selection list" on page 462.

## Subfile

The Subfile component is responsible for recognizing 5250 subfiles.

A recognized subfile can be rendered by the following widgets:

- Subfile (check box)
- Subfile (drop-down)
- Subfile (popup)

The following figure shows how a subfile component might appear on your host screen:

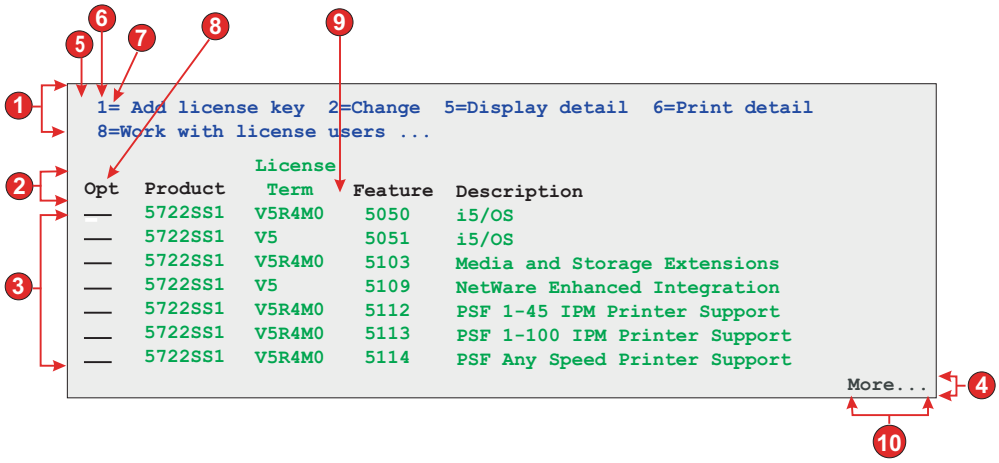



*Figure 23. Subfile component example*

1. Action region
2. Header region
3. Data region
4. Marker region

**Action settings**

5. String before the leading token (a space)
6. Leading token type (digit)
7. Delimiter (an equal sign (=))

**Header setting**

8. Beginning text for header (Opt)

**Data setting**

9. Column delimiter (a space)

**Marker setting**

10. Marker text (More...)

To launch Subfile Settings, follow these steps:

1. Go to the **HATS Projects** view and click on your **Project Settings**.
2. Select the **Rendering** tab.
3. In the **Rendering** tab, expand the **Components** tree and select the **Subfile** component.

4. Click on the **Settings** button to the right to display the Subfile Settings window.

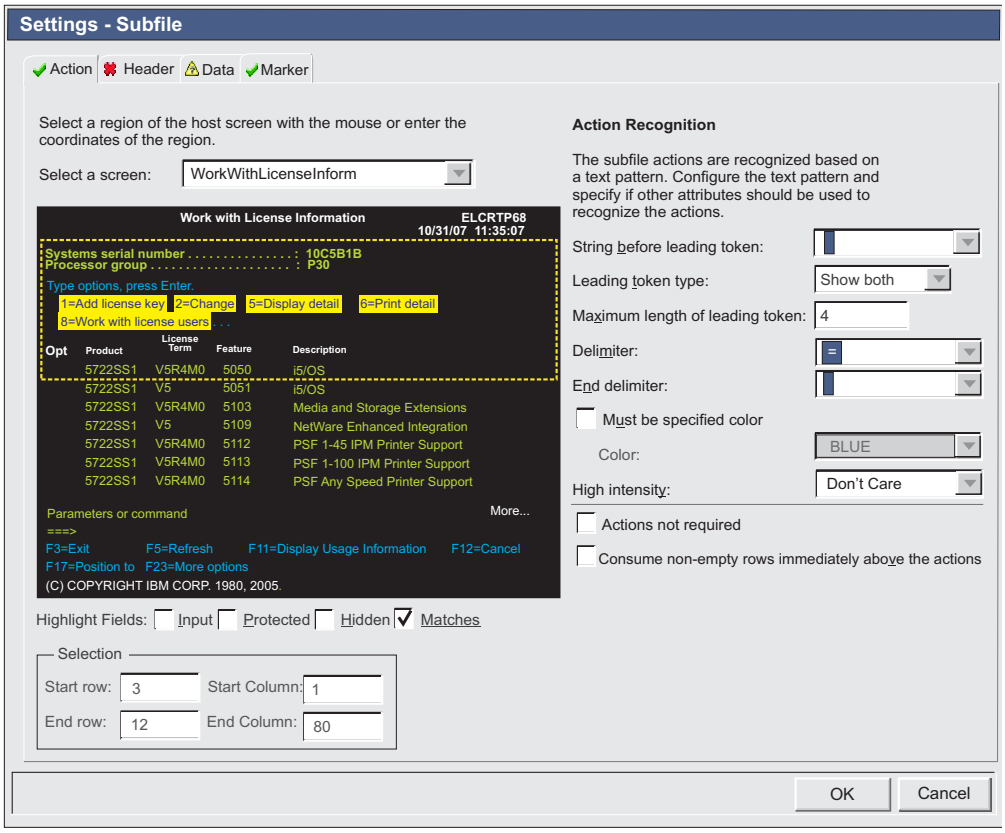



*Figure 24. Subfile settings example*

The Subfile component settings consist of four tabs, Action, Header, Data, and Marker. Each tab is used to define the criteria for recognizing a particular element of the subfile. Icons on each tab indicate the following states:

- Recognized: This state means the element of the subfile is recognized on the selected screen. If the **Matches** option is selected, the recognized element is highlighted. To change the highlighting color, use the **Pattern matches** setting in the **Screen capture highlighting colors** section of the HATS preferences. For more information, see "Using HATS preferences" on page 126.
- Unrecognized: This state means the element of the subfile is not recognized. You must adjust the screen region or recognition criteria to tune the settings until recognized.
- Unknown: This state indicates that the recognition of this element depends on another element that is not yet recognized. Tune the unrecognized settings on the other tabs first, then come back to this tab.

When all four tabs display recognized indicators, click **OK**. You can see the recognized result in the widget preview.

Each recognition tab is described below.

## Action

Select the screen you want to use from the drop-down list. You can also specify the region of the **Selection** area on the host screen by either using your mouse, or by entering the coordinates of the region in the following boxes:

**Start row**
The first row to look for the subfile action. The default is **3**.

**End row**
The last row to look for the subfile action. The default is **12**.

**Start column**
The first column to look for the subfile action. The default is **1**.

**End column**
The last column to look for the subfile action. The default is **80**.

You can highlight certain fields on your host screen by selecting the different options beside **Highlight fields**. To see where input fields are defined on the screen, select **Input**. To see what fields are protected, select **Protected**. To highlight hidden fields, select **Hidden**. To see matching patterns, select **Matches**. To modify the highlighting colors of the input, protected, hidden, or pattern matching fields, see "Using HATS preferences" on page 126.

**Action Recognition**
Additional action recognition criteria can be configured by selecting the following individual settings.

**String before leading token**
String. This value specifies the string before the leading token. The default value is a space.

**Leading token type**
String. This value is the type of token this host application uses for the host command. In the example where **1=Edit**, the **1** token is the host command, and its type is a digit. In the example where **A=Edit**, the **A** token is the host command, and its type is a letter. The allowable values are **Digit**, **Letter**, **Show both**. The default is **Show both**.

**Maximum length of leading token**
The maximum length of the leading token. The default is **4**.

**Delimiter**
String. This value is the token used in the subfile action to delimit the host command from the description of the host command. In the example where **1=Edit**, **1** is the host command, the **=** token is the delimiter, and **Edit** is the description of the host command. The allowable value is one or more strings concatenated by the vertical bar (|). The default is **=**. Example: `=| = |-| - |:`.

**End delimiter**
String. This is used to recognize the end of a subfile action item description. Separate multiple character with a vertical bar (|).

**Must be specified color**
This box is used in conjunction with previous settings to designate that the subfile action's host command must be of a specific type and the subfile action must be a specific color. For example, the subfile action must be a digit, and must be blue.

**Color** See description for the previous setting.

**High intensity**
String. This value specifies whether the subfile action uses the high intensity field attribute. The allowable values are **Yes**, **No**, **Don't care**. The default is **Don't care**.

**Actions not required**
Select this box if you want the subfile to be recognized even if it does not have any actions. The subfile header, data, and marker are still recognized and rendered.

**Consume non-empty rows immediately above the actions**
Select this box if you want rows on the screen that contain text immediately above and adjacent to the actions, which meet the recognition criteria (color and/or intensity), consumed but not displayed by the subfile widget.

## Header

Specify the region of the **Selection** area on the host screen by either using your mouse, or by entering the coordinates of the region in the following boxes:

**Start row**
The first row to look for the subfile header. The default is **3**.

**End row**
The last row to look for the subfile header. The default is **16**.

**Start column**
The first column to look for the subfile header. The default is **1**.

**End column**
The last column to look for the subfile header. The default is **80**.

You can highlight certain fields on your host screen by selecting the different options beside **Highlight fields**. To see where input fields are defined on the screen, select **Input**. To see what fields are protected, select **Protected**. To highlight hidden fields, select **Hidden**. To see matching patterns, select **Matches**. To modify the highlighting colors of the input, protected, hidden, or pattern matching fields, see "Using HATS preferences" on page 126.

**Header Recognition**
Additional header recognition criteria can be configured by selecting the following individual settings.

**Recognize by text**
Boolean. This value specifies whether the subfile header should be recognized based on strings. The allowable values are **true** or **false**. The default is **true**.

**Begins with**
String. Strings that can be used to identify subfile headers; use **\n** to define multi-row headers. The allowable values should be concatenated with the vertical bar (|). More specific headers should be listed first. For example, if **Option**, **Opt** , and **Opt** are all possible headers, they should be ordered to ensure that the most specific header is listed first, **Option|Opt |Opt** The default is **true**. Example: **Description\nof Record|Option|Opt |?**.

**Must be specified color**
Boolean. This value is used in conjunction with the **Text** setting to designate that the subfile headers must have a specific string and must be a specific color. For example, the subfile header must contain the word **Opt.** and it must be **blue**. The allowable values are **true** or **false**. The default is **false**.

**Color** Specifies the color of the subfile header.

**High intensity**
Select this box to specify whether the Subfile Header can be recognized by text at the top of a table of data, which uses the high intensity field attribute.

**Specified color**
Select this box to specify if the subfile header can be recognized by text of a specific color at the top of a table of data.

**Color** Specifies the color of the subfile header.

## Data

Specify the region of the **Selection** area on the host screen by either using your mouse, or by entering the coordinates of the region in the following boxes:

**Start row**
The first row to look for the subfile data. The default is **4**.

**End row**
The last row to look for the subfile data. The default is **22**.

**Start column**
The first column to look for the subfile data. The default is **1**.

**End column**
The last column to look for the subfile data. The default is **80**.

You can highlight certain fields on your host screen by selecting the different options beside **Highlight fields**. To see where input fields are defined on the screen, select **Input**. To see what fields are protected, select **Protected**. To highlight hidden fields, select **Hidden**. To see matching patterns, select **Matches**. To modify the highlighting colors of the input, protected, hidden, or pattern matching fields, see "Using HATS preferences" on page 126.

**Data Recognition**
Additional data recognition criteria can be configured by selecting the following individual settings.

**Recognition Option:**
This setting specifies how the subfile data region is recognized.

**Field** `Web-only`
Recognize subfile data row by row.

**Note:** For rich client applications, if you want the see the subfile rendered row by row use the following subfile widget settings:

- Select **Show alternative row colors**
- Clear **Show grid lines**
- For **Column header source**, select **From component**
- Select **Use classic header style**
- Select **Override font** and select **Monospaced** font

**Table (field)** `Web-only`
Recognize subfile data by field information.

**Table (visual)**
Recognize subfile data that appears as a table with visible vertical column delimiters.

**Table** Recognize subfile data that appears as a table using user defined column breaks.

**Column delimiter:**
Enabled only if the **Recognition Option:** is set to **Table (visual)**. The string of characters which must exist in the same columns of every row in the selected region for the host screen column to be marked as a table column. If consecutive columns are each marked as a delimiting column, only the last column is actually rendered as the delimiting column of the generated table. The default is a space character.

**Column breaks:**
Enabled only if the **Recognition Option:** is set to **Table**. A comma-separated list of column numbers where column breaks are wanted. For example, specifying **1,7,22** would indicate that new columns should begin in columns 1, 7 and 22. When the **Restore Defaults** button is clicked, the **Column breaks** field is filled with values based on the **Table (visual)** default recognition values.

**Notes:**

1. Column breaks can also be added or removed by using the mouse to select a column in the host screen view, right-clicking, and selecting either **Add** or **Remove**. Column breaks are indicated by a dotted vertical line on the host screen view.
2. Column breaks cannot split an input field.
3. For information about using this setting with DBCS support see "Subfile" on page 462.

**Override action input field start column**
If selected, you can override the starting column of the action input field to prevent other input fields from being rendered incorrectly as drop-down lists.

**Column Start**
Start column of the action input field. The default value is **1**.

**Override action input field length**
If selected, you can override the length of the action input field to prevent other input fields from being rendered incorrectly as drop-down lists.

**Field Length**
Field length of the action input field. The default value is **1**.

**Data not required**
Select this box if you want the subfile to be recognized even if it does not have any data. If any subfile actions are detected, they are still rendered as drop-down lists.

**Include empty rows**
Select this box if you want rows that are empty in a table on the host screen to be included when the table is transformed. Clear the box if you want empty rows to be discarded.

## Marker

Specify the region of the **Selection** area on the host screen by either using your mouse, or by entering the coordinates of the region in the following boxes:

**Start row:**
The first row to look for the subfile marker. The default is **4**.

**End row:**
The last row to look for the subfile marker. The default is **22**.

**Start column**
The first column to look for the subfile marker. The default is **1**.

**End column**
The last column to look for the subfile marker. The default is **80**.

You can highlight certain fields on your host screen by selecting the different options beside **Highlight fields**. To see where input fields are defined on the screen, select **Input**. To see what fields are protected, select **Protected**. To highlight hidden fields, select **Hidden**. To see matching patterns, select **Matches**. To modify the highlighting colors of the input, protected, hidden, or pattern matching fields, see "Using HATS preferences" on page 126.

**Marker Recognition**
Additional marker recognition criteria can be configured by selecting the following individual settings.

**Recognize by text**
If selected, the end marker of the subfile can be recognized by the specified text string.

**Text** String. This can be used to identify the subfile end marker. Multiple values are allowed. Example: **More... | Bottom | End | +**. The default value is **More... | Bottom | +**.

**High intensity**
If selected, the end marker of the subfile can be recognized by a high intensity field below the data table.

**No visual text is required**
If selected, the high intensity field can be an empty string.

**Empty row**
If selected, the end marker of the subfile can be recognized by an empty row within the specified region below the data table.

---

## Table

The Table component is responsible for recognizing tables. It can be used instead of the Table (visual) component if you must indicate specific column numbers to use for column breaks in order to achieve the result you want.

Care should be taken when using the Table component in a default rendering set because the Table component will recognize almost every screen in the selected region, and no rendering items lower in the list will be recognized.

A recognized table can be rendered by the following widgets:

- Graph (horizontal bar, line, vertical bar)
- Table
- Enhanced grid (Dojo) **Web-only**

The following figure shows how a table component might appear on your host screen:

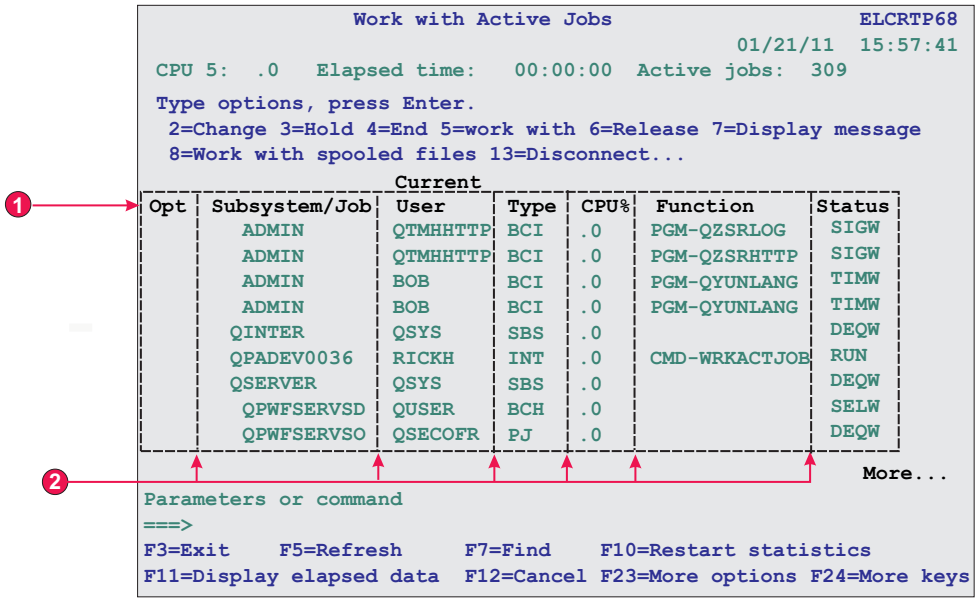



*Figure 25. Table component example*

1. Title rows
2. Column breaks

The following settings can be configured for this component:

**Column breaks:**
A comma-separated list of column numbers where column breaks are wanted. For example, specifying **1,7,22** would indicate that new columns should begin in columns 1, 7 and 22. When the **Restore Defaults** button is clicked, the **Column breaks** field is filled with values based on the **Visual table** default recognition values.

**Notes:**

1. Column breaks can also be added or removed by using the mouse to select a column in the host screen view, right-clicking, and selecting either **Add column** or **Remove column**. Left-clicking on a column can also add or remove a column break. Column breaks are indicated by a dotted vertical line on the host screen view.
2. Column breaks cannot split an input field.
3. For information about using this setting with DBCS support see "Table" on page 462.

**Include empty rows**
If selected, empty rows (rows without any visible text or input fields) will be included in the rendered table or graph.

**Table rows to exclude**
The comma-separated or ranged set of rows to exclude from the recognized table. This setting is useful for hiding specific rows of data on a table or graph. The rows represented by this value are based on the recognized table, not on the actual host screen. Example values: **1,2,3....**, **1,2-4**, **3-5**.

**Table columns to exclude**
The comma-separated or ranged set of logical columns to exclude from the

recognized table. The columns represented by this value are based on the recognized table, not on the actual host screen. Example values: **1,2,3....**, **1,2-4**, **3-5**.

**Number of title rows**
Specifies the number of rows, starting at the top of the region, that should be rendered as the header. If the value is greater than 1, for each column, the header lines will be rendered as a string in a single row.

---

## Table (field)

The Table (field) component is responsible for recognizing logical columns of vertically aligned fields. For a region to be recognized as a table (field), the left and right boundaries of each field contained within the selected region (including partial fields) must be the same as the left and right boundaries of the field directly above (if the field in question is not on the first row) and directly below (unless the field in question is not on the last row).

A recognized table (field) can be rendered by the following widgets:

- Graph (horizontal bar, line, vertical bar)
- Table
- Enhanced grid (Dojo) **Web-only**

The following figure shows how a table (field) component might appear on your host screen:

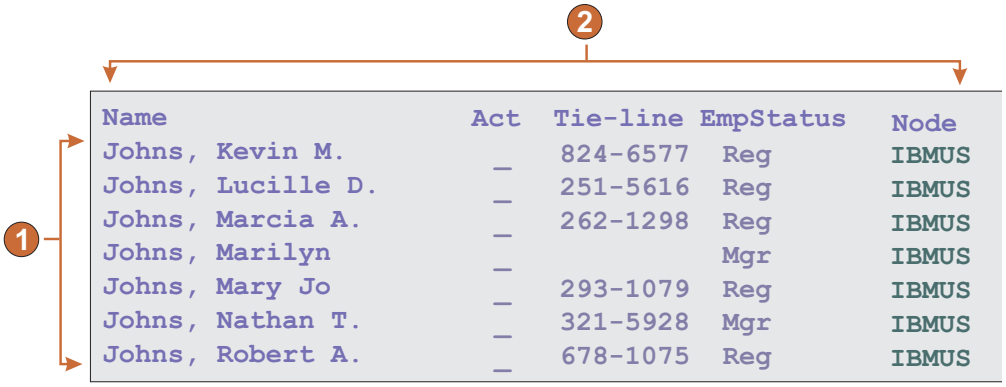



*Figure 26. Table (field) component example*

1. Rows
2. Columns

The following settings can be configured for this component:

**Include empty rows**
If selected, empty rows (rows without any visible text or input fields) will be included in the rendered table or graph.

**Rows to exclude**
The comma-separated or ranged set of rows to exclude from the recognized table. This setting is useful for hiding specific rows of data on a

table or graph. The rows represented by this value are based on the recognized table, not on the actual host screen. Example values: **1,2,3...**, **1,2-4**, **3-5**.

**Columns to exclude**
The comma-separated or ranged set of logical columns to exclude from the recognized table. The columns represented by this value are based on the recognized table, not on the actual host screen. Example values: **1,2,3...**, **1,2-4**, **3-5**.

**Minimum row count**
The minimum number of rows the specified region must contain for the region to be recognized as a table.

**Note:** The number of recognized rows is evaluated before the specified rows are excluded (see **Rows to exclude**).

**Minimum column count**
The minimum number of logical columns (not host screen columns) the specified region must contain for the region to be recognized as a table.

**Note:** The number of recognized columns is evaluated before the specified columns are excluded (see **Columns to exclude**).

**Extract column header text from row above table**
Select this box if you want the text from the line above the table to be used as the column headers of the table. The column headers must be in protected fields.

**Note:** Unless you have access to the host screen map which created the table, use the Table (visual) component to recognize a tabular region. If this is not possible, turn on the **Input**, **Protected**, and **Hidden** options on the first page of the Insert Host Component wizard to help determine where fields break within the selected host screen region.

---

## Table (visual)

The Table (visual) component is responsible for recognizing tables. The Table (visual) component is different from the Table (field) component in that it does not require that the underlying host screen fields line up vertically. It looks for columns in the selected region which contain the column delimiter for each row in the selection. Rule of thumb: If a region has the appearance of a table, the Table (visual) component should be able to recognize it.

A recognized table can be rendered by the following widgets:

- Graph (horizontal bar, line, vertical bar)
- Table
- Enhanced grid (Dojo) `Web-only`

The following figure shows how a Table (visual) component might appear on your host screen:

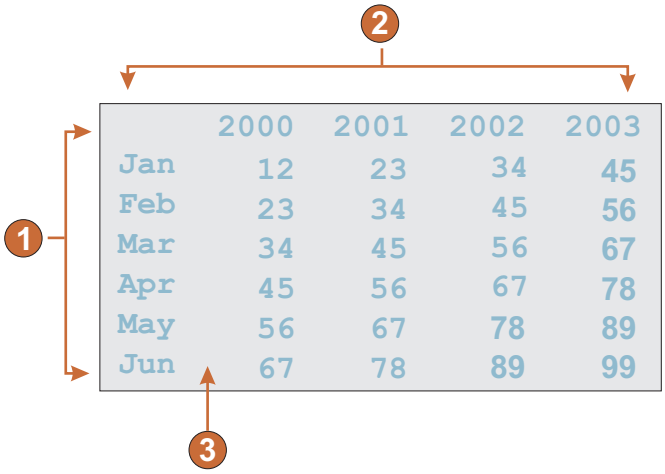



*Figure 27. Table (visual) component example*

1. Rows (minimum row count = 1)
2. Columns (minimum column count = 1)
3. Column delimiter (a space)

The following settings can be configured for this component:

**Column delimiter**
Required. The string of characters which must exist in every row of the selected region for the host screen column to be marked as a table column. If consecutive columns are each marked as a delimiting column, only the last column is actually rendered as the delimiting column of the generated table.

**Include empty rows**
If selected, empty rows (rows without any visible text or input fields) will be included in the rendered table or graph.

**Rows to exclude**
The comma-separated or ranged set of rows to exclude from the recognized table. This setting is useful for hiding specific rows of data on a table or graph. The rows represented by this value are based on the recognized table, not on the actual host screen. Example values: **1,2,3....**, **1,2-4**, **3-5**.

**Columns to exclude**
The comma-separated or ranged set of logical columns to exclude from the recognized table. The columns represented by this value are based on the recognized table, not on the actual host screen. Example values: **1,2,3....**, **1,2-4**, **3-5**.

**Minimum row count**
The minimum number of rows the specified region must contain for the region to be recognized as a table.

**Note:** The number of recognized rows is evaluated before the specified rows are excluded (see **Rows to exclude**).

**Minimum column count**
The minimum number of logical columns (not host screen columns) the specified region must contain for the region to be recognized as a table.

**Note:** The number of recognized columns is evaluated before the specified columns are excluded (see **Columns to exclude**).

**Extract column header text from row above table**
Select this box if you want the text from the line above the table to be used as the column headers of the table. The column headers must be in protected fields.

---

## Text

The Text component is responsible for recognizing visible text, including text within an input field.

This component can be rendered using the Label widget.

The following figure shows how a text component might appear on your host screen:

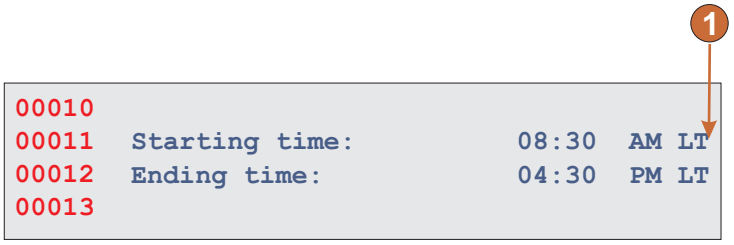



*Figure 28. Text component example*

1. Line break (preserved)

The following settings can be configured for this component:

**Preserve line breaks**
If selected, multiple selected rows will be rendered as multiple rows on the resulting Web page. If cleared, the text from each row will be concatenated together into a single string.

---

## URL

The URL component is responsible for recognizing links. It searches only within protected, non-hidden fields. This component can be rendered using the Link widget.

The following figure shows how a URL component might appear on your host screen:


*Figure 29. URL component example*

1. Start indicator (http://)

The following settings can be configured for this component:

**Start Indicators**
Specify one or more character strings to recognize the beginning of an URL on the host screen

## Widget settings

When you customize a host component, you specify how it will be recognized. When you customize a widget, you specify how the widget will appear on the Web page.

You can customize the settings of the widgets described in the following sections.

When you set or modify widget settings in any widget settings dialog, the widget preview area is automatically updated to show you the effect of the changes you make, before you click OK to commit those changes.

### Button

The Button widget is responsible for rendering HTML buttons. Clicking one of these buttons will generally cause a host-specific action to occur (for example, an aid key will be sent to the host or an input field will be updated).

This widget renders data supplied by the following components:

- Function key
- Light pen (attention)
- Selection list
- Selection field (ENPTUI)

The following figure shows how a button widget appears on a transformation, using the data from the Function key component example as input:

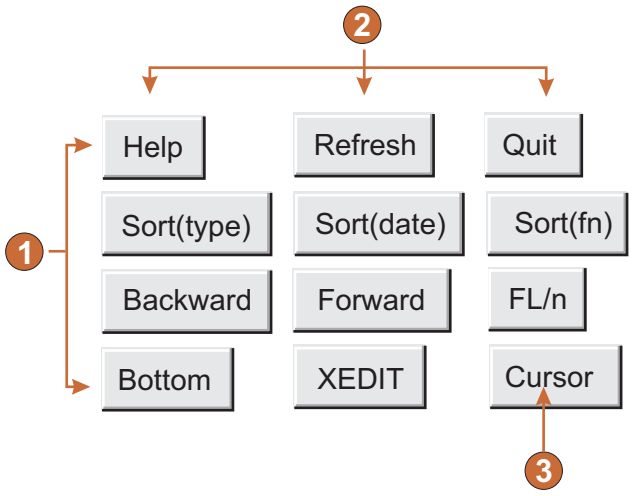



*Figure 30. Button widget example*

1. Rows
2. Columns (3 per row)
3. Caption type (show description)

The following settings can be configured for this widget:

**Caption type**
Specifies how the caption for each button is determined. The value of the leading token and the description are derived from the component; you can select what appears on the button caption. For example, if the host screen had a menu item that read **4. Mail**, you can have the caption display **4**, or **Mail**, or **4. Mail**.

**Trim spaces on caption**
If selected, white space (extra space) is trimmed from both ends of the caption.

**Layout**
Specifies how the buttons will be arranged on the HTML page. Options: **Table**, **Separated**.

**Notes:**

1. This setting is not applicable in default rendering if the function key component is being used.
2. The widget preview does not always match the full page preview. This happens because the widget preview simply renders the component's output. It does not try to preserve the alignment of the screen when it renders, as is done in the full page preview.

**Number of columns per row**
The number of buttons to display horizontally before wrapping to the next line. Tweaking this setting can cause your buttons to appear vertically (if this value is 1) or horizontally (if this value is equal to or greater than the number of buttons to be rendered).

**Separator**
Specifies what separator you want to use to separate the rendered buttons on the HTML page. You can select from the drop-down list or type in your own in the entry field.

**Enable foreground colors** `Web-only`
If selected, the button text is rendered using the color extracted from the host screen by the component (only applicable when the function key component is used).

Color information is extracted for the first host field containing any part of the function key. If the function key is split over two fields, such as `F12=` and `Exit`, the color information for the field containing `F12=` is extracted.

**Button style class** `Web-only`
Optional. The CSS style class associated with each generated button. The value of the class attribute of the HTML button tag will be set to this value. The default value is **HATSBUTTON**. See “Using style sheets” on page 317 for more information.

**Style** `Web-only`
Optional. Any CSS properties that you want to override. For example, you can specify **font-color: red; font-size: 18pt;** in this field to change the font color and size for this widget. The properties you enter apply to each element of this widget. Use the launcher button next to this field to open a

style properties dialog. This dialog frees you from the need to understand CSS to change the font, color, or other style settings for the widget. See "Using style sheets" on page 317 for more information.

**Override font** `RCP-only`
Select this option and use the supplied **Font** setting controls to override the default font name, font style, and font size.

---

## Button table

The Button Table widget is responsible for rendering a table of vertically arranged HTML buttons.

This widget renders data supplied by the following components:

- Function key
- Selection list

The following figure shows how a button table widget appears on a transformation, using the data from the Function key component example as input:

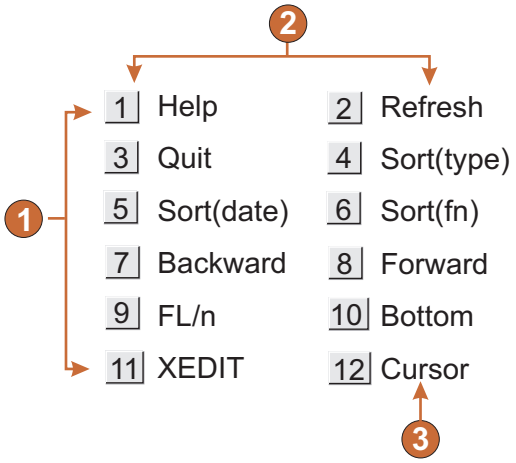



*Figure 31. Button table widget example*

1. Rows
2. Number of columns per row (2)
3. Caption

The following settings can be configured for this widget:

**Number of columns per row**
The number of buttons to display horizontally before wrapping to the next line.

**Notes:**

1. This setting is not applicable in default rendering if the function key component is being used.
2. The widget preview does not always match the full page preview. This happens because the widget preview simply renders the component's

output. It does not try to preserve the alignment of the screen when it renders, as is done in the full page preview.

**Button style class** `Web-only`
Optional. The CSS style class associated with each generated button. The value of the class attribute of the HTML button tag will be set to this value. The default value is **HATSBUTTON**. See "Using style sheets" on page 317 for more information.

**Caption style class** `Web-only`
Optional. The CSS style class associated with each item description (to the right of the generated button). The default value is **HATSCAPTION**. See "Using style sheets" on page 317 for more information.

**Table style class** `Web-only`
Optional. The CSS style class associated with the generated table. The value of the class attribute of the HTML table tag will be set to this value. The default is **HATSTABLE**. See "Using style sheets" on page 317 for more information.

**Style** `Web-only`
Optional. Any CSS properties that you want to override. For example, you can specify **font-color: red; font-size: 18pt;** in this field to change the font color and size for this widget. The properties you enter apply to each element of this widget. Use the launcher button next to this field to open a style properties dialog. This dialog frees you from the need to understand CSS to change the font, color, or other style settings for the widget. See "Using style sheets" on page 317 for more information.

**Override font** `RCP-only`
Select this option and use the supplied **Font** setting controls to override the default font name, font style, and font size.

**Override color** `RCP-only`
Select this option and use the supplied **Foreground color** and **Background color** setting controls to override foreground and background colors.

---

## Calendar `Web-only`

The Calendar widget is responsible for rendering an input field with an associated launcher button, link, or image. This launcher opens a pop-up calendar date picker which your users can use to select a date; this date is then inserted into the associated input field.

**Notes:**

1. If you are testing your project using the Web browser built in to the Rational SDP, using the WebSphere Test Environment, you may see the calendar widget come up in a browser window bigger than expected and with additional unneeded controls. This will not happen when the application is run in an external browser, or when you run a deployed application.
2. To display the pop-up calendar and image launcher when the HATS application is accessed through a proxy server, you must configure the HATS application to use the proxy server. For instructions see "Configuring HATS applications to use a proxy server" on page 35.
3. For DBCS considerations when using this widget see "SBCS eliminate maximum length" on page 462.

The following figure shows how the calendar widget and its date picker appear on a transformation when the **Use inline calendar** setting is selected:

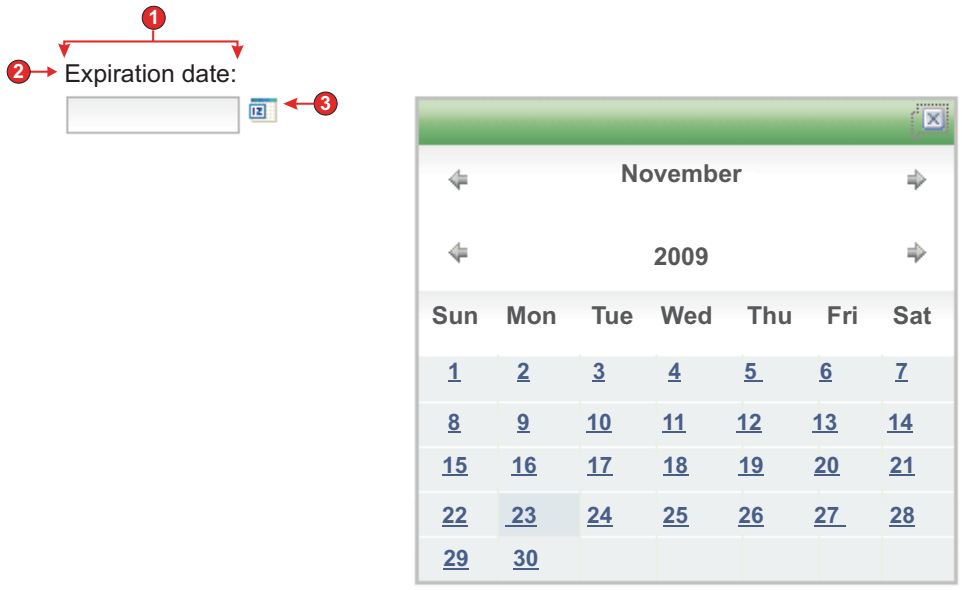



*Figure 32. Example of calendar widget with inline date picker*

The following figure shows how the calendar widget and its date picker appear on a transformation when the **Use inline calendar** setting is cleared:

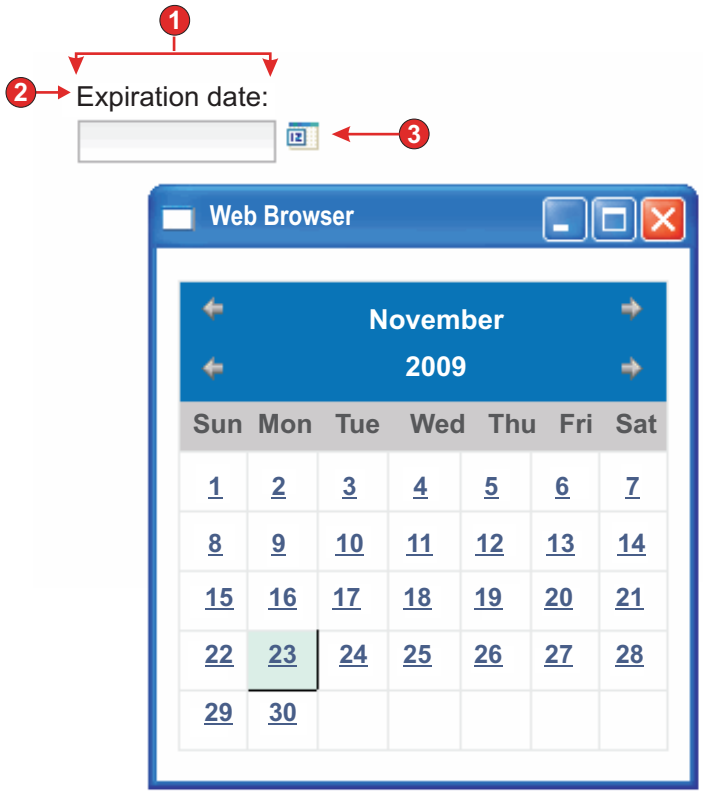



*Figure 33. Example of calendar widget with windowed date picker*

1. Number of columns per row (1)
2. Caption (from component)
3. Launcher type (image)

**Note:** The examples have no captions for the launchers because the launchers are images. If the launchers are buttons or links, the launcher captions are shown on the buttons or in the links.

The following settings can be configured for this widget:

**Pattern**

Required. When the user selects a date from the calendar widget, the selected date must be formatted to map correctly to what is expected in the host application's input field. This setting specifies the pattern HATS uses to correctly format the selected date. For information about the meaning of symbols in the pattern, see http://java.sun.com/j2se/1.3/docs/api/java/text/SimpleDateFormat.html.

When you enter a pattern and save it, the HATS Toolkit performs the following conversions to ensure correct symbols are used:

- Uppercase `D` to lowercase `d`
- Uppercase `Y` to lowercase `y`
- Lowercase `m` to uppercase `M`
- Lowercase `e` to uppercase `E`

**Note:** The HATS Toolkit always displays the pattern in uppercase characters. To see the actual case of the pattern symbols, see the pattern setting for the com.ibm.hats.transform.widgets.CalendarWidget class in the source of the application.hap file.

You can spread the selected date over multiple consecutive host input fields by separating pattern elements with a tilde (~). For example, suppose you have the following input fields on your host screen:

**Month: [ ] Day: [ ] Year: [ ]**

If you specify a pattern of MM~DD~YYYY, a single launcher will be created. Once your user has selected a date, it will be spread over your three input fields: the 2-digit month will be placed in the first input field, the 2-digit day will be placed in the second input field, and the 4-digit year will be placed in the third input field.

**Note:** Click the **Build** button to use an aid to specify a pattern that spans multiple host input fields.

**Pattern locale**
Specifies the locale of the pattern. This setting is useful if you have entered a pattern segment which should output a day name (example: Wednesday). Options: **Use server locale**, **Use browser locale**, or **Use specified locale**.

**Locale** If you specify **Use specified locale**, select the locale for which to format the date specified by the user.

**Restrict earliest selectable date**
If selected, the value specified will be the earliest date a user can select from the pop-up calendar.

**Notes:**

1. The date pattern is MM/DD/YYYY for all locales.
2. A user is not prevented from manually entering an earlier date directly into the associated input field because of this setting.

**Restrict latest selectable date**
If selected, the value specified will be the latest date a user can select from the pop-up calendar.

**Notes:**

1. The date pattern is MM/DD/YYYY for all locales.
2. A user is not prevented from manually entering a later date directly into the associated input field because of this setting.

**Default value**
Optional. This field represents the initial date selected on the pop-up calendar when the host application does not pre-fill the associated input field with a valid date between the specified restricted dates. It is also used

when the user enters an incorrect date (including a date out of range) before selecting the pop-up calendar control. The interaction between this field and the associated host input field is handled as follows:

- If the host application pre-fills the input field with zeros or an incorrect date format, the default value does not overwrite the host field data. To update the associated input field, the user must either manually enter a date, or select a date with the pop-up calendar.
- If the default value is not specified or is in an incorrect format, today's date will be initially selected on the pop-up calendar.
- If you want to pre-fill the associated input field, add an **Insert Data** action to your screen customization event before applying the transformation.

**Note:** The date pattern is MM/DD/YYYY for all locales.

**Caption source**
Specifies how the caption for the generated input field is determined. Options: **From component** (use the extracted caption from the component) or **Custom** (use the specified caption).

**Custom caption**
Optional. Specifies the caption for the input field.

**Note:** Leave this setting blank to not generate a caption for the input field.

**Number of columns per row**
Specifies how many instances of this widget should appear in each row in the rendered Web page.

**Notes:**

1. This setting is not applicable in default rendering.
2. The widget preview does not always match the full page preview. This happens because the widget preview simply renders the component's output. It does not try to preserve the alignment of the screen when it renders, as is done in the full page preview.

**Launcher type**
Specifies the style of the launcher. The launcher is used to open the calendar date picker. Options **Button**, **Link**, or **Image**.

**Caption**
The caption of the launcher button or link.

**Button style class**
Optional. The CSS style class associated with the generated launcher button. The value of the class attribute of the HTML button tag will be set to this value. The default value is **HATSBUTTON** . See “Using style sheets” on page 317 for more information.

**Link style class**
Optional. The CSS style class associated with the generated launcher link. The value of the class attribute of the HTML link tag will be set to this value. The default value is **HATSLINK**. See “Using style sheets” on page 317 for more information.

**Note:** Depending on the Web browser, there may be limitations to using the Tab key to tab to this launcher if either the **Link** or **Image** style

is selected. For a list of supported Web browsers and limitations, see "System Requirements for Host Access Transformation Services" at http://www.ibm.com/support/docview.wss?uid=swg27011794 and "Host Access Transformation Services 9.7 - Known issues and workarounds" at http://www.ibm.com/support/docview.wss?uid=ibm10876092.

**Use inline calendar**
If selected, when the calendar date picker is launched, it is displayed in the current Web page. If cleared, the date picker is displayed in a new browser window. The default for newly created projects is selected. The default for projects migrated from releases before HATS V7.5.1 is cleared.

**Notes:**

1. Using the inline calendar date picker is helpful in cases where pop-up blockers prevent the display of the date picker in a new browser window.
2. Use of the inline calendar date picker is not supported for HATS applications for mobile devices.

**Read only**
If selected, rendered input fields are read only. A read only input field appears as a regular input field, but does not allow the user to modify its contents. This is useful in cases where you want to display the contents of a non-protected field to a user, but you do not want the user to modify the contents.

**Strip underscores on input field**
Select this box if you want to strip the underscores from text when it is rendered.

**Trim spaces on input field**
Selecting this trims leading and trailing spaces from the input field.

**Enable foreground colors**
If selected, host screen foreground colors are rendered.

Colors are mapped to CSS stylesheet classes representing that color. For example, if a host screen field is marked as **RED**, the Field widget will enclose the generated HTML for that field in a tag whose class name attribute is set to **HRED**. This allows for you to remap host screen colors on your generated Web page.

**Note:** The blacktheme.css directly maps host screen field foreground colors (for example, red text on the host screen will appear as red text on the generated Web page). However, other stylesheet files like monochrome.css, map field colors differently in an attempt to create a consistent, modern style.

**Enable extended attributes**
If selected, extended field attributes (blink, reverse video, underline, and column separator) are rendered. Also, for 3270 Web applications, extended field colors are mapped (see the **Enable foreground colors** setting for more information). See “Using style sheets” on page 317 for more information.

**Blink style**
The CSS style setting you want to use to render blinking text from the host screen.

**Reverse video style**
The CSS style setting you want to use to render reverse video text from the host screen.

**Underline style**
The CSS style setting you want to use to render underlined text from the host screen.

**Column separator style**
The CSS style setting you want to use to render column separators from the host screen.

**Input field style class**
Optional. The CSS style class associated with the generated input field. The value of the class attribute of the HTML input tag will be set to this value. The default value is **HATSINPUT**. See "Using style sheets" on page 317 for more information.

**Caption style class**
Optional. The CSS style class associated with the generated input field's caption. The default value is **HATSCAPTION**. See "Using style sheets" on page 317 for more information.

**Table style class**
Optional. The CSS style class associated with the generated table. If more than one set of input fields is rendered, an HTML table will be generated to enclose these input fields. The value of the class attribute of the HTML table tag will be set to this value. The default value is **HATSTABLE**. See "Using style sheets" on page 317 for more information.

**Style** Optional. Any CSS properties that you want to override. For example, you can specify **font-color: red; font-size: 18pt;** in this field to change the font color and size for this widget. The properties you enter apply to each element of this widget. Use the launcher button next to this field to open a style properties dialog. This dialog frees you from the need to understand CSS to change the font, color, or other style settings for the widget. See "Using style sheets" on page 317 for more information.

---

## Check box

The Check box widget is responsible for rendering HTML check boxes.

This widget renders data supplied by the following components:

- Input field
- Item selection
- Light pen (selection)
- Selection field (ENPTUI)

The following figure shows how a check box widget appears on a transformation, using the data from the input field of the Input field with hints component example as input:

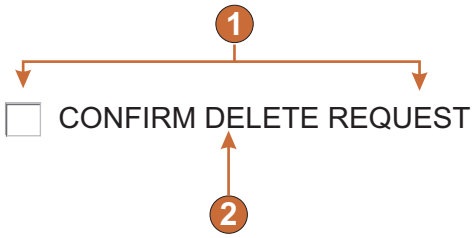



*Figure 34. Check box widget example*

1. Number of columns per row (1)
2. Caption (from component)

The following settings can be configured for this widget:

**Selected value**
Required. The string that is inserted in the host screen input field when the check box is selected. This value is also used to set the initial state of the check box when the Web page is loaded.

**Deselected value**
Required. The string that is inserted in the host screen input field when the check box is deselected. This value is also used to set the initial state of the check box when the Web page is loaded.

**Caption source**
Specifies how the caption for the generated check box is determined. Options: **From component** (use the extracted caption from the component) or **Custom** (use the specified caption).

**Custom caption**
Optional. Specifies the caption for the check box.

**Note:** Leave this setting blank to not generate a caption for the check box.

**Trim spaces on caption**
If selected, white space (extra space) is trimmed from both ends of the caption.

**Number of columns per row**
The number of check boxes to display horizontally before wrapping to the next line.

**Notes:**

1. This setting is not applicable in default rendering.
2. The widget preview does not always match the full page preview. This happens because the widget preview simply renders the component's output. It does not try to preserve the alignment of the screen when it renders, as is done in the full page preview.

**Check box style class** `Web-only`
Optional. The CSS style class associated with the generated check box. The value of the class attribute of the HTML input tag will be set to this value. The default value is **HATSCHECKBOX**. See “Using style sheets” on page 317 for more information.

**Caption style class** `Web-only`

Optional. The CSS style class associated with the caption of the generated check box. The default value is **HATSCAPTION**. See "Using style sheets" on page 317 for more information.

**Table style class** `Web-only`

Optional. The CSS style class associated with the generated table. If more than one check box is rendered, an HTML table will be generated to enclose these check boxes. The value of the class attribute of the HTML table tag will be set to this value. The default value is **HATSTABLE**. See "Using style sheets" on page 317 for more information.

**Style** `Web-only`

Optional. Any CSS properties that you want to override. For example, you can specify **font-color: red; font-size: 18pt;** in this field to change the font color and size for this widget. The properties you enter apply to each element of this widget. Use the launcher button next to this field to open a style properties dialog. This dialog frees you from the need to understand CSS to change the font, color, or other style settings for the widget. See "Using style sheets" on page 317 for more information.

**Override font** `RCP-only`

Select this option and use the supplied **Font** setting controls to override the default font name, font style, and font size.

---

## Combo `RCP-only`

The Combo widget is responsible for rendering editable drop-down combination boxes.

**Notes:**

1. For security reasons, the Combo widget should not be used to render password fields. Characters typed into a password field rendered with the Combo widget will be visible as the user types.
2. For DBCS considerations when using this widget see "AutoIME switching" on page 459.

This widget renders data supplied by the following components:

- Command line
- Input field
- Input field with hints
- Item selection

The following figure shows how a combo widget appears on a transformation, using the data from the Input field with hints component example as input:

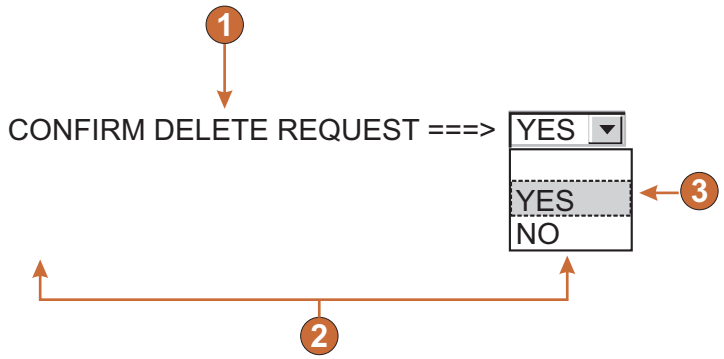



*Figure 35. Combo widget example*

1. Caption (from component)
2. Number of columns per row (defaults to 1)
3. Fill (from hints)

The following settings can be configured for this widget:

**Fill from global variable**
If selected, fill the drop-down from the specified global variables.

**Global variable containing list values**
Specifies the name of the indexed global variable containing the set of values. An item for each index in the global variable will be created in the drop-down.

**Shared**
HATS local and shared global variables can have the same name. Select this box if you want to use the shared global variable to populate the list items. When this box is cleared, the local global variable is used.

**Global variable containing list captions**
Optional. Specifies the name of the indexed global variable containing the set of captions. The size of the global variable specified by this value should be greater than or equal to the size specified in the preceding setting. The indexes in this indexed global variable should also match up with the indexes in the values global variable (so that the actual value and caption shown to the user are in sync). If this value is not specified, the caption for each item in the drop-down will be its value.

**Fill from string**
If selected, fill the drop-down from the specified string.

**List items**
Optional. Specifies the string of items to include in the drop-down list. Items should be separated with a semicolon (;). To have the list item caption be different than the list item value, enter both separated by an equal sign (=). For example, a value of **Apple=A;Grape=G** renders a drop-down list with two items: **Apple** and **Grape**. Selecting the first item causes an **A** to be inserted in the associated host screen input field.

If you want both the item in the drop-down and the value inserted in the host screen to be the same, you only need to enter the item. For example, **Apple=A;G**. In this example, a **G** appears in the drop-down and in the host screen input field.

**Fill from hints**
If selected, fill the drop-down from hints recognized by the component (only applicable when the Input Field with Hints component is used).

**Caption source**
Specifies how the caption for the generated drop-down is determined. Options: **From component** (use the extracted caption from the component) or **Custom** (use the specified caption).

**Custom caption**
Optional. Specifies the caption for the drop-down.

**Note:** Leave this setting blank to not generate a caption for the drop-down.

**Number of columns per row**
The number of drop-downs to display horizontally before wrapping to the next line. Default is 1.

**Notes:**

1. This setting is not applicable in default rendering.
2. The widget preview does not always match the full page preview. This happens because the widget preview simply renders the component's output. It does not try to preserve the alignment of the screen when it renders, as is done in the full page preview.

**Override maximum length**
If selected, the specified maximum length value will be set on each generate input field. This setting is useful for manually restricting the number of characters that can be entered into an input field. If cleared, the maximum number of allowable characters is determined by the recognizing component.

**Maximum length**
The maximum number of characters that can be entered into each input field.

**Enable foreground colors**
If selected, host screen foreground colors are rendered.

Colors are mapped by the rich client template.

**Note:** This setting is overridden by the **Override color > Foreground color** setting.

**Enable extended attributes**
If selected, extended field attributes (blink, reverse video, underline, and column separator) are rendered.

Extended field attributes are mapped by the rich client template.

**Override font**
Select this option and use the supplied **Font** setting controls to override the default font name, font style, and font size.

**Override color**
Select this option and use the supplied **Foreground color** and **Background color** setting controls to override foreground and background colors.

---

## Dialog

The Dialog widget is responsible for rendering dialog boxes. It renders data supplied by the Dialog component.

The following figure shows how a Dialog widget appears on a transformation using the data from the Dialog component example as input:

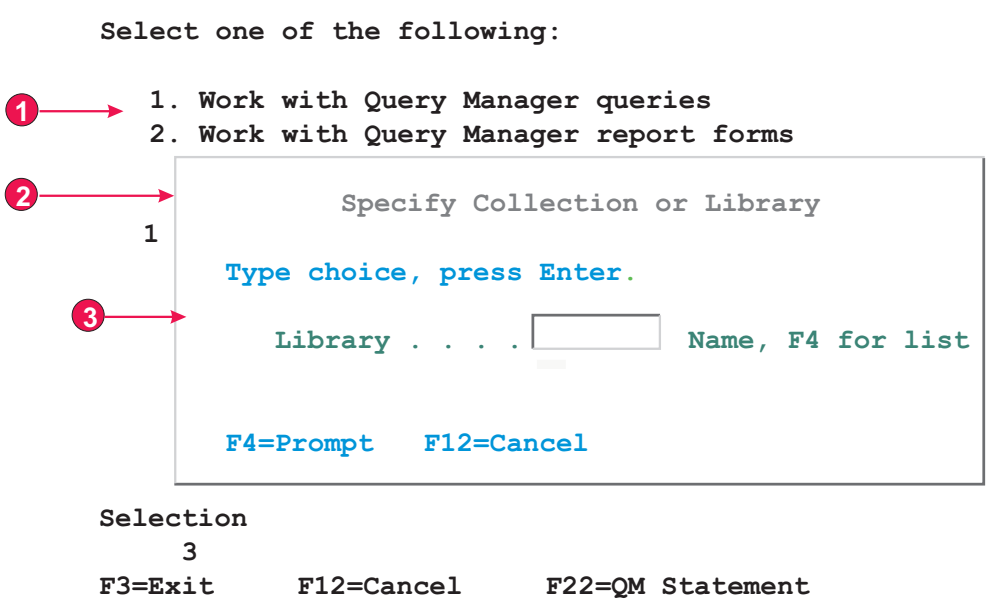



*Figure 36. Dialog widget example*

1. Disabled area
2. Specialized border
3. Border characters removed

The following settings can be configured for this widget:

**Hide disabled area outside dialog**
Specifies whether to hide the area outside of the dialog.

**Disabled area style class** `Web-only`
Specifies the CSS style class to use for the disabled area.

**Use specialized borders** `Web-only`
Specifies whether to use specialized borders instead of the original text.

**Remove border characters** `Web-only`
Specifies whether to show the original border characters.

**Border style class**
Specifies the CSS style class to use for the border.

**Border style** `Web-only`

**Left** Specifies the CSS style class to use for the left border.

**Right** Specifies the CSS style class to use for the right border.

**Top** Specifies the CSS style class to use for the top border.

**Bottom**
Specifies the CSS style class to use for the bottom border.

**Corner style** `Web-only`

**Top left**
Specifies the CSS style class to use for the top left border corner.

**Top right**
Specifies the CSS style class to use for the top right border corner.

**Bottom left**
Specifies the CSS style class to use for the bottom left border corner.

**Bottom right**
Specifies the CSS style class to use for the bottom right border corner.

---

## Drop-down (data entry)

The Drop-down (data entry) widget is responsible for rendering drop-down lists.

This widget renders data supplied by the following components:

- Command line
- Input field
- Input field with hints
- Item selection

The following figure shows how a data entry drop-down list widget appears on a transformation, using the data from the Input field with hints component example as input:

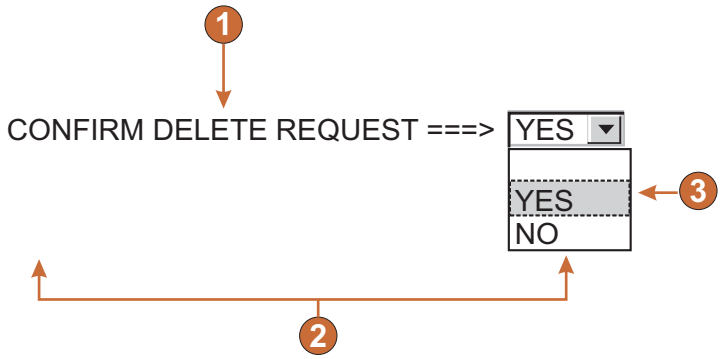



*Figure 37. Data entry drop-down list widget example*

1. Caption (from component)
2. Number of columns per row (1)
3. Fill (from hints)

The following settings can be configured for this widget:

**Fill from global variable**
If selected, fill the drop-down list from the specified global variables.

**Global variable containing list values**
Specifies the name of the indexed global variable containing the set of values. An item for each index in the global variable will be created in the drop-down list.

**Shared**
HATS local and shared global variables can have the same name. Select this box if you want to use the shared global variable to populate the list items. When this box is cleared, the local global variable is used.

**Global variable containing list captions**
Optional. Specifies the name of the indexed global variable containing the set of captions. The size of the global variable specified by this value should be greater than or equal to the size specified in the preceding setting. The indexes in this indexed global variable should also match up with the indexes in the values global variable (so that the actual value and caption shown to the user are in sync). If this value is not specified, the caption for each item in the drop-down list will be its value.

**Fill from string**
If selected, fill the drop-down list from the specified string.

**List items**
Optional. Specifies the string of items to include in the drop-down list. Items should be separated with a semicolon (;). To have the list item caption be different than the list item value, enter both separated by an equal sign (=). For example, a value of **Apple=A;Grape=G** renders a drop-down list with two items: **Apple** and **Grape**. Selecting the first item causes an **A** to be inserted in the associated host screen input field.

If you want both the item in the drop-down list and the value inserted in the host screen to be the same, you only need to enter the item. For example, **Apple=A;G**. In this example, a **G** appears in the drop-down list and in the host screen input field.

**Fill from hints**
If selected, fill the drop-down list from hints recognized by the component (only applicable when the Input Field with Hints component is used).

**Caption source**
Specifies how the caption for the generated drop-down list is determined. Options: **From component** (use the extracted caption from the component) or **Custom** (use the specified caption).

**Custom caption**
Optional. Specifies the caption for the drop-down list.

**Note:** Leave this setting blank to not generate a caption for the drop-down list.

**Number of displayable rows**
The number of rows in the drop-down list that are visible to the user. If

more than this number of items are in the list, scroll bars appear so the user can scroll through the entire list. The default is **1**.

**Number of columns per row**
The number of drop-down lists to display horizontally before wrapping to the next line.

**Notes:**

1. This setting is not applicable in default rendering.
2. The widget preview does not always match the full page preview. This happens because the widget preview simply renders the component's output. It does not try to preserve the alignment of the screen when it renders, as is done in the full page preview.

**Auto submit on select**
If selected, once a selection is made in the drop-down list, it is submitted.

**Show submit button**
If selected, a submit button is rendered next to the drop-down list.

**Submit button caption**
Optional. Specifies the caption of the submit button.

**Drop-down style class** `Web-only`
Optional. The CSS style class associated with the generated drop-down list. The value of the class attribute of the HTML drop-down tag will be set to this value. The default value is **HATSDROPDOWN**. See "Using style sheets" on page 317 for more information.

**List option style class** `Web-only`
Optional. The CSS style class associated with each option in the drop-down list. The default value is **HATSOPTION**. See "Using style sheets" on page 317 for more information.

**Caption style class** `Web-only`
Optional. The CSS style class associated with the caption of the generated drop-down list. The default value is **HATSCAPTION**. See "Using style sheets" on page 317 for more information.

**Table style class** `Web-only`
Optional. The CSS style class associated with the generated table. If more than one drop-down list is rendered, an HTML table will be generated to enclose these drop-down lists. The value of the class attribute of the HTML table tag will be set to this value. The default value is **HATSTABLE**. See "Using style sheets" on page 317 for more information.

**Style** `Web-only`
Optional. Any CSS properties that you want to override. For example, you can specify **font-color: red; font-size: 18pt;** in this field to change the font color and size for this widget. The properties you enter apply to each element of this widget. Use the launcher button next to this field to open a style properties dialog. This dialog frees you from the need to understand CSS to change the font, color, or other style settings for the widget. See "Using style sheets" on page 317 for more information.

**Override font** `RCP-only`
Select this option and use the supplied **Font** setting controls to override the default font name, font style, and font size.

**Override color** RCP-only

Select this option and use the supplied **Foreground color** and **Background color** setting controls to override foreground and background colors.

---

## Drop-down (selection)

The Drop-down (selection) widget is responsible for rendering HTML drop-down lists.

This widget renders data supplied by the following components:

- Function key
- Selection list

The following figure shows how a selection list drop-down list widget appears on a transformation, using the data from the Selection list component example as input:

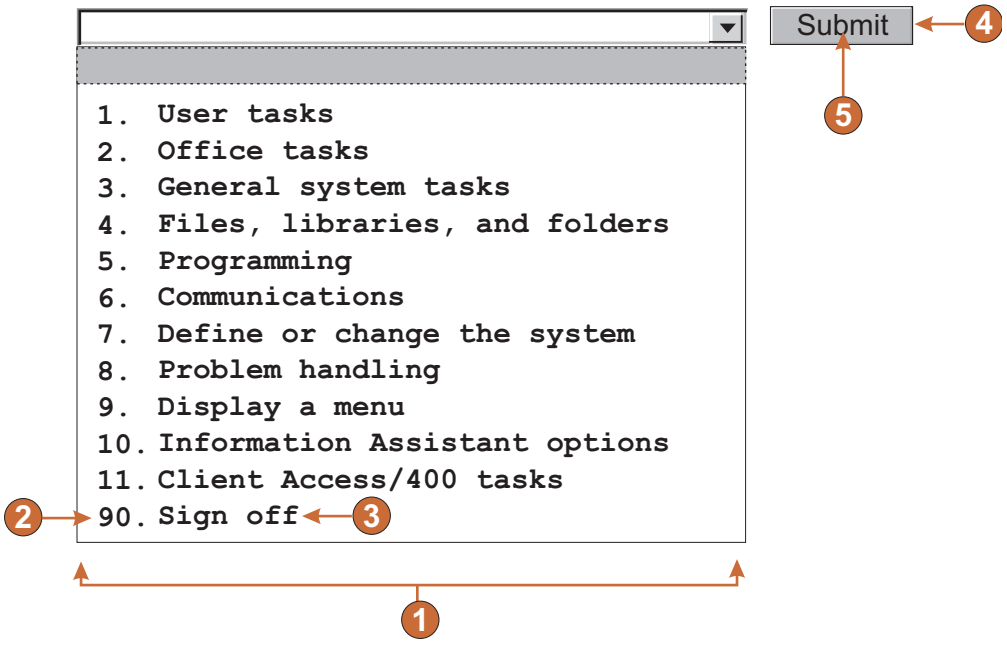



*Figure 38. Selection list drop-down list widget example*

1. Caption type (show both)
2. Leading token (show both)
3. Description (show both)
4. Submit button (show submit button)
5. Submit button caption (submit)

The following settings can be configured for this widget:

**Caption type**

Specifies how the leading token of the list item is displayed. The value of the leading token and the description are derived from the component; you

can select what appears in the drop-down list. For example, if the host screen had a menu item that read **4. Mail**, you can have the caption display **4**, or **Mail**, or **4. Mail**.

**Auto submit on select**
If selected, once a selection is made in the drop-down list, it is submitted. If the drop-down list contains function keys, the selected list item's key is sent to the host. If the drop-down list contains selection list items, the selected item's value is inserted into the target field and the specified action key is sent to the host.

**Show submit button**
If selected, a submit button is rendered next to the drop-down list. Does not apply when using this widget with the function key component.

**Submit button caption**
Optional. Specifies the caption of the submit button.

**Button style class** `Web-only`
Optional. The CSS style class associated with the generated submit button. The value of the class attribute of the HTML button tag will be set to this value. The default value is **HATSBUTTON**. See "Using style sheets" on page 317 for more information.

**Drop-down style class** `Web-only`
Optional. The CSS style class associated with the generated drop-down list. The value of the class attribute of the HTML drop-down tag will be set to this value. The default value is **HATSDROPDOWN**. See "Using style sheets" on page 317 for more information.

**List option style class** `Web-only`
Optional. The CSS style class associated with each option in the drop-down list. The default value is **HATSOPTION**. See "Using style sheets" on page 317 for more information.

**Caption style class** `Web-only`
Optional. The CSS style class associated with the caption of the generated drop-down list. The default value is **HATSCAPTION**. See "Using style sheets" on page 317 for more information.

**Table style class** `Web-only`
Optional. The CSS style class associated with the generated table. If more than one drop-down list is rendered, an HTML table will be generated to enclose these drop-down lists. The value of the class attribute of the HTML table tag will be set to this value. The default value is **HATSTABLE**. See "Using style sheets" on page 317 for more information.

**Style** `Web-only`
Optional. Any CSS properties that you want to override. For example, you can specify **font-color: red; font-size: 18pt;** in this field to change the font color and size for this widget. The properties you enter apply to each element of this widget. Use the launcher button next to this field to open a style properties dialog. This dialog frees you from the need to understand CSS to change the font, color, or other style settings for the widget. See "Using style sheets" on page 317 for more information.

**Override font** `RCP-only`
Select this option and use the supplied **Font** setting controls to override the default font name, font style, and font size.

**Override color** `RCP-only`
Select this option and use the supplied **Foreground color** and **Background color** setting controls to override foreground and background colors.

---

## Field

The Field widget is responsible for rendering input fields and text recognized by the Field component.

**Note:** For DBCS considerations when using this widget see "SBCS eliminate maximum length" on page 462.

The following figure shows how a field widget appears on a transformation, using the data from the Field component example as input:


*Figure 39. Field widget example*

1. Protected text

The primary goal of the table layout of the Field widget is to preserve the layout of the original host screen when it is rendered in a GUI. In default rendering, a table is used to preserve the layout of the screen. This table typically has 80 columns and 24 rows. Each of the different alignment settings in the Field widget (**Normal**, **Character by character** `Web-only` , and **Word by word**) change how the table cells in this table are created.

With normal alignment, each host field is represented by a table cell within the table. The cell may consume one or more columns. See the "colspan" HTML attribute for more details. If a field starts at column 10 on the host screen, it will be positioned at column 10 in the table. Normal alignment guarantees that each field begins at the correct column location.

Global rules can impact the alignment of a page because typically a global rule consumes more page space than the field it has transformed. For example, imagine you have an input field that is five characters in length. Default rendering allocates five columns in the table for this field. If the field is rendered by a global rule that takes up more than five characters of space, the table cell it resides in is forced to expand, that is, the cell's width is increased. This impacts the entire table because the entire column is affected. Typically cells to the right of the field, in any row, are shifted to the right since the columns containing the transformed field are now wider. Therefore, a field starting at column 60 of row 5 will always be aligned with a field starting at column 60 of row 20.

Character by character alignment is supported only in HATS Web projects. This alignment option causes each character of a field, except for input fields, to be rendered in its own table cell. Although using this option guarantees that every character on the screen is located at the correct starting location, it also causes the size of the generated page to be very large.

Word by word alignment causes each word on the screen to be rendered in its own table cell. This means that a single field made up of multiple words will be rendered in more than one table cell. This guarantees that every word on the screen is rendered at the correct location. This option is typically used when a host screen has a mix of input fields scattered throughout the screen. Input fields can cause a page to become misaligned since input fields consume more page space than regular text.

The following example shows how the different alignment options map fields and characters into table cells in default rendering. The table cells are invisible in actual default rendering.

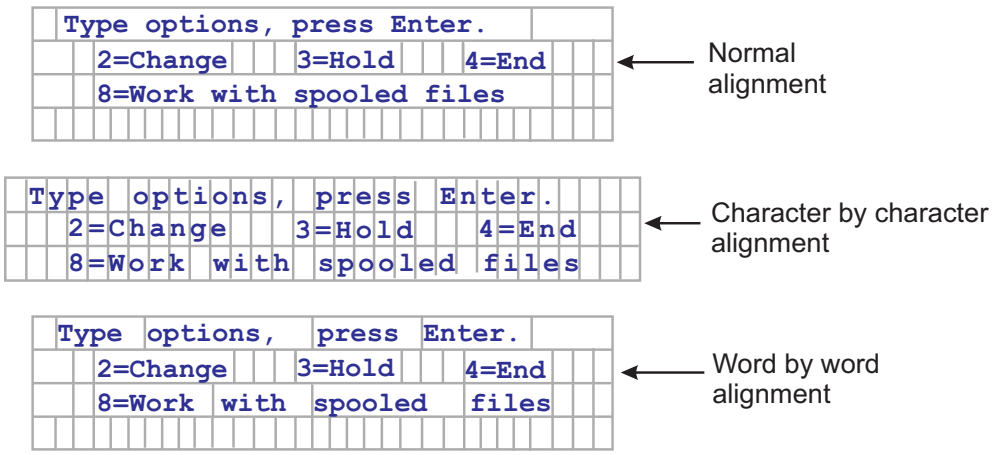



*Figure 40. Field widget alignment examples*

The following settings can be configured for this widget:

**Read only**

If selected, input fields are not rendered. If an input field contains text (and is not password protected), the text is displayed instead. This setting is useful if you want to show users a page of information, but you do not want them editing it.

**Layout**

Use this setting to specify the layout of the output rendered by this widget. Select **Table** to render the output in a table, with the goal of preserving the layout of the original host screen. This is the default for Web applications not optimized for mobile devices. Select **Separated** to render the output using inline span tags to differentiate between fields, with the goal of reducing the amount of HTML and blank space. This is the default for Web applications optimized for mobile devices.

**Alignment**

When using table layout, indicate how you want HATS to map text characters into table cells in order to improve their alignment (**Normal**, **Character by character** `Web-only` , or **Word by word**)

**Table style class** `Web-only`

Optional. The CSS style class associated with the generated table. The value of the class attribute of the HTML table tag will be set to this value. The default value is **HATSFIELDTABLE**. See "Using style sheets" on page 317 for more information.

**nowrap** Web-only

When you specify **Word by word** for the **Alignment** setting, multiple fields are aligned word by word in the transformation. However, another widget could create output that causes cells in the default rendering table to compress, thereby causing the text to wrap. To ensure that the text does not wrap, a nowrap flag is automatically added to the HTML output when word by word alignment is used. Use this **nowrap** setting to turn off the automatic addition of the nowrap flag to the HTML output.

Setting **nowrap** to **false**, with any value of the **alignment** settings, prevents the addition of the nowrap flag to the HTML output of the field widget, enabling the text in the cell to wrap. You should only set **nowrap** to **false** when you are using word by word alignment and continue to have alignment problems.

In default rendering, if a table cell wraps, both vertical and horizontal alignment can be adversely affected, causing undesirable output.

If you set **nowrap** to **true** with any value of the **alignment** setting, wrapping of text to the next line within a table cell does not occur.

**Note:** The **nowrap** setting does not appear in the GUI as a configurable setting. It must be set in the source of the application.hap file. For example, to set **nowrap** to **false**, on the HATS Projects view double-click the **Project Settings** file in the project folder to launch the editor for the application.hap file. On the **Source** tab locate the class, name="com.ibm.hats.transform.widgets.FieldWidget", in the <classSettings> tag and add the following setting:

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

**Allow cursor positioning on protected fields**

If selected, clicking one of these protected fields causes the host cursor to be set to that specific position. This setting is useful if your host application is cursor position-sensitive and needs for your user to position the cursor on a protected field.

For Web applications, how precisely the cursor position can be set is determined by the **Alignment** setting (**Normal**, **Character by character** Web-only , or **Word by word**).

For rich client applications, the cursor position can be set anywhere within the field. The **Alignment** setting does not affect how precisely the cursor position can be set.

At runtime in the browser, HATS is notified when one of these fields is clicked. HATS updates the cursor position internally and then visually indicates the new cursor position to the user. By default, the visual indication is to set the background color of the text to yellow. For Web applications, this visual indication can be controlled by the developer using the CSS style, **HCURSORINDICATOR**. See “Using style sheets” on page 317 for more information. With this setting enabled, the following capability is provided:

- Allows you to position the cursor by clicking on an area of a protected field.

- Allows you to see the cursor's initial position when the page loads, even if the cursor is not inside an unprotected field.

**Note:** This setting alone will not work correctly when the same area of a screen is rendered multiple times using the Field widget. For example, in the case where the same region of the screen is rendered on multiple tabs in a tabbed folder, you must in addition render the protected fields as links (see **Allow tabbing using links** below) in order to provide your users with the ability to position the cursor on that region of the screen.

**Allow tabbing using links** `Web-only`

If selected, each protected field is rendered as an HTML link which allows tabbing between protected fields. The cursor position will display as an underlined character.

**Note:** Selecting this option causes an HTML anchor <A> tag to be generated for each protected field on the page and as a result increases the size of the Web page.

To avoid unnecessarily increasing the size of the resulting Web page, this option is not selected by default.

**Link style class** `Web-only`

The CSS style class associated with each generated link. The value of the class attribute of the HTML link tag will be set to this value. The default value is **HATSPROTLINK**. See "Using style sheets" on page 317 for more information.

**Action key**

Optional. Specifies the host AID key mnemonic that should be sent to the host when a protected field is selected. For example: **[enter]** , which causes the cursor to be set to the specified location and for enter to be sent, or **[pf1]** , which causes the cursor to be set to the specified location and for pf1 to be sent.

**Trim spaces on links**

If selected, white space (extra space) is trimmed from both ends of the protected field links. This setting is useful for reducing the amount of text that is rendered as links.

**Enable cursor positioning option on input fields** `Web-only`

Select this box to allow the user to switch from data input mode to cursor positioning mode for input fields. When in data input mode, the user can enter data into the input field. When in cursor positioning mode, the user can tab to or otherwise position the cursor on any character in the input field. This setting is useful for Web projects that are optimized for mobile devices that do not have other cursor positioning capabilities. For more information about other settings for this option that do not appear in the HATS Toolkit GUI, see "Considerations and limitations for mobile devices" on page 44.

**Enable foreground colors**

If selected, host screen foreground colors are rendered.

For Web applications, colors are mapped to CSS stylesheet classes representing that color. For example, if a host screen field is marked as **RED**, the Field widget will enclose the generated HTML for that field in a tag whose class name attribute is set to **HRED**. This allows for you to remap host screen colors on your generated Web page.

**Note:** The blacktheme.css directly maps host screen field foreground colors (for example, red text on the host screen will appear as red text on the generated Web page). However, other stylesheet files like monochrome.css, map field colors differently in an attempt to create a consistent, modern style.

For rich client applications, colors are mapped by the rich client template.

**Note:** This setting is overridden by the **Override color > Foreground color** setting.

**Field style class** `Web-only`
Optional. The CSS style class associated with the generated field. The value of the class attribute of the HTML tag will be set to this value. The default value is **HATSFIELD**. See "Using style sheets" on page 317 for more information.

**Enable extended attributes**
If selected, extended field attributes (blink `Web-only` , reverse video, underline, and column separator) are rendered. Also, for 3270 Web applications, extended field colors are mapped (see the **Enable foreground colors** setting for more information).

For rich client applications, reverse video is rendered using the color mapped by the rich client template as the background color and black as the foreground color. Column separators are rendered with dashed lines under the field.

**Blink style** `Web-only`
The CSS style setting you want to use to render blinking text from the host screen.

**Reverse video style** `Web-only`
The CSS style setting you want to use to render reverse video text from the host screen.

**Underline style** `Web-only`
The CSS style setting you want to use to render underlined text from the host screen.

**Column separator style** `Web-only`
The CSS style setting you want to use to render column separators from the host screen.

See "Using style sheets" on page 317 for more information.

**Render using monospace font**
If selected, the field is rendered using a monospaced font. Otherwise, it is rendered using the default font for the locale environment.

**Enable 3270 numeric lock**
If selected, the user can enter only numbers 0 through 9, plus sign, minus sign, period, comma, and Hindi numerals into 3270 numeric fields. If cleared, any characters can be entered. The default is cleared.

**Note:** For Web applications, you can customize the list of acceptable keys in the lxgwfunctions.js file in the allowNumLockOnly() and allowNumLockOnlyForIEMobile() functions, as appropriate, using character code values.

**Strip underscores on input field**
Select this box if you want to strip the underscores from text when it is rendered.

**Trim spaces on input field**
Selecting this trims leading and trailing spaces from the input field.

**Style** `Web-only`
Optional. Use the launcher button next to this field to open a style properties dialog. This dialog frees you from the need to understand CSS to change the font, color, or other style settings for the widget. See "Using style sheets" on page 317 for more information.

**Note:** Because of the large number of tags generated by this widget, you may considerably increase the download time of your page if you specify a value for this setting. Specify style overrides in the enclosing transformation or the included stylesheet file.

**Override font** `RCP-only`
Select this option and use the supplied **Font** setting controls to override the default font name, font style, and font size.

**Override color** `RCP-only`
Select this option and use the supplied **Foreground color** and **Background color** setting controls to override foreground and background colors.

---

## Graph (horizontal bar, line, vertical bar)

The graph widgets are responsible for rendering JPEG graph images.

This widget renders data supplied by the following components:

- Table
- Table (field)
- Table (visual)

**Notes:**

1. To display the graph widget when the HATS application is accessed through a proxy server, you must configure the HATS application to use the proxy server. For instructions see "Configuring HATS applications to use a proxy server" on page 35.
2. This widget may not be suitable for rendering extreme ranges of data.

The following examples show how the horizontal bar graph, line graph, and vertical bar graphs appear on a transformation, using the data from the Table (visual) component example as input:

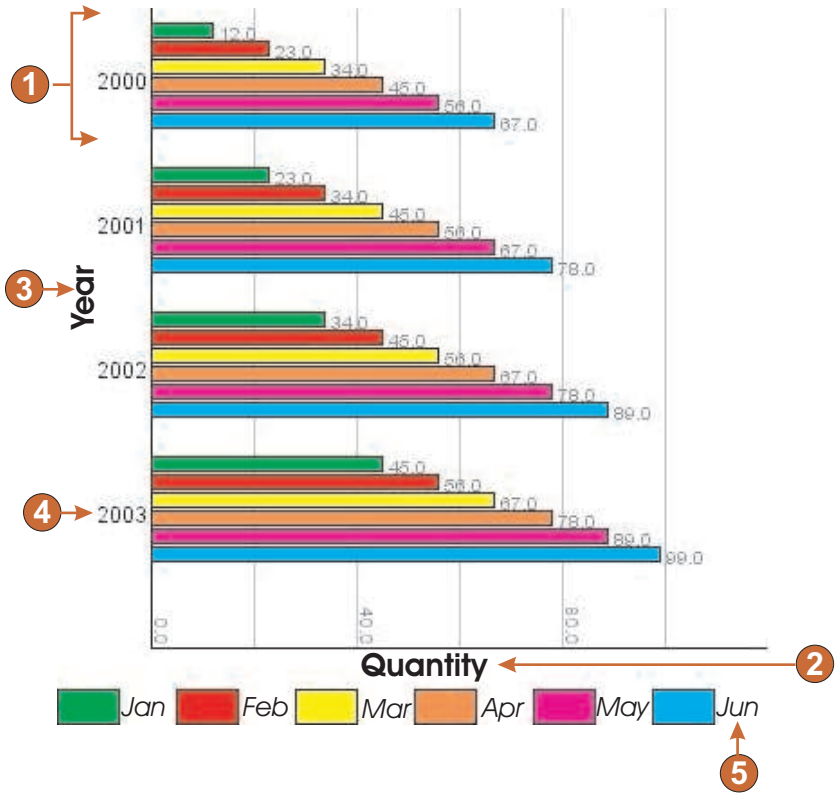



*Figure 41. Horizontal bar graph widget example*

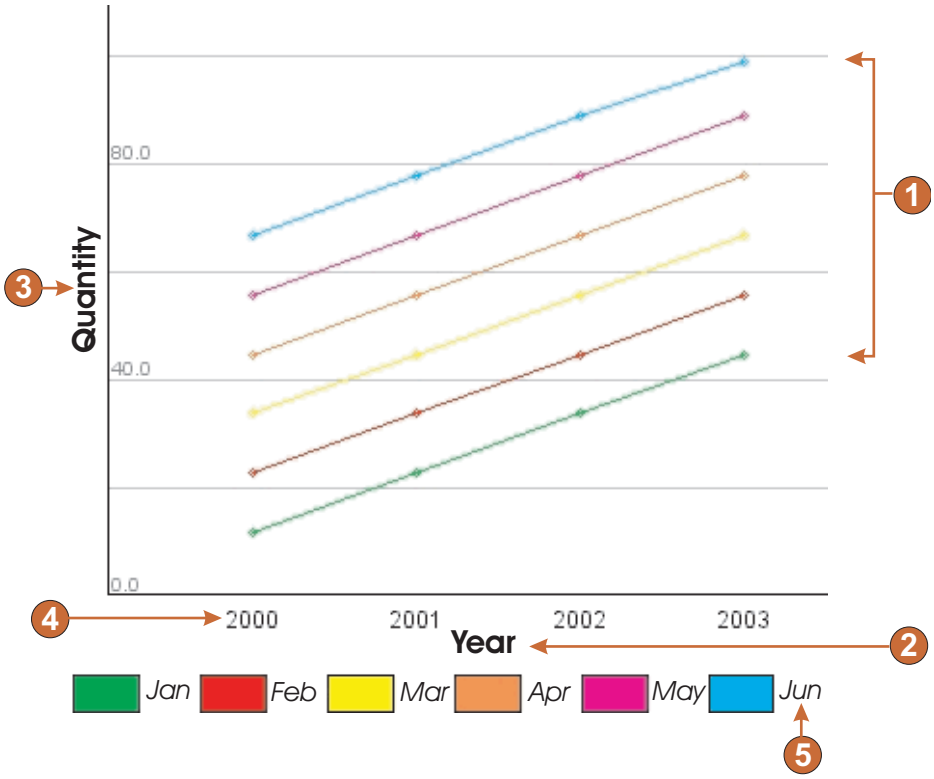



*Figure 42. Line graph widget example*

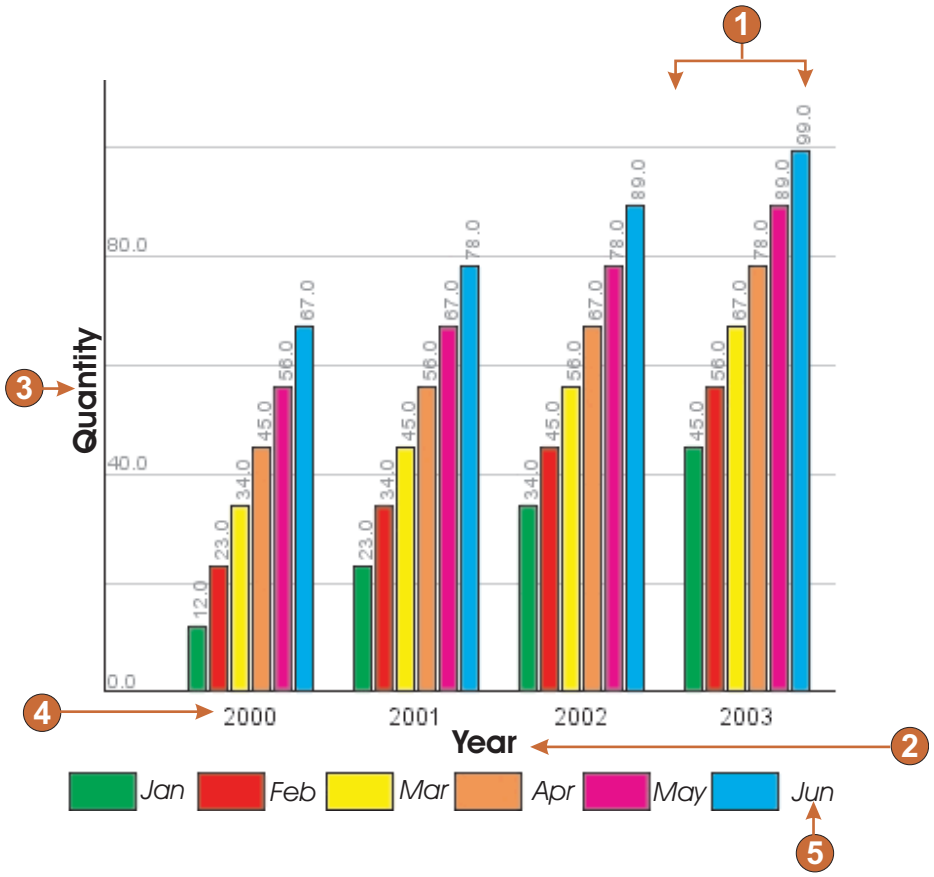



*Figure 43. Vertical bar graph widget example*

1. Data sets (6)
2. X-axis title (Year; Quantity on horizontal bar graph)
3. Y-axis title (Quantity; Year on horizontal bar graph)
4. Data point label (extract)
5. Data set label (extract)

The following settings can be configured for this widget:

**Number of data sets**
Specifies the number of data sets to be depicted in the graph. This setting is only displayed in the **Insert Host Component** wizard.

**Data set source**
Specifies how to obtain values for one or more data sets. Options: **Row** (each row of the table constitutes one set of data to be graphed) or **Column** (each column of the table constitutes one set of data to be graphed).

**Width (pixels)**
Width, in pixels, of the graph.

**Height (pixels)**
Height, in pixels, of the graph.

**Background color**
Specifies the color to use as the background of the graph. Click the color selector button to display a color palette.

**Background image**
Enter the name of the image to use for the background of the graph (relative to the Web project's Web Content\Common\Images folder or the rich client project's Rich Client Content\Images folder). Click the **Browse** button to import an image from your system.

**Default Font**
Specify the font, type and font size.

**X-axis title**
Optional. String specifying the X-axis label of the graph.

**Y-axis title**
Optional. String specifying the Y-axis label of the graph.

**Axis color**
Specifies the color to draw the axes of the graph in. Click the color selector button to display a color palette.

**Label color**
Specifies the color to draw the labels of the graph in. Click the color selector button to display a color palette.

**Text antialiasing**
If selected, text antialiasing will be used to smooth the edges of the drawn text.

**Alternate text**
Optional. Specifies the text that appears if the rendered graph image cannot be loaded by the user's client browser.

As with the Number of data sets settings, the following settings are only displayed in the **Insert Host Component wizard**.

**Extract data point labels**
Select this box if you want labels for the graph's data points to be extracted from a row (or column) of the table.

**Row (or Column)**
Specify the number of the row (or column) from which data point labels should be extracted. The label for this entry field (Row or Column) is dependent upon the value specified for the **Data set source** setting.

**Extract data set labels (for legend)**
Select this box if you want labels for the graph's data sets to be extracted from a row (or column) of the table.

**Row (or Column)**
Specify the number of the row (or column) from which data set labels should be extracted. The label for this entry field (Row or Column) is dependent upon the value specified for the **Data set source** setting.

**Data sets**
Click this button to display the Data Source Settings dialog, which enables you to specify the following additional settings for the data sources:

**Data set 'n', 'row' or 'column'**
The number ('n') of these fields matches the value specified in the Data set source setting. Enter the number of any row or column of data you want to use for the data set. This enables you to reorder or duplicate sets of data in the graph. The last part of the label for this entry field is dependent upon the value specified for the Data set source setting. The entry field label matches the value specified for the Data set source setting.

**color** There is a color button for each of the Data set 'n', 'row' or 'column' settings. The buttons show the color to use for the data set in the graph. Click the button to display a color palette if you want to change the data set color.

**legend label**
Supply the label to use for the legend. Only enabled if **Extract data set labels (for legend)** is not selected.

---

## Label

The Label widget is responsible for rendering text.

This widget renders data supplied by the following components.

- Text
- HTML DDS keyword `Web-only`

The following figure shows how a label widget appears on a transformation, using the data from the Text component example as input:

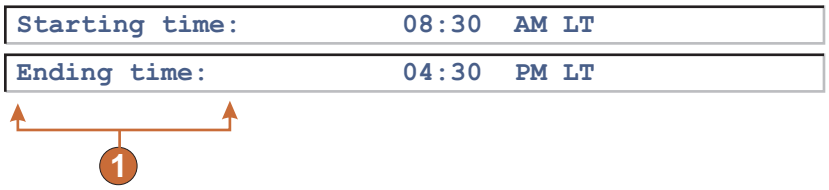



*Figure 44. Label widget example*

1. Label

The following settings can be configured for this widget:

**Trim spaces**
If selected, white space (extra spaces) is removed from both ends of the selected text.

**Style class** `Web-only`
Optional. The CSS style class associated with the generated text. The default value is **HATSCAPTION**. See "Using style sheets" on page 317 for more information.

**Style** `Web-only`
Optional.Use the launcher button next to this field to open a style properties dialog. This dialog frees you from the need to understand CSS to change the font, color, or other style settings for the widget. See "Using style sheets" on page 317 for more information.

**Override font** `RCP-only`
Select this option and use the supplied **Font** setting controls to override the default font name, font style, and font size.

**Override color** `RCP-only`
Select this option and use the supplied **Foreground color** and **Background color** setting controls to override foreground and background colors.

---

## Link

The Link widget is responsible for rendering HTML links.

This widget renders data supplied by the following components:

- Function key
- Light pen (attention)
- Selection list
- URL
- Selection field (ENPTUI)

The following figure shows how a link widget appears on a transformation, using the data from the Selection list component example as input:

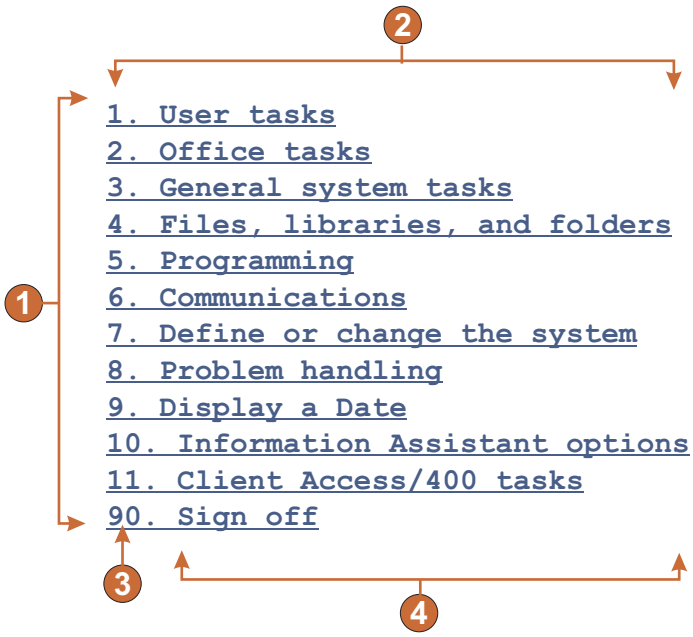



*Figure 45. Link widget example*

1. Rows
2. Number of columns per row (1)
3. Leading token (show both)
4. Description (show both)

The following settings can be configured for this widget:

**Caption type**
Specifies how the caption for each link is determined. The value of the

leading token and the description are derived from the component; you can select what appears as the text for the link. For example, if the host screen had a menu item that read **4. Mail**, you can have the text display **4**, or **Mail**, or **4. Mail**.

**Trim spaces on caption**
If selected, white space (extra space) is trimmed from both ends of the caption.

**Layout**
Specifies how the links will be arranged on the HTML page. Options: **Table**, **Separated**.

**Notes:**

1. This setting is not applicable in default rendering if the function key component is being used.
2. The widget preview does not always match the full page preview. This happens because the widget preview simply renders the component's output. It does not try to preserve the alignment of the screen when it renders, as is done in the full page preview.

**Number of columns per row**
The number of links to display horizontally before wrapping to the next line.

**Separator**
Specifies what separator you want to use to separate the rendered links on the HTML page. You can select from the drop-down list or type in your own in the entry field.

**Target** `Web-only`
Select from the drop-down list the target where you want to display the contents of the linked URL; the same, new, parent, or top window.

**Link style class** `Web-only`
Optional. The CSS style class associated with each generated link. The value of the class attribute of the HTML link tag will be set to this value. The default value is **HATSLINK**. See "Using style sheets" on page 317 for more information.

**Style** `Web-only`
Optional. Use the launcher button next to this field to open a style properties dialog. This dialog frees you from the need to understand CSS to change the font, color, or other style settings for the widget. See "Using style sheets" on page 317 for more information.

**Override font** `RCP-only`
Select this option and use the supplied **Font** setting controls to override the default font name, font style, and font size.

**Override color** `RCP-only`
Select this option and use the supplied **Foreground color** and **Background color** setting controls to override foreground and background colors.

---

## Link (item selection)

The link (item selection) widget is responsible for rendering the item selection component as links.

The following figure shows how a link (item selection) widget appears on a transformation, using the data from the Item selection component example as input:

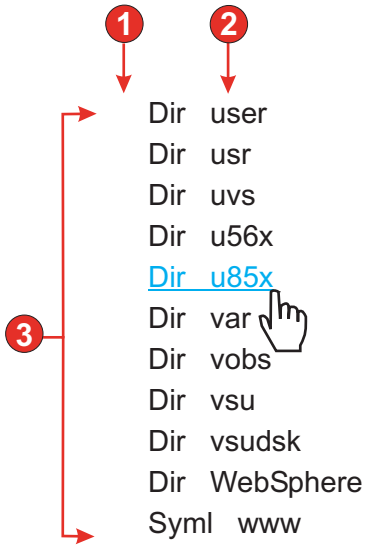



*Figure 46. Link (item selection) widget example*

1. Show input field (no)
2. Number of columns per row (defaults to 1)
3. Layout (table)

The following setting can be configured for this widget:

**Layout**
Specifies how the links will be arranged on the HTML page. Options: **Table**, **Separated**.

**Number of columns per row**
Specifies how many instances of this widget should appear on each row of the rendered HTML page.

**Separator**
Specifies what separator you want to use to separate the rendered links on the HTML page. You can select from the drop-down list or type in your own in the entry field.

**Show input field**
Specifies whether to display an input field for each link, and the location of the input field (to the left/right of the links). Options: **No**, **Left of captions**, **Right of captions**.

**Selection value**
Specifies characters to place to the left of the input field.

**Key to automatically send**
Specifies which AID key to send to the host, or none.

**Link style class**
Optional. The CSS style class associated with the generated launcher link. The value of the class attribute of the HTML link tag will be set to this

value. The default value is **HATSPROTLINK HBLANK**. See "Using style sheets" on page 317 for more information.

**Style** Optional.Use the launcher button next to this field to open a style properties dialog. This dialog frees you from the need to understand CSS to change the font, color, or other style settings for the widget. See "Using style sheets" on page 317 for more information.

---

## List

The List widget is responsible for rendering list boxes.

This widget renders data supplied by the following components:

- Command line
- Input field
- Input field with hints
- Item selection

The following figure shows how a list widget appears on a transformation.

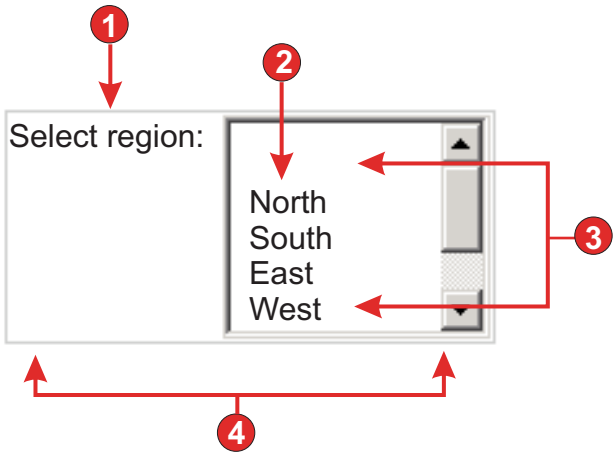



*Figure 47. List widget example*

1. Caption (custom)
2. Fill (from string)
3. Number of displayable rows (defaults to 5)
4. Number of columns per row (defaults to 1)

The following settings can be configured for this widget:

**Fill from global variable**
If selected, fill the list from the specified global variables.

**Global variable containing list values**
Specifies the name of the indexed global variable containing the set of values. An item for each index in the global variable will be created in the list.

**Shared**
HATS local and shared global variables can have the same name.

Select this box if you want to use the shared global variable to populate the list items. When this box is cleared, the local global variable is used.

**Global variable containing list captions**
Optional. Specifies the name of the indexed global variable containing the set of captions. The size of the global variable specified by this value should be greater than or equal to the size specified in the preceding setting. The indexes in this indexed global variable should also match up with the indexes in the values global variable (so that the actual value and caption shown to the user are in sync). If this value is not specified, the caption for each item in the list will be its value.

**Fill from string**
If selected, fill the list from the specified string.

**List items**
Optional. Specifies the string of items to include in the list. Items should be separated with a semicolon (;). To have the list item caption be different than the list item value, enter both separated by an equal sign (=). For example, a value of **Apple=A;Grape=G** renders a list with two items: **Apple** and **Grape**. Selecting the first item causes an **A** to be inserted in the associated host screen input field.

If you want both the item in the list and the value inserted in the host screen to be the same, you only need to enter the item. For example, **Apple=A;G**. In this example, a **G** appears in the list and in the host screen input field.

**Fill from hints**
If selected, fill the list from hints recognized by the component (only applicable when the Input Field with Hints component is used).

**Caption source**
Specifies how the caption for the generated list is determined. Options: **From component** (use the extracted caption from the component) or **Custom** (use the specified caption).

**Custom caption**
Optional. Specifies the caption for the list.

**Note:** Leave this setting blank to not generate a caption for the list.

**Number of displayable rows**
The number of rows in the list box that are visible to the user. If more than this number of items are in the list, scroll bars appear so the user can scroll through the entire list. The default is 5.

**Number of columns per row**
The number of lists to display horizontally before wrapping to the next line. The default is 1.

**Notes:**

1. This setting is not applicable in default rendering.
2. The widget preview does not always match the full page preview. This happens because the widget preview simply renders the component's output. It does not try to preserve the alignment of the screen when it renders, as is done in the full page preview.

**Auto submit on select**
If selected, once a selection is made in the list, it is submitted.

**Show submit button**
If selected, a submit button is rendered next to the list.

**Submit button caption**
Optional. Specifies the caption of the submit button.

**List style class** `Web-only`
Optional. The CSS style class associated with the generated list. The value of the class attribute of the HTML list tag will be set to this value. The default value is **HATSDROPDOWN**. See "Using style sheets" on page 317 for more information.

**List option style class** `Web-only`
Optional. The CSS style class associated with each option in the list. The default value is **HATSOPTION**. See "Using style sheets" on page 317 for more information.

**Caption style class** `Web-only`
Optional. The CSS style class associated with the generated list's caption. The default value is **HATSCAPTION**. See "Using style sheets" on page 317 for more information.

**Table style class** `Web-only`
Optional. The CSS style class associated with the generated table. If more than one list is rendered, an HTML table will be generated to enclose these lists. The value of the class attribute of the HTML table tag will be set to this value. The default value is **HATSTABLE**. See "Using style sheets" on page 317 for more information.

**Style** `Web-only`
Optional. Any CSS properties that you want to override. For example, you can specify **font-color: red; font-size: 18pt;** in this field to change the font color and size for this widget. The properties you enter apply to each element of this widget. Use the launcher button next to this field to open a style properties dialog. This dialog frees you from the need to understand CSS to change the font, color, or other style settings for the widget. See "Using style sheets" on page 317 for more information.

**Override font** `RCP-only`
Select this option and use the supplied **Font** setting controls to override the default font name, font style, and font size.

**Override color** `RCP-only`
Select this option and use the supplied **Foreground color** and **Background color** setting controls to override foreground and background colors.

---

## Popup

The Popup widget is responsible for rendering an input field with an associated launcher button, link, or image. This launcher opens a popup which your user can use to select a valid value for the associated input field.

This widget renders data supplied by the following components:

- Command line
- Input field

- Input field with hints
- Item selection

**Notes:**

1. To display the popup widget image launcher when the HATS application is accessed through a proxy server, you must configure the HATS application to use the proxy server. For instructions see "Configuring HATS applications to use a proxy server" on page 35.
2. For DBCS considerations when using this widget see "SBCS eliminate maximum length" on page 462.

The following figure shows how a popup widget appears on a transformation, using the data from the Input field with hints component example as input:

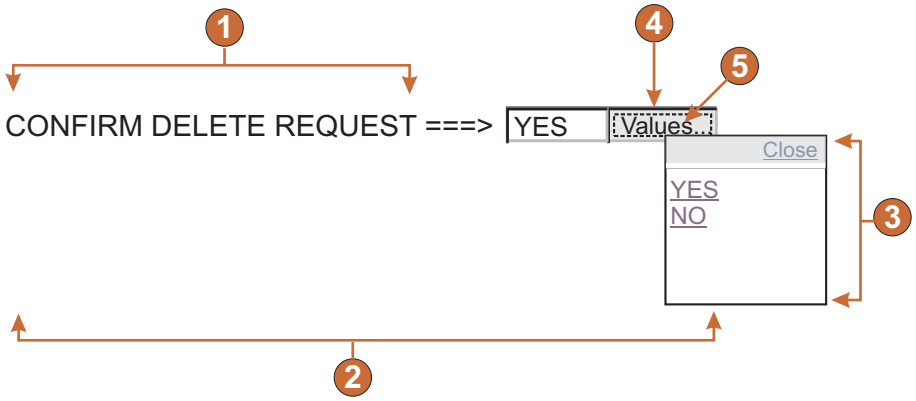



*Figure 48. Popup widget example*

1. Caption (from component)
2. Number of columns per row (1)
3. Fill (from hints)
4. Launcher type (button)
5. Launcher caption (Values)

The following settings can be configured for this widget:

**Fill from global variable**
If selected, fill the popup list from the specified global variables.

**Global variable containing list values**
Specifies the name of the indexed global variable containing the set of values. A link for each index in the global variable will be created in the popup window.

**Shared**
HATS local and shared global variables can have the same name. Select this box if you want to use the shared global variable to populate the list items. When this box is cleared, the local global variable is used.

**Global variable containing list captions**
Optional. Specifies the name of the indexed global variable containing the set of captions. The size of the global variable specified by this value should be greater than or equal to the size specified in the preceding setting. The indexes in this indexed

global variable should also match up with the indexes in the values global variable (so that the actual value and caption shown to the user are in sync). If this value is not specified, the text for each link will be its value.

**Fill from string**
If selected, fill the popup list from the specified string.

**List items**
Optional. Specifies the string of items to include in the popup window. Items should be separated with a semicolon (;). To have the list item caption be different than the list item value, enter both separated by an equal sign (=). For example, a value of **Apple=A;Grape=G** renders a popup window with two items: **Apple** and **Grape**. Selecting the first item causes an **A** to be inserted in the associated host screen input field.

If you want both the item in the popup window and the value inserted in the host screen to be the same, you only need to enter the item. For example, **Apple=A;G**. In this example, a **G** appears in the popup window and in the host screen input field.

**Fill from hints**
If selected, populate the popup from hints recognized by the component (only applicable when the **Input Field with Hints** component is used).

**Caption source**
Specifies how the caption for the generated input field is determined. Options: **From component** (use the extracted caption from the component) or **Custom** (use the specified caption).

**Custom caption**
Optional. Specifies the caption for the input field. Note: Leave this setting blank to not generate a caption for the associated input field.

**Number of columns per row**
The number of input fields to display horizontally before wrapping to the next line.

**Notes:**

1. This setting is not applicable in default rendering.
2. The widget preview does not always match the full page preview. This happens because the widget preview simply renders the component's output. It does not try to preserve the alignment of the screen when it renders, as is done in the full page preview.

**Enable cursor positioning option on input fields** `Web-only`
Select this box to allow the user to switch from data input mode to cursor positioning mode for input fields. When in data input mode, the user can enter data into the input field. When in cursor positioning mode, the user can tab to or otherwise position the cursor on any character in the input field. This setting is useful for Web projects that are optimized for mobile devices that do not have other cursor positioning capabilities. For more information about other settings for this option that do not appear in the HATS Toolkit GUI, see “Considerations and limitations for mobile devices” on page 44.

**Read only**
If selected, rendered input fields are read only. A read only input field

appears as a regular input field, but does not allow the user to modify its contents. This is useful in cases where you want to display the contents of a non-protected field to a user, but you do not want the user to modify the contents. In the case of the Popup widget, it enables you to control the input supplied into the field. Only values available in the popup can be supplied to the field.

**Enable foreground colors**
If selected, host screen foreground colors are rendered.

For Web applications, colors are mapped to CSS stylesheet classes representing that color. For example, if a host screen field is marked as **RED**, the Field widget will enclose the generated HTML for that field in a tag whose class name attribute is set to **HRED**. This allows for you to remap host screen colors on your generated Web page.

**Note:** The blacktheme.css directly maps host screen field foreground colors (for example, red text on the host screen will appear as red text on the generated Web page). However, other stylesheet files like monochrome.css, map field colors differently in an attempt to create a consistent, modern style.

For rich client applications, colors are mapped by the rich client template.

**Note:** This setting is overridden by the **Override color > Foreground color** setting.

**Enable extended attributes**
If selected, extended field attributes (blink, reverse video, underline, and column separator) are rendered. Also, for 3270 Web applications, extended field colors are mapped (see the preceding setting for more information). See "Using style sheets" on page 317 for more information.

For rich client applications, extended field attributes are mapped by the rich client template.

**Blink style** `Web-only`
The CSS style setting you want to use to render blinking text from the host screen.

**Reverse video style** `Web-only`
The CSS style setting you want to use to render reverse video text from the host screen.

**Underline style** `Web-only`
The CSS style setting you want to use to render underlined text from the host screen.

**Column separator style** `Web-only`
The CSS style setting you want to use to render column separators from the host screen.

**Launcher type**
Specifies the style of the launcher. The launcher is used to open the popup. Options **Button**, **Link**, or **Image**.

**Image filename**
The name of the image file. This is only enabled if launcher type is an Image. Click **Import** to import an image into your project. A preview window displays the currently selected image.

**Caption**
The caption of the launcher button or link. This is only enabled if launcher type is either Link or Button.

**Link style class** `Web-only`
Optional. The CSS style class associated with the generated launcher link. The value of the class attribute of the HTML link tag will be set to this value. The default value is **HATSLINK**. See "Using style sheets" on page 317 for more information.

**Button style class** `Web-only`
Optional. The CSS style class associated with the generated launcher button. The value of the class attribute of the HTML button tag will be set to this value. The default value is **HATSBUTTON**. See "Using style sheets" on page 317 for more information.

**Close link caption** `Web-only`
The caption to appear for the Close action.

**Input field style class** `Web-only`
Optional. The CSS style class associated with the generated input field. The value of the class attribute of the HTML input tag will be set to this value. The default value is **HATSINPUT**. See "Using style sheets" on page 317 for more information.

**Caption style class** `Web-only`
Optional. The CSS style class associated with the generated input field's caption. The default value is **HATSCAPTION**. See "Using style sheets" on page 317 for more information.

**Table style class** `Web-only`
Optional. The CSS style class associated with the generated table. If more than one set of input fields is rendered, an HTML table will be generated to enclose these input fields. The value of the class attribute of the HTML table tag will be set to this value. The default value is **HATSTABLE**. See "Using style sheets" on page 317 for more information.

**Popup window style class** `Web-only`
Optional. The CSS style class associated with the generated popup window. The default value is **HATSPOPUP**. See "Using style sheets" on page 317 for more information.

**Close link style class** `Web-only`
Optional. The CSS style class associated with the link used to close the popup window. The default value is **HATSLINK**. See "Using style sheets" on page 317 for more information.

**Item link style class** `Web-only`
Optional. The CSS style class associated with the links used to select an item from the popup. The default value is **HATSPOPUPITEMLINK**. See "Using style sheets" on page 317 for more information.

**Style** `Web-only`
Optional.Use the launcher button next to this field to open a style properties dialog. This dialog frees you from the need to understand CSS to change the font, color, or other style settings for the widget. See "Using style sheets" on page 317 for more information.

**Note:** There are times when the popup widget may appear too translucent. If you want to make the widget more opaque, you must modify the cascading style sheet (CSS) that your template is using by doing the following steps:

1. Open the CSS your template is using.
2. Under the **DIV.HATSPOPUP** class, search for: filter: **alpha (opacity=40, style=0)**
3. Changing the opacity value will adjust the transparency of the popup widget. Where **0** is transparent, and **100** is completely opaque.

**Override font** `RCP-only`
Select this option and use the supplied **Font** setting controls to override the default font name, font style, and font size.

**Override color** `RCP-only`
Select this option and use the supplied **Foreground color** and **Background color** setting controls to override foreground and background colors.

---

## Radio button (data entry)

The Radio button (data entry) widget is responsible for rendering a set of radio buttons.

This widget renders data supplied by the following components:

- Command line
- Input field
- Input field with hints

The following figure shows how a data entry radio button widget appears on a transformation, using the data from the Input field with hints component example as input:

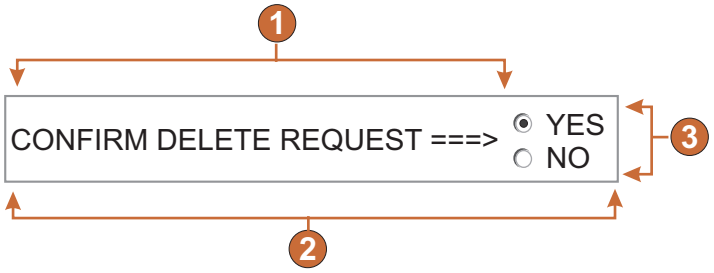



*Figure 49. Data entry radio button widget example*

1. Caption (from component)
2. Number of columns per row (1)
3. Fill (from hints)

The following settings can be configured for this widget:

**Fill from global variable**
If selected, populate the set of radio buttons from the specified global variables.

**Global variable containing list values**
Specifies the name of the indexed global variable containing the set of values. A radio button for each index in the global variable will be created.

**Shared**
HATS local and shared global variables can have the same name. Select this box if you want to use the shared global variable to populate the list items. When this box is cleared, the local global variable is used.

**Global variable containing list captions**
Optional. Specifies the name of the indexed global variable containing the set of captions. The size of the global variable specified by this value should be greater than or equal to the size specified in the preceding setting. The indexes in this indexed global variable should also match up with the indexes in the values global variable (so that the actual value and caption shown to the user are in sync). If this value is not specified, the caption for each radio button will be its value.

**Fill from string**
If selected, populate the set of radio buttons from the specified string.

**List items**
Optional. Specifies the string of items to include in the set of radio buttons. Items should be separated with a **semicolon (;)**. To have the list item caption be different than the list item value, enter both separated by an equal sign (=). For example, a value of **Apple=A;Grape=G** renders radio buttons for two items: **Apple** and **Grape**. Selecting the first item causes an **A** to be inserted in the associated host screen input field.

If you want both the item in the radio button set and the value inserted in the host screen to be the same, you only need to enter the item. For example, **Apple=A;G**. In this example, a **G** appears next to the radio button and in the host screen input field.

**Fill from hints**
If selected, populate the set of radio buttons from hints recognized by the component (only applicable when the Input Field with Hints component is used).

**Caption source**
Specifies how the caption for the generated set of radio buttons is determined. Options: **From component** (use the extracted caption from the component) or **Custom** (use the specified caption).

**Custom caption**
Optional. Specifies the caption for the set.

**Note:** Leave this setting blank to not generate a caption for the set.

**Trim spaces on caption**
If selected, white space (extra space) is trimmed from both ends of the caption.

**Number of columns per row**
The number of radio buttons to display horizontally before wrapping to the next line.

**Label style class** `Web-only`
Optional. The CSS style class associated with the generated caption for the set of radio buttons. The default value is **HATSCAPTION**. See "Using style sheets" on page 317 for more information.

**Radio button style class** `Web-only`
Optional. The CSS style class associated with each radio button. The default value is **HATSRADIOBUTTON**. See "Using style sheets" on page 317 for more information.

**Radio button description style class** `Web-only`
Optional. The CSS style class associated with each radio button description. The default value is **HATSTEXT**. See "Using style sheets" on page 317 for more information.

**Table style class** `Web-only`
Optional. The CSS style class associated with the generated table. If more than one radio button is rendered, an HTML table will be generated to enclose them. The value of the class attribute of the HTML table tag will be set to this value. The default value is **HATSTABLE**. See "Using style sheets" on page 317 for more information.

**Style** `Web-only`
Optional.Use the launcher button next to this field to open a style properties dialog. This dialog frees you from the need to understand CSS to change the font, color, or other style settings for the widget. See "Using style sheets" on page 317 for more information.

**Override font** `RCP-only`
Select this option and use the supplied **Font** setting controls to override the default font name, font style, and font size.

---

## Radio button (item selection)

The Radio button (item selection) widget is responsible for rendering a set of radio buttons.

This widget renders data supplied by the following component:

- Item selection

The following figure shows how a radio button (item selection) widget appears on a transformation, using the data from the Item selection component example as input:

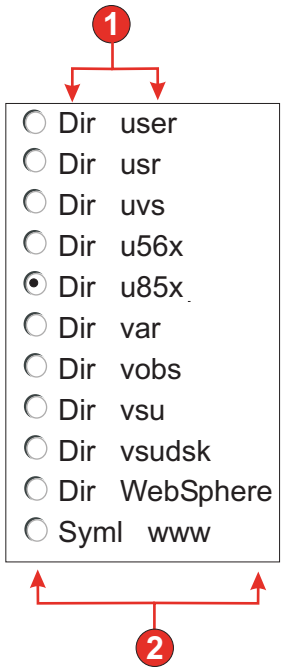



*Figure 50. Radio button (item selection) widget example*

1. Caption (from component)
2. Number of columns per row (defaults to 1)

The following settings can be configured for this widget:

**Caption**
Specifies how the caption for the generated input field is determined. Options: **From component** (use the extracted caption from the component) or **Custom** (use the specified caption).

**Custom caption**
Optional. Specifies the caption for the input field.

**Note:** Leave this setting blank to not generate a caption for the check box.

**Trim spaces on caption**
If selected, white space (extra space) is trimmed from both ends of the caption.

**Number of columns per row**
The number of input fields to display horizontally before wrapping to the next line.

**Selection value**
The value to be entered in the input field of the host screen when the value is selected. Options: **/**, **S**, or **1**.

**Auto submit on select**
If selected, once a radio button is selected, the item selection is submitted. If the radio button was generated from a function key, the selected item's key is sent to the host.

**Show submit button**
If selected, a submit button is rendered below the set of radio buttons. Does not apply when using this widget with the function key component.

**Submit button caption**
Optional. Specifies the caption of the submit button.

**Button style class** `Web-only`
Optional. The CSS style class associated with the generated submit button. The value of the class attribute of the HTML button tag will be set to this value. The default value is **HATSBUTTON**. See "Using style sheets" on page 317 for more information.

**Radio button style class** `Web-only`
Optional. The CSS style class associated with each radio button. The default value is **HATSRADIOBUTTON**. See "Using style sheets" on page 317 for more information.

**Radio button description style class** `Web-only`
Optional. The CSS style class associated with each radio button description. The default value is **HATSTEXT**. See "Using style sheets" on page 317 for more information.

**Label style class** `Web-only`
Optional. The CSS style class associated with the generated caption for the set of radio buttons. The default value is **HATSCAPTION**. See "Using style sheets" on page 317 for more information.

**Table style class** `Web-only`
Optional. The CSS style class associated with the generated table. If more than one radio button is rendered, an HTML table will be generated to enclose them. The value of the class attribute of the HTML table tag will be set to this value. The default value is **HATSTABLE**. See "Using style sheets" on page 317 for more information.

**Style** `Web-only`
Optional.Use the launcher button next to this field to open a style properties dialog. This dialog frees you from the need to understand CSS to change the font, color, or other style settings for the widget. See "Using style sheets" on page 317 for more information.

**Override font** `RCP-only`
Select this option and use the supplied **Font** setting controls to override the default font name, font style, and font size.

---

## Radio button (selection)

The Radio button (selection) widget is responsible for rendering a set of radio buttons.

This widget renders data supplied by the following components:

- Function key
- Light pen (attention)
- Selection list
- Selection field (ENPTUI)

The following figure shows how a selection list radio button widget appears on a transformation, using the data from the Selection list component example as input:

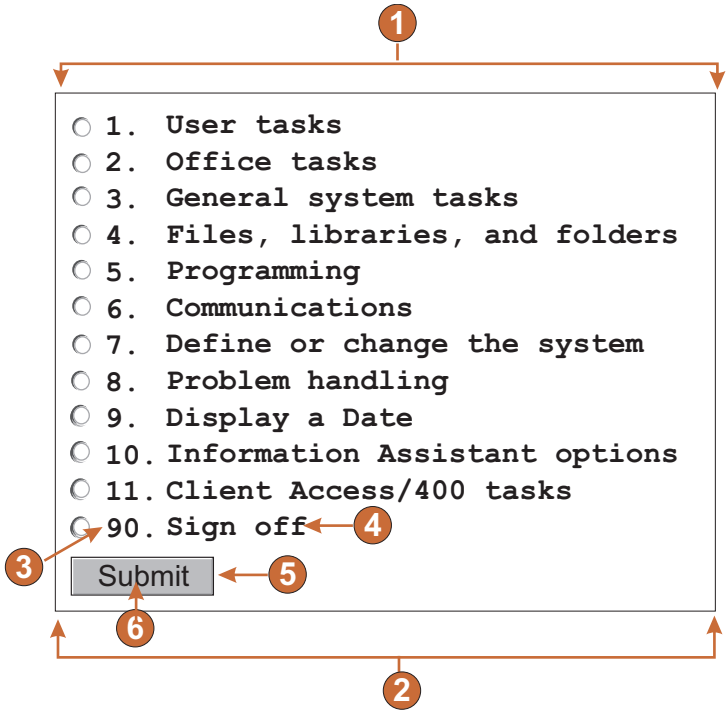



*Figure 51. Selection list radio button widget example*

1. Caption type (show both)
2. Number of columns per row (1)
3. Leading token (show both)
4. Description (show both)
5. Submit button (show submit button)
6. Submit button caption (submit)

The following settings can be configured for this widget:

**Caption type**
Specifies how the caption for each button is determined. The value of the leading token and the description are derived from the component; you can select what appears on the button caption. For example, if the host screen had a menu item that read **4. Mail**, you can have the caption display **4**, or **Mail**, or **4. Mail**.

**Trim spaces on caption**
If selected, white space (extra space) is trimmed from both ends of the caption.

**Number of columns per row**
The number of radio buttons to display horizontally before wrapping to the next line.

**Auto submit on select**
If selected, once a radio button is selected, the selection is submitted. If the radio button was generated from a function key, the selected item's key is

sent to the host. If the radio button was generated from a selection list, the selected item's value is inserted into the target field and the specified action key is sent to the host.

**Show submit button**
If selected, a submit button is rendered below the set of radio buttons. Does not apply when using this widget with the function key component.

**Submit button caption**
Optional. Specifies the caption of the submit button.

**Button style class** `Web-only`
Optional. The CSS style class associated with the generated submit button. The value of the class attribute of the HTML button tag will be set to this value. The default value is **HATSBUTTON**. See "Using style sheets" on page 317 for more information.

**Radio button style class** `Web-only`
Optional. The CSS style class associated with each radio button. The default value is **HATSRADIOBUTTON**. See "Using style sheets" on page 317 for more information.

**Radio button description style class** `Web-only`
Optional. The CSS style class associated with each radio button description. The default value is **HATSTEXT**. See "Using style sheets" on page 317 for more information.

**Label style class** `Web-only`
Optional. The CSS style class associated with the generated caption for the set of radio buttons. The default value is **HATSCAPTION**. See "Using style sheets" on page 317 for more information.

**Table style class** `Web-only`
Optional. The CSS style class associated with the generated table. If more than one radio button is rendered, an HTML table will be generated to enclose them. The value of the class attribute of the HTML table tag will be set to this value. The default value is **HATSTABLE**. See "Using style sheets" on page 317 for more information.

**Style** `Web-only`
Optional.Use the launcher button next to this field to open a style properties dialog. This dialog frees you from the need to understand CSS to change the font, color, or other style settings for the widget. See "Using style sheets" on page 317 for more information.

**Override font** `RCP-only`
Select this option and use the supplied **Font** setting controls to override the default font name, font style, and font size.

---

## Scrollbar (ENPTUI)

The scrollbar (ENPTUI) widget renders data supplied from the Scrollbar field (ENPTUI) component.

The following figure shows how a Scrollbar (ENPTUI) widget appears on a transformation, using the data from the Scrollbar field (ENPTUI) component example as input:

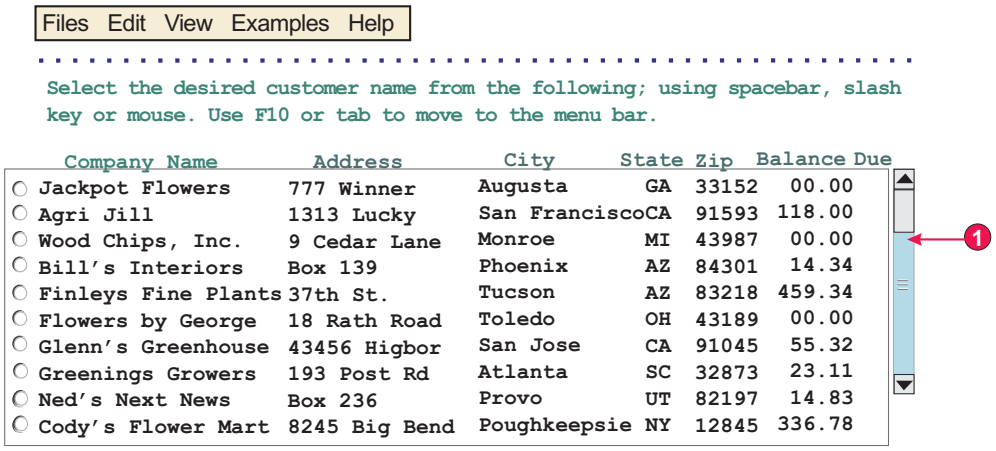



*Figure 52. Scrollbar (ENPTUI) widget example*

1. Scrollbar

The following settings can be configured for this widget:

**Scrollbar style class** `Web-only`
: Optional. The CSS style class associated with the scrollbar appearance. The default value is **HATSESCROLL**. See "Using style sheets" on page 317 for more information.

**Scrollbar button style class** `Web-only`
: Optional. The CSS style class associated with the scrollbar buttons. The default value is **HATSESCROLLBUTTON**. See "Using style sheets" on page 317 for more information.

**Button arrow color** `Web-only`
: The color of the button arrow. The default value is **Black**.

**Style** `Web-only`
: Optional. Any CSS properties that you want to override. For example, you can specify **font-color: red; font-size: 18pt;** in this field to change the font color and size for this widget. The properties you enter apply to each element of this widget. Use the launcher button next to this field to open a style properties dialog. This dialog frees you from the need to understand CSS to change the font, color, or other style settings for the widget. See "Using style sheets" on page 317 for more information.

**Override color** `RCP-only`
: Select this option and use the supplied **Background color** setting control to override background color.

---

## Subfile (check box)

The subfile (check box) widget is responsible for rendering a 5250 subfile.

This widget renders data supplied by the subfile component.

**Note:** For DBCS considerations when using this widget see "SBCS eliminate maximum length" on page 462.

The following figure shows how a subfile (check box) widget appears on a transformation, using the data from the Subfile component example as input:

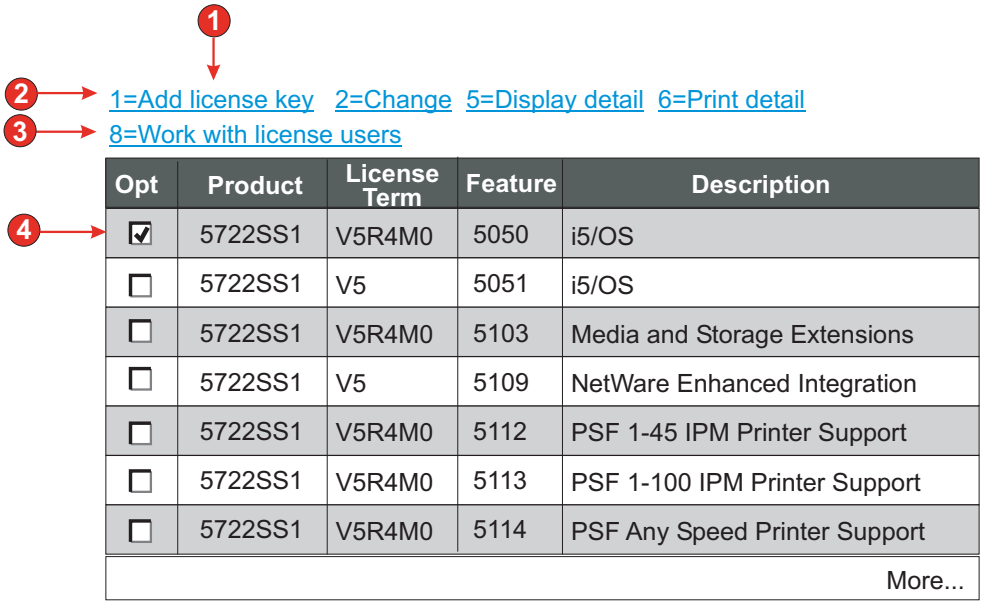



*Figure 53. Subfile (check box) widget example*

1. Caption type (show both)
2. Actions type (link)
3. Actions location (above subfile)
4. Check box

The following settings can be configured for this widget:

**Caption type:**
Specifies how the caption for each check box is determined. The value of the leading token and the description are derived from the component; you can select what appears as the caption. For example, if the host screen had an action that read **5. Display**, you can have the caption display **5**, or **Display**, or **5. Display**.

**Actions type**
Specifies whether to use a **Button** or a **Link** to trigger the actions.

**Actions location**
Specifies the location of the buttons or links that trigger the actions. Options are **Above and below subfile**, **Above subfile**, and **Below subfile**.

**Rows per record**

**Show alternating row colors** `RCP-only`
Select this box to render rows in alternating colors. RGB color values are determined by the ITableColorProvider object returned by your template. See the HATS RCP API Reference section of the HATS Information Center at http://www.ibm.com/support/knowledgecenter/SSXKAY_9.7.0?topic=/com.ibm.hats.doc/doc/rcpjavadoc/index.html for more information about the RcpTemplate and ITableColorProvider classes.

**Number of rows**
Specify the fixed number of rows per record in your subfiles.

**Columns placement** `Web-only`

This widget can be configured to display primary columns of data, and optionally to have a details section to view additional detail columns. This setting is useful when displaying table data on a mobile device. For more information about these settings and about other settings for this option that do not appear in the HATS Toolkit GUI, see "Considerations and limitations for mobile devices" on page 44.

**Primary columns**
Use this field to specify which columns to display, and in what order. The field allows for numeric any-order listing of column numbers with expansion shorthand, and columns are allowed to be displayed more than once or not at all, for example, **1,3,2,5-7,2-3,9**.

A hyphen (-) between two numbers indicates a range of columns. You can use the asterisk character (*) to indicate the last available column. An asterisk alone means columns 1 through n where n is the last available column. You can also use 5* as a shorthand for 5-*. These mean the same thing: 1-*, *.

For Web projects, this setting defaults to *****, or all columns. For Web projects optimized for mobile devices, this setting defaults to **1-2**, or columns 1 and 2.

**Note:** The numbers represent columns provided to the widget by the component. They do not represent the columns originally recognized by the component, which may or may not be excluded, before providing them to the widget.

**Detail columns**
Use this optional field to specify which detail columns to display when requested, and in what order. The field allows for numeric any-order listing with expansion shorthand.

For Web projects, this setting defaults to empty, or no detail columns section. For Web projects optimized for mobile devices, this setting defaults to **3***, or column 3 through the last column.

**Keep detail columns on the server**
Select this option to keep the detail columns section on the server until requested by the user. This reduces the amount of HTML output at initial rendering. The default is cleared.

**Highlighted rows** `RCP-only`
The comma-separated or ranged set of rows to be highlighted when rendered. This setting is useful for highlighting important rows in the generated table. The rows represented by this value are based on the recognized table, not on the actual host screen. Example values: **1,2,3....**, **1,2-4**, **3-5**. RGB color values are determined by the ITableColorProvider object returned by your template. See the HATS RCP API Reference section of the HATS Information Center at http://www.ibm.com/support/knowledgecenter/SSXKAY_9.7.0?topic=/com.ibm.hats.doc/doc/rcpjavadoc/index.html for more information about the RcpTemplate and ITableColorProvider classes.

**Highlighted columns** `RCP-only`
The comma-separated or ranged set of columns to be highlighted when

rendered. This setting is useful for highlighting important columns in the generated table. The columns represented by this value are based on the recognized table, not on the actual host screen. Example values: **1,2,3....**, **1,2-4**, **3-5**. RGB color values are determined by the ITableColorProvider object returned by your template. See the HATS RCP API Reference section of the HATS Information Center at http://www.ibm.com/support/knowledgecenter/SSXKAY_9.7.0?topic=/com.ibm.hats.doc/doc/rcpjavadoc/index.html for more information about the RcpTemplate and ITableColorProvider classes.

**Show grid lines** `RCP-only`
Select this box to show grid lines in the rendered subfile.

**Column header source** `RCP-only`
Select from the drop-down whether the text for the column headers is supplied by the subfile header component (**From component**) or by you (**Custom**). If supplied by the subfile header component, see the component description for "Header" on page 209 for more information.

**Column headers (comma-separated)** `RCP-only`
If **Column header source** is set to **Custom**, supply here the column headers separated by commas.

**Use classic header style** `RCP-only`

Select this box to display a classic header style. This also displays multi-row headers, if **Column header source** is set to **From component** and multi-row headers are detected by the subfile header component. Clear this box to display default SWT table control headers in a single row only.

**Trim spaces on headers** `RCP-only`
If selected, the text inside of cells of a header row is trimmed of whitespace (extra spaces) from both ends.

**Enable foreground colors**
If selected, host screen foreground colors are rendered.

For Web applications, colors are mapped to CSS stylesheet classes representing that color. For example, if a host screen field is marked as **RED**, the Field widget will enclose the generated HTML for that field in a tag whose class name attribute is set to **HRED**. This allows for you to remap host screen colors on your generated Web page.

**Note:** The blacktheme.css directly maps host screen field foreground colors (for example, red text on the host screen will appear as red text on the generated Web page). However, other stylesheet files like monochrome.css, map field colors differently in an attempt to create a consistent, modern style.

For rich client applications, colors are mapped by the rich client template.

**Note:** This setting is overridden by the **Override color > Foreground color** setting.

**Field style class** `Web-only`
Optional. The CSS style class associated with the generated field. The value of the class attribute of the HTML tag will be set to this value. The default value is **HATSFIELD**. See "Using style sheets" on page 317 for more information.

**Enable extended attributes**
If selected, extended field attributes (blink, reverse video, underline, and column separator) are rendered with the specified style. See "Using style sheets" on page 317 for more information.

**Blink style** `Web-only`
The CSS style setting you want to use to render blinking text from the host screen.

**Reverse video style** `Web-only`
The CSS style setting you want to use to render reverse video text from the host screen.

**Underline style** `Web-only`
The CSS style setting you want to use to render underlined text from the host screen.

**Column separator style** `Web-only`
The CSS style setting you want to use to render column separators from the host screen.

**Strip underscores on input field**
Select this box if you want to strip the underscores from text when it is rendered.

**Trim spaces on input field**
Select this box to trim leading and trailing spaces from the input field.

**Enable cursor positioning option on input fields** `Web-only`
Select this box to allow the user to switch from data input mode to cursor positioning mode for input fields. When in data input mode, the user can enter data into the input field. When in cursor positioning mode, the user can tab to or otherwise position the cursor on any character in the input field. This setting is useful for Web projects that are optimized for mobile devices that do not have other cursor positioning capabilities. For more information about other settings for this option that do not appear in the HATS Toolkit GUI, see "Considerations and limitations for mobile devices" on page 44.

**Provide spreadsheet file**
If selected, a launcher is provided that will retrieve the table data in spreadsheet file format.

**File name prefix**
Specifies a prefix for the spreadsheet file name. The default is **spreadsheet**. The complete file name will be in the form of `FileNamePrefix_TimeStamp`, where the timestamp format is MMDDYY_HHMMSS, for example, spreadsheet_041506_113020.

**File name extension**
The extension name for the type of spreadsheet format. Options are **.csv** and **.xls**. The .csv file are pure text files, with data columns separated by commas. The .xls files are Microsoft Excel Biff3 files. The default is **.csv**.

**Launcher type**
The type of launcher to display to the user. Options are **Link**, **Image**, and **Button**. The default is **Link** (Web projects) or **Button** (rich client projects).

**Launcher caption**
The caption for the launcher to display to the user. The default is **Download** (Web projects) or **Export** (rich client projects).

**Button style class** `Web-only`
The CSS style class associated with the generated launcher button. The value of the class attribute of the HTML button tag will be set to this value. The default value is **HATSBUTTON**. See "Using style sheets" on page 317 for more information.

**Link style class** `Web-only`
Optional. The CSS style class associated with the generated launcher link. The value of the class attribute of the HTML link tag will be set to this value. The default value is **HATSLINK**. See "Using style sheets" on page 317 for more information.

**Link style class** `Web-only`
Optional. The CSS style class associated with the generated action link. The value of the class attribute of the HTML link tag will be set to this value. The default value is **HATSLINK**. See "Using style sheets" on page 317 for more information.

**Button style class** `Web-only`
Optional. The CSS style class associated with the generated action button. The value of the class attribute of the HTML button tag will be set to this value. The default value is **HATSBUTTON**. See "Using style sheets" on page 317 for more information.

**Check box style class** `Web-only`
Optional. The CSS style class associated with the generated check box. The value of the class attribute of the HTML check box tag will be set to this value. The default value is **HATSCHECKBOX**. See "Using style sheets" on page 317 for more information.

**Table style class** `Web-only`
Optional. The CSS style class associated with the generated table. The value of the class attribute of the HTML table tag will be set to this value. The default value is **HATSTABLE**. See "Using style sheets" on page 317 for more information.

**Table cell style class** `Web-only`
Optional. The CSS style class associated with each cell in the generated table. The value of the class attribute of the HTML table cell (td) tag will be set to this value. The default value is **HATSTABLECELL**. See "Using style sheets" on page 317 for more information.

**Header row style class** `Web-only`
Optional. The CSS style class associated with each header row or column in the generated table. The value of the class attribute of the HTML table row or cell tag will be set to this value. The default value is **HATSTABLEHEADER**. See "Using style sheets" on page 317 for more information.

**Odd row style class** `Web-only`
Optional. The CSS style class associated with each odd numbered row in the table. The value of the class attribute of the each HTML table row tag

in an odd row number will be set to this value. The default value is **HATSTABLEODDROW**. See "Using style sheets" on page 317 for more information.

**Even row style class** `Web-only`
Optional. The CSS style class associated with each even numbered row in the table. The value of the class attribute of the each HTML table row tag in an even row number will be set to this value. The default value is **HATSTABLEEVENROW**. See "Using style sheets" on page 317 for more information.

**Style** `Web-only`
Optional.Use the launcher button next to this field to open a style properties dialog. This dialog frees you from the need to understand CSS to change the font, color, or other style settings for the widget. See "Using style sheets" on page 317 for more information.

**Override font** `RCP-only`
Select this option and use the supplied **Font** setting controls to override the default font name, font style, and font size.

**Override color** `RCP-only`
Select this option and use the supplied **Foreground color** and **Background color** setting controls to override foreground and background colors.

**Note:** For rich client projects, the various color settings for this widget will be applied in the following order (from highest priority to lowest priority):

1. Highlighted rows / columns
2. Override color
3. Enable extended attributes / Enable foreground colors
4. Show alternating row colors

**fieldTextAlignment** `Web-only`
For Web projects, if you use a non-monospace font such as Sans Serif, the rendered data does not align properly with spacing problems between words. You can use the source setting, fieldTextAlignment, to correct the alignment of the data. To enable this setting on a project-wide basis, open the source view of the application.hap file, locate the class for the SubfileCheckboxWidget widget, and change the value of the setting to `NO_SEGMENTING` as shown in the following example:

```
<class name="com.ibm.hats.transform.widgets.SubfileCheckboxWidget">
.
   <setting name="fieldTextAlignment" value="NO_SEGMENTING"/>
.
</class>
```

To enable this setting on a component-level basis in a transformation, open the source view for the transformation, locate the HATS:Component tag for the SubfileCheckboxWidget widget, and change the value of the setting to `NO_SEGMENTING` as shown in the following example:

```
...
widget="com.ibm.hats.transform.widgets.SubfileCheckboxWidget"
...
widgetSettings="...|fieldTextAlignment:NO_SEGMENTING|..."
...
```

The values for this setting are `WORD_BY_WORD` and `NO_SEGMENTING`. The default value is **WORD_BY_WORD**.

**Note:** This setting has no effect when you use the Table (visual) recognition option for data recognition in the Subfile component.

---

## Subfile (drop-down)

The subfile (drop-down) widget is responsible for rendering a 5250 subfile.

This widget renders data supplied by the subfile component.

**Note:** For DBCS considerations when using this widget see "SBCS eliminate maximum length" on page 462.

The following figure shows how a subfile (drop-down) widget appears on a transformation, using the data from the Subfile component example as input:

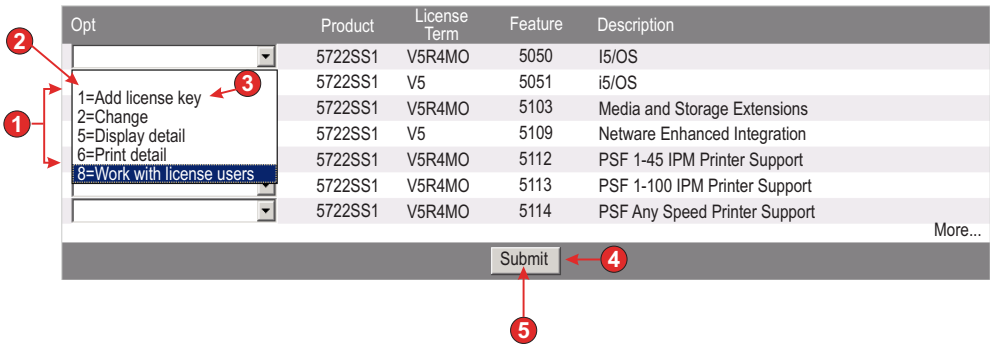



*Figure 54. Subfile (drop-down) widget example*

1. Caption type (show both)
2. Leading token
3. Description
4. Submit button (show submit button)
5. Submit button caption (Submit)

The following settings can be configured for this widget:

**Caption type:**
Specifies how the caption for each item in the drop-down is determined. The value of the leading token and the description are derived from the component; you can select what appears as the caption. For example, if the host screen had an action that read **5. Display**, you can have the caption display **5**, or **Display**, or **5. Display**.

**Show subfile actions**
Select this box if you want to display the descriptive text of the subfile actions, such as **2=Add**, with the rendered subfile in addition to the drop-down list. The check box is not enabled when you select **Show description**.

**Auto submit on select**
If selected, once a selection is made in the drop-down, it is submitted.

**Show submit button**
If selected, a submit button is rendered below the subfile.

**Submit button caption**
Optional. Specifies the caption of the submit button.

**Button style class** `Web-only`
Optional. The CSS style class associated with the generated submit button. The value of the class attribute of the HTML button tag will be set to this value. The default value is **HATSBUTTON**. See "Using style sheets" on page 317 for more information.

**Button row style class** `Web-only`
Optional. The CSS style class associated with the generated table row which contains the submit button. The default value is **HATSTABLEHEADER**. See "Using style sheets" on page 317 for more information.

**Rows per record**

**Show alternating row colors** `RCP-only`
Select this box to render rows in alternating colors. RGB color values are determined by the ITableColorProvider object returned by your template. See the HATS RCP API Reference section of the HATS Information Center at http://www.ibm.com/support/knowledgecenter/SSXKAY_9.7.0?topic=/com.ibm.hats.doc/doc/rcpjavadoc/index.html for more information about the RcpTemplate and ITableColorProvider classes.

**Number of rows**
Specify the fixed number of rows per record in your subfiles.

**Columns placement** `Web-only`

This widget can be configured to display primary columns of data, and optionally to have a details section to view additional detail columns. This setting is useful when displaying table data on a mobile device. For more information about these settings and about other settings for this option that do not appear in the HATS Toolkit GUI, see "Considerations and limitations for mobile devices" on page 44.

**Primary columns**
Use this field to specify which columns to display, and in what order. The field allows for numeric any-order listing of column numbers with expansion shorthand, and columns are allowed to be displayed more than once or not at all, for example, **1,3,2,5-7,2-3,9**.

A hyphen (-) between two numbers indicates a range of columns. You can use the asterisk character (*) to indicate the last available column. An asterisk alone means columns 1 through n where n is the last available column. You can also use 5* as a shorthand for 5-*. These mean the same thing: 1-*, *.

For Web projects, this setting defaults to *, or all columns. For Web projects optimized for mobile devices, this setting defaults to **1-2**, or columns 1 and 2.

**Note:** The numbers represent columns provided to the widget by the component. They do not represent the columns originally recognized by the component, which may or may not be excluded, before providing them to the widget.

**Detail columns**
Use this optional field to specify which detail columns to display

when requested, and in what order. The field allows for numeric any-order listing with expansion shorthand.

For Web projects, this setting defaults to empty, or no detail columns section. For Web projects optimized for mobile devices, this setting defaults to **3***, or column 3 through the last column.

**Keep detail columns on the server**
Select this option to keep the detail columns section on the server until requested by the user. This reduces the amount of HTML output at initial rendering. The default is cleared.

**Highlighted rows** `RCP-only`
The comma-separated or ranged set of rows to be highlighted when rendered. This setting is useful for highlighting important rows in the generated table. The rows represented by this value are based on the recognized table, not on the actual host screen. Example values: **1,2,3....**, **1,2-4**, **3-5**. RGB color values are determined by the ITableColorProvider object returned by your template. See the HATS RCP API Reference section of the HATS Information Center at http://www.ibm.com/support/knowledgecenter/SSXKAY_9.7.0?topic=/com.ibm.hats.doc/doc/rcpjavadoc/index.html for more information about the RcpTemplate and ITableColorProvider classes.

**Highlighted columns** `RCP-only`
The comma-separated or ranged set of columns to be highlighted when rendered. This setting is useful for highlighting important columns in the generated table. The columns represented by this value are based on the recognized table, not on the actual host screen. Example values: **1,2,3....**, **1,2-4**, **3-5**. RGB color values are determined by the ITableColorProvider object returned by your template. See the HATS RCP API Reference section of the HATS Information Center at http://www.ibm.com/support/knowledgecenter/SSXKAY_9.7.0?topic=/com.ibm.hats.doc/doc/rcpjavadoc/index.html for more information about the RcpTemplate and ITableColorProvider classes.

**Show grid lines** `RCP-only`
Select this box to show grid lines in the rendered subfile.

**Column header source** `RCP-only`
Select from the drop-down whether the text for the column headers is supplied by the subfile header component (**From component**) or by you (**Custom**). If supplied by the subfile header component, see the component description for "Header" on page 209 for more information.

**Column headers (comma-separated)** `RCP-only`
If **Column header source** is set to **Custom**, supply here the column headers separated by commas.

**Use classic header style** `RCP-only`

Select this box to display a classic header style. This also displays multi-row headers, if **Column header source** is set to **From component** and multi-row headers are detected by the subfile header component. Clear this box to display default SWT table control headers in a single row only.

**Trim spaces on headers** `RCP-only`
If selected, the text inside of cells of a header row is trimmed of whitespace (extra spaces) from both ends.

**Enable foreground colors**
If selected, host screen foreground colors are rendered.

For Web applications, colors are mapped to CSS stylesheet classes representing that color. For example, if a host screen field is marked as **RED**, the Field widget will enclose the generated HTML for that field in a tag whose class name attribute is set to **HRED**. This allows for you to remap host screen colors on your generated Web page.

**Note:** The blacktheme.css directly maps host screen field foreground colors (for example, red text on the host screen will appear as red text on the generated Web page). However, other stylesheet files like monochrome.css, map field colors differently in an attempt to create a consistent, modern style.

For rich client applications, colors are mapped by the rich client template.

**Note:** This setting is overridden by the **Override color > Foreground color** setting.

**Field style class** `Web-only`
Optional. The CSS style class associated with the generated field. The value of the class attribute of the HTML tag will be set to this value. The default value is **HATSFIELD**. See "Using style sheets" on page 317 for more information.

**Enable extended attributes**
If selected, extended field attributes (blink, reverse video, underline, and column separator) are rendered with the specified style. See "Using style sheets" on page 317 for more information.

**Blink style** `Web-only`
The CSS style setting you want to use to render blinking text from the host screen.

**Reverse video style** `Web-only`
The CSS style setting you want to use to render reverse video text from the host screen.

**Underline style** `Web-only`
The CSS style setting you want to use to render underlined text from the host screen.

**Column separator style** `Web-only`
The CSS style setting you want to use to render column separators from the host screen.

**Strip underscores on input field**
Select this box if you want to strip the underscores from text when it is rendered.

**Trim spaces on input field**
Select this box to trim leading and trailing spaces from the input field.

**Enable cursor positioning option on input fields** `Web-only`
Select this box to allow the user to switch from data input mode to cursor positioning mode for input fields. When in data input mode, the user can enter data into the input field. When in cursor positioning mode, the user can tab to or otherwise position the cursor on any character in the input field. This setting is useful for Web projects that are optimized for mobile devices that do not have other cursor positioning capabilities. For more

information about other settings for this option that do not appear in the HATS Toolkit GUI, see "Considerations and limitations for mobile devices" on page 44.

**Provide spreadsheet file**
If selected, a launcher is provided that will retrieve the table data in spreadsheet file format.

**File name prefix**
Specifies a prefix for the spreadsheet file name. The default is **spreadsheet**. The complete file name will be in the form of `FileNamePrefix_TimeStamp`, where the timestamp format is MMDDYY_HHMMSS, for example, spreadsheet_041506_113020.

**File name extension**
The extension name for the type of spreadsheet format. Options are **.csv** and **.xls**. The .csv file are pure text files, with data columns separated by commas. The .xls files are Microsoft Excel Biff3 files. The default is **.csv**.

**Launcher type**
The type of launcher to display to the user. Options are **Link**, **Image**, and **Button**. The default is **Link** (Web projects) or **Button** (rich client projects).

**Launcher caption**
The caption for the launcher to display to the user. The default is **Download** (Web projects) or **Export** (rich client projects).

**Button style class** `Web-only`
The CSS style class associated with the generated launcher button. The value of the class attribute of the HTML button tag will be set to this value. The default value is **HATSBUTTON**. See "Using style sheets" on page 317 for more information.

**Link style class** `Web-only`
Optional. The CSS style class associated with the generated launcher link. The value of the class attribute of the HTML link tag will be set to this value. The default value is **HATSLINK**. See "Using style sheets" on page 317 for more information.

**Drop-down style class** `Web-only`
Optional. The CSS style class associated with the generated drop-down. The value of the class attribute of the HTML drop-down tag will be set to this value. The default value is **HATSDROPDOWN**. See "Using style sheets" on page 317 for more information.

**List option style class** `Web-only`
Optional. The CSS style class associated with each option in the drop-down. The default value is **HATSOPTION**. See "Using style sheets" on page 317 for more information.

**Table style class** `Web-only`
Optional. The CSS style class associated with the generated table. The value of the class attribute of the HTML table tag will be set to this value. The default value is **HATSTABLE**. See "Using style sheets" on page 317 for more information.

**Table cell style class** `Web-only`

Optional. The CSS style class associated with each cell in the generated table. The value of the class attribute of the HTML table cell (td) tag will be set to this value. The default value is **HATSTABLECELL**. See "Using style sheets" on page 317 for more information.

**Header row style class** `Web-only`

Optional. The CSS style class associated with each header row or column in the generated table. The value of the class attribute of the HTML table row or cell tag will be set to this value. The default value is **HATSTABLEHEADER**. See "Using style sheets" on page 317 for more information.

**Odd row style class** `Web-only`

Optional. The CSS style class associated with each odd numbered row in the table. The value of the class attribute of the each HTML table row tag in an odd row number will be set to this value. The default value is **HATSTABLEODDROW**. See "Using style sheets" on page 317 for more information.

**Even row style class** `Web-only`

Optional. The CSS style class associated with each even numbered row in the table. The value of the class attribute of the each HTML table row tag in an even row number will be set to this value. The default value is **HATSTABLEEVENROW**. See "Using style sheets" on page 317 for more information.

**Style** `Web-only`

Optional. Use the launcher button next to this field to open a style properties dialog. This dialog frees you from the need to understand CSS to change the font, color, or other style settings for the widget. See "Using style sheets" on page 317 for more information.

**Override font** `RCP-only`

Select this option and use the supplied **Font** setting controls to override the default font name, font style, and font size.

**Override color** `RCP-only`

Select this option and use the supplied **Foreground color** and **Background color** setting controls to override foreground and background colors.

**Note:** For rich client projects, the various color settings for this widget will be applied in the following order (from highest priority to lowest priority):

1. Highlighted rows / columns
2. Override color
3. Enable extended attributes / Enable foreground colors
4. Show alternating row colors

**fieldTextAlignment** `Web-only`

For Web projects, if you use a non-monospace font such as Sans Serif, the rendered data does not align properly with spacing problems between words. You can use the source setting, fieldTextAlignment, to correct the alignment of the data. To enable this setting on a project-wide basis, open the source view of the application.hap file, locate the class for the SubfileWidgetV6 widget, and change the value of the setting to `NO_SEGMENTING` as shown in the following example:

```
<class name="com.ibm.hats.transform.widgets.SubfileWidgetV6">
.
   <setting name="fieldTextAlignment" value="NO_SEGMENTING"/>
.
</class>
```

To enable this setting on a component-level basis in a transformation, open the source view for the transformation, locate the HATS:Component tag for the SubfileWidgetV6 widget, and change the value of the setting to `NO_SEGMENTING` as shown in the following example:

```
...
widget="com.ibm.hats.transform.widgets.SubfileWidgetV6"
...
widgetSettings="...|fieldTextAlignment:NO_SEGMENTING|..."
...
```

The values for this setting are `WORD_BY_WORD` and `NO_SEGMENTING`. The default value is **WORD_BY_WORD**.

**Note:** This setting has no effect when you use the Table (visual) recognition option for data recognition in the Subfile component.

---

## Subfile (popup)

The subfile (popup) widget is responsible for rendering a 5250 subfile.

This widget renders data supplied by the subfile component.

**Note:** For DBCS considerations when using this widget see "SBCS eliminate maximum length" on page 462.

The following figure shows how a subfile widget appears on a transformation, using the data from the Subfile component example as input:

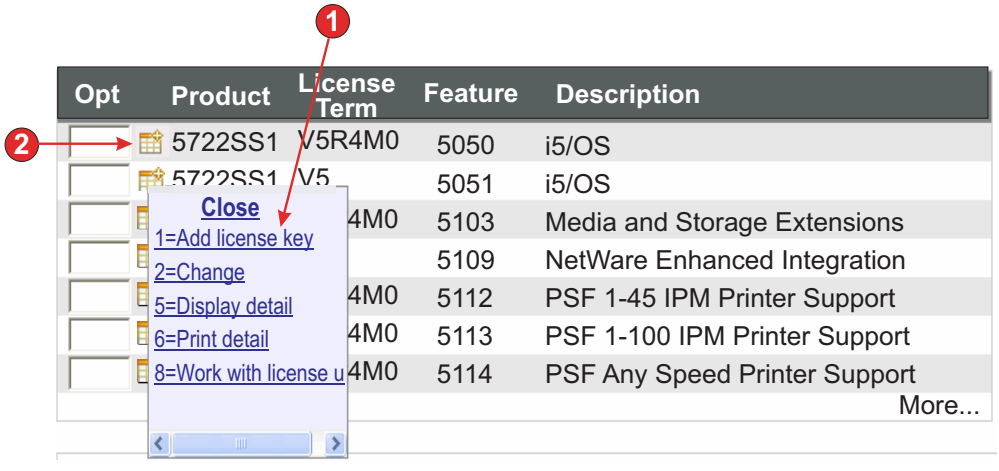



*Figure 55. Subfile (popup) widget example*

1. Caption type (show both)
2. Launcher type (image)

The following settings can be configured for this widget:

**Caption type:**
Specifies how the caption for each item in the pop-up dialog is determined. The value of the leading token and the description are derived from the component; you can select what appears as the caption. For example, if the host screen had an action that read **5. Display**, you can have the caption display **5**, or **Display**, or **5. Display**.

**Show subfile actions**
Select this box if you want to display the descriptive text of the subfile actions, such as **2=Add**, with the rendered subfile in addition to the pop-up dialog. The check box is not enabled when you select **Show description**.

**Auto submit on select**
If selected, once a selection is made in the pop-up dialog, it is submitted.

**Show submit button**
If selected, a submit button is rendered below the subfile.

**Submit button caption**
Optional. Specifies the caption of the submit button.

**Button style class** `Web-only`
Optional. The CSS style class associated with the generated submit button. The value of the class attribute of the HTML button tag will be set to this value. The default value is **HATSBUTTON**. See "Using style sheets" on page 317 for more information.

**Button row style class** `Web-only`
Optional. The CSS style class associated with the generated table row which contains the submit button. The default value is **HATSTABLEHEADER**. See "Using style sheets" on page 317 for more information.

**Launcher type**
Specifies whether to use an **Image**, a **Button**, or a **Link** to open the pop-up dialog.

**Image filename**
Optional. File name of the image to use for the launcher.

**Caption**
Optional. Caption to use for the launcher button or link.

**Link style class** `Web-only`
Optional. The CSS style class associated with the generated action link. The value of the class attribute of the HTML link tag will be set to this value. The default value is **HATSLINK**. See "Using style sheets" on page 317 for more information.

**Button style class** `Web-only`
Optional. The CSS style class associated with the generated action button. The value of the class attribute of the HTML button tag will be set to this value. The default value is **HATSBUTTON**. See "Using style sheets" on page 317 for more information.

**Close link caption** `Web-only`
The caption to appear for the Close link action.

**Rows per record**

**Show alternating row colors** `RCP-only`

Select this box to render rows in alternating colors. RGB color values are determined by the ITableColorProvider object returned by your template. See the HATS RCP API Reference section of the HATS Information Center at http://www.ibm.com/support/knowledgecenter/SSXKAY_9.7.0?topic=/com.ibm.hats.doc/doc/rcpjavadoc/index.html for more information about the RcpTemplate and ITableColorProvider classes.

**Number of rows**

Specify the fixed number of rows per record in your subfiles.

**Columns placement** `Web-only`

This widget can be configured to display primary columns of data, and optionally to have a details section to view additional detail columns. This setting is useful when displaying table data on a mobile device. For more information about these settings and about other settings for this option that do not appear in the HATS Toolkit GUI, see "Considerations and limitations for mobile devices" on page 44.

**Primary columns**

Use this field to specify which columns to display, and in what order. The field allows for numeric any-order listing of column numbers with expansion shorthand, and columns are allowed to be displayed more than once or not at all, for example, **1,3,2,5-7,2-3,9**.

A hyphen (-) between two numbers indicates a range of columns. You can use the asterisk character (*) to indicate the last available column. An asterisk alone means columns 1 through n where n is the last available column. You can also use 5* as a shorthand for 5-*. These mean the same thing: 1-*, *.

For Web projects, this setting defaults to *, or all columns. For Web projects optimized for mobile devices, this setting defaults to **1-2**, or columns 1 and 2.

**Note:** The numbers represent columns provided to the widget by the component. They do not represent the columns originally recognized by the component, which may or may not be excluded, before providing them to the widget.

**Detail columns**

Use this optional field to specify which detail columns to display when requested, and in what order. The field allows for numeric any-order listing with expansion shorthand.

For Web projects, this setting defaults to empty, or no detail columns section. For Web projects optimized for mobile devices, this setting defaults to **3***, or column 3 through the last column.

**Keep detail columns on the server**

Select this option to keep the detail columns section on the server until requested by the user. This reduces the amount of HTML output at initial rendering. The default is cleared.

**Highlighted rows** `RCP-only`

The comma-separated or ranged set of rows to be highlighted when rendered. This setting is useful for highlighting important rows in the generated table. The rows represented by this value are based on the recognized table, not on the actual host screen. Example values: **1,2,3....**,

**1,2-4**, **3-5**. RGB color values are determined by the ITableColorProvider object returned by your template. See the HATS RCP API Reference section of the HATS Information Center at http://www.ibm.com/support/knowledgecenter/SSXKAY_9.7.0?topic=/com.ibm.hats.doc/doc/rcpjavadoc/index.html for more information about the RcpTemplate and ITableColorProvider classes.

**Highlighted columns** `RCP-only`
The comma-separated or ranged set of columns to be highlighted when rendered. This setting is useful for highlighting important columns in the generated table. The columns represented by this value are based on the recognized table, not on the actual host screen. Example values: **1,2,3....**, **1,2-4**, **3-5**. RGB color values are determined by the ITableColorProvider object returned by your template. See the HATS RCP API Reference section of the HATS Information Center at http://www.ibm.com/support/knowledgecenter/SSXKAY_9.7.0?topic=/com.ibm.hats.doc/doc/rcpjavadoc/index.html for more information about the RcpTemplate and ITableColorProvider classes.

**Show grid lines** `RCP-only`
Select this box to show grid lines in the rendered subfile.

**Column header source** `RCP-only`
Select from the drop-down whether the text for the column headers is supplied by the subfile header component (**From component**) or by you (**Custom**). If supplied by the subfile header component, see the component description for "Header" on page 209 for more information.

**Column headers (comma-separated)** `RCP-only`
If **Column header source** is set to **Custom**, supply here the column headers separated by commas.

**Use classic header style** `RCP-only`

Select this box to display a classic header style. This also displays multi-row headers, if **Column header source** is set to **From component** and multi-row headers are detected by the subfile header component. Clear this box to display default SWT table control headers in a single row only.

**Trim spaces on headers** `RCP-only`
If selected, the text inside of cells of a header row is trimmed of whitespace (extra spaces) from both ends.

**Enable foreground colors**
If selected, host screen foreground colors are rendered.

For Web applications, colors are mapped to CSS stylesheet classes representing that color. For example, if a host screen field is marked as **RED**, the Field widget will enclose the generated HTML for that field in a tag whose class name attribute is set to **HRED**. This allows for you to remap host screen colors on your generated Web page.

**Note:** The blacktheme.css directly maps host screen field foreground colors (for example, red text on the host screen will appear as red text on the generated Web page). However, other stylesheet files like monochrome.css, map field colors differently in an attempt to create a consistent, modern style.

For rich client applications, colors are mapped by the rich client template.

**Note:** This setting is overridden by the **Override color > Foreground color** setting.

**Field style class** `Web-only`
Optional. The CSS style class associated with the generated field. The value of the class attribute of the HTML tag will be set to this value. The default value is **HATSFIELD**. See "Using style sheets" on page 317 for more information.

**Enable extended attributes**
If selected, extended field attributes (blink, reverse video, underline, and column separator) are rendered with the specified style. See "Using style sheets" on page 317 for more information.

**Blink style** `Web-only`
The CSS style setting you want to use to render blinking text from the host screen.

**Reverse video style** `Web-only`
The CSS style setting you want to use to render reverse video text from the host screen.

**Underline style** `Web-only`
The CSS style setting you want to use to render underlined text from the host screen.

**Column separator style** `Web-only`
The CSS style setting you want to use to render column separators from the host screen.

**Strip underscores on input field**
Select this box if you want to strip the underscores from text when it is rendered.

**Trim spaces on input field**
Select this box to trim leading and trailing spaces from the input field.

**Enable cursor positioning option on input fields** `Web-only`
Select this box to allow the user to switch from data input mode to cursor positioning mode for input fields. When in data input mode, the user can enter data into the input field. When in cursor positioning mode, the user can tab to or otherwise position the cursor on any character in the input field. This setting is useful for Web projects that are optimized for mobile devices that do not have other cursor positioning capabilities. For more information about other settings for this option that do not appear in the HATS Toolkit GUI, see "Considerations and limitations for mobile devices" on page 44.

**Provide spreadsheet file**
If selected, a launcher is provided that will retrieve the table data in spreadsheet file format.

**File name prefix**
Specifies a prefix for the spreadsheet file name. The default is **spreadsheet**. The complete file name will be in the form of `FileNamePrefix_TimeStamp`, where the timestamp format is MMDDYY_HHMMSS, for example, spreadsheet_041506_113020.

**File name extension**
The extension name for the type of spreadsheet format. Options are

**.csv** and **.xls**. The .csv file are pure text files, with data columns separated by commas. The .xls files are Microsoft Excel Biff3 files. The default is **.csv**.

**Launcher type**
The type of launcher to display to the user. Options are **Link**, **Image**, and **Button**. The default is **Link** (Web projects) or **Button** (rich client projects).

**Launcher caption**
The caption for the launcher to display to the user. The default is **Download** (Web projects) or **Export** (rich client projects).

**Button style class** `Web-only`
The CSS style class associated with the generated launcher button. The value of the class attribute of the HTML button tag will be set to this value. The default value is **HATSBUTTON**. See "Using style sheets" on page 317 for more information.

**Link style class** `Web-only`
Optional. The CSS style class associated with the generated launcher link. The value of the class attribute of the HTML link tag will be set to this value. The default value is **HATSLINK**. See "Using style sheets" on page 317 for more information.

**Input field style class** `Web-only`
Optional. The CSS style class associated with the generated input field. The value of the class attribute of the HTML input tag will be set to this value. The default value is **HATSINPUT**. See "Using style sheets" on page 317 for more information.

**Window style class** `Web-only`
Optional. The CSS style class associated with the generated window. The default value is **HATSPOPUP**. See "Using style sheets" on page 317 for more information.

**Close link style class** `Web-only`
Optional. The CSS style class associated with the link used to close the window. The default value is **HATSLINK**. See "Using style sheets" on page 317 for more information.

**Item link style class** `Web-only`
Optional. The CSS style class associated with the links used to select an item from the widget. The default value is **HATSPOPUPITEMLINK**. See "Using style sheets" on page 317 for more information.

**Table style class** `Web-only`
Optional. The CSS style class associated with the generated table. The value of the class attribute of the HTML table tag will be set to this value. The default value is **HATSTABLE**. See "Using style sheets" on page 317 for more information.

**Table cell style class** `Web-only`
Optional. The CSS style class associated with each cell in the generated table. The value of the class attribute of the HTML table cell (td) tag will be set to this value. The default value is **HATSTABLECELL**. See "Using style sheets" on page 317 for more information.

**Header row style class** `Web-only`

Optional. The CSS style class associated with each header row or column in the generated table. The value of the class attribute of the HTML table row or cell tag will be set to this value. The default value is **HATSTABLEHEADER**. See "Using style sheets" on page 317 for more information.

**Odd row style class** `Web-only`

Optional. The CSS style class associated with each odd numbered row in the table. The value of the class attribute of the each HTML table row tag in an odd row number will be set to this value. The default value is **HATSTABLEODDROW**. See "Using style sheets" on page 317 for more information.

**Even row style class** `Web-only`

Optional. The CSS style class associated with each even numbered row in the table. The value of the class attribute of the each HTML table row tag in an even row number will be set to this value. The default value is **HATSTABLEEVENROW**. See "Using style sheets" on page 317 for more information.

**Style** `Web-only`

Optional.Use the launcher button next to this field to open a style properties dialog. This dialog frees you from the need to understand CSS to change the font, color, or other style settings for the widget. See "Using style sheets" on page 317 for more information.

**Override font** `RCP-only`

Select this option and use the supplied **Font** setting controls to override the default font name, font style, and font size.

**Override color** `RCP-only`

Select this option and use the supplied **Foreground color** and **Background color** setting controls to override foreground and background colors.

**Note:** For rich client projects, the various color settings for this widget will be applied in the following order (from highest priority to lowest priority):

1. Highlighted rows / columns
2. Override color
3. Enable extended attributes / Enable foreground colors
4. Show alternating row colors

**fieldTextAlignment** `Web-only`

For Web projects, if you use a non-monospace font such as Sans Serif, the rendered data does not align properly with spacing problems between words. You can use the source setting, fieldTextAlignment, to correct the alignment of the data. To enable this setting on a project-wide basis, open the source view of the application.hap file, locate the class for the SubfilePopupWidget widget, and change the value of the setting to `NO_SEGMENTING` as shown in the following example:

```
<class name="com.ibm.hats.transform.widgets.SubfilePopupWidget">
.
   <setting name="fieldTextAlignment" value="NO_SEGMENTING"/>
.
</class>
```

To enable this setting on a component-level basis in a transformation, open the source view for the transformation, locate the HATS:Component tag for

the SubfilePopupWidget widget, and change the value of the setting to NO_SEGMENTING as shown in the following example:

```
...
widget="com.ibm.hats.transform.widgets.SubfilePopupWidget"
...
widgetSettings="...|fieldTextAlignment:NO_SEGMENTING|..."
...
```

The values for this setting are WORD_BY_WORD and NO_SEGMENTING. The default value is **WORD_BY_WORD**.

**Note:** This setting has no effect when you use the Table (visual) recognition option for data recognition in the Subfile component.

---

## Table

The Table widget is responsible for rendering a GUI table.

This widget renders data supplied by the following components:

- Table
- Table (field)
- Table (visual)

**Note:** For DBCS considerations when using this widget see "SBCS eliminate maximum length" on page 462.

The following figure shows how a table widget appears on a transformation, using the data from the Table (field) component example as input:

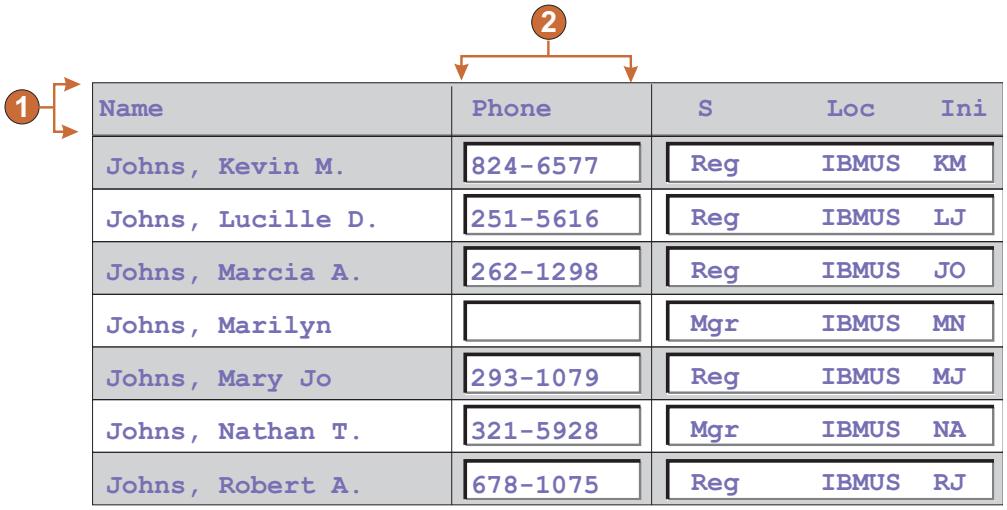



*Figure 56. Table widget example*

1. Header row (1)
2. Header column (not specified)

The following settings can be configured for this widget:

**Header row** `Web-only`

The comma-separated or ranged set of rows which should be rendered as a table header. This setting is useful for highlighting important rows in the generated table. The rows represented by this value are based on the recognized table, not on the actual host screen. Example values: **1,2,3....**, **1,2-4**, **3-5**. See the description for **Header row style class** setting.

**Highlighted rows** `RCP-only`

The comma-separated or ranged set of rows to be highlighted when rendered. This setting is useful for highlighting important rows in the generated table. The rows represented by this value are based on the recognized table, not on the actual host screen. Example values: **1,2,3....**, **1,2-4**, **3-5**. RGB color values are determined by the ITableColorProvider object returned by your template. See the HATS RCP API Reference section of the HATS Information Center at http://www.ibm.com/support/knowledgecenter/SSXKAY_9.7.0?topic=/com.ibm.hats.doc/doc/rcpjavadoc/index.html for more information about the RcpTemplate and ITableColorProvider classes.

**Highlighted columns** `RCP-only`

The comma-separated or ranged set of columns to be highlighted when rendered. This setting is useful for highlighting important columns in the generated table. The columns represented by this value are based on the recognized table, not on the actual host screen. Example values: **1,2,3....**, **1,2-4**, **3-5**. RGB color values are determined by the ITableColorProvider object returned by your template. See the HATS RCP API Reference section of the HATS Information Center at http://www.ibm.com/support/knowledgecenter/SSXKAY_9.7.0?topic=/com.ibm.hats.doc/doc/rcpjavadoc/index.html for more information about the RcpTemplate and ITableColorProvider classes.

**Show alternating row colors** `RCP-only`

Select this box to render rows in alternating colors. RGB color values are determined by the ITableColorProvider object returned by your template. See the HATS RCP API Reference section of the HATS Information Center at http://www.ibm.com/support/knowledgecenter/SSXKAY_9.7.0?topic=/com.ibm.hats.doc/doc/rcpjavadoc/index.html for more information about the RcpTemplate and ITableColorProvider classes.

**Show grid lines** `RCP-only`

Select this box to show grid lines in the rendered table.

**Column header source** `RCP-only`

Select from the drop-down whether the text for the column headers is supplied by the table component (**From component**) or by you (**Custom**). If supplied by the table component, see the component description of the **Number of title rows** setting for "Table" on page 212 for more information.

**Column headers (comma-separated)** `RCP-only`

If **Column header source** is set to **Custom**, supply here the column headers separated by commas.

**Trim spaces on headers**

If selected, the text inside of cells of a header row is trimmed of whitespace (extra spaces) from both ends.

**Header column** `Web-only`

The comma-separated or ranged set of table columns which should be rendered as a header. This setting is useful for highlighting important

columns (generally the first column) in the generated table. The columns represented by this value are based on the recognized table, not on the actual host screen. Example values: **1,2,3....**, **1,2-4**, **3-5**. See the description for **Header row style class** setting.

**Disable input**
If selected, rendered input fields are read only. A read only input field appears as a regular input field, but does not allow the user to modify its contents. This is useful in cases where you want to display the contents of a non-protected field to a user, but you do not want the user to modify the contents.

**Columns placement** `Web-only`

This widget can be configured to display primary columns of data, and optionally to have a details section to view additional detail columns. This setting is useful when displaying table data on a mobile device. For more information about these settings and about other settings for this option that do not appear in the HATS Toolkit GUI, see "Considerations and limitations for mobile devices" on page 44.

**Primary columns**
Use this field to specify which columns to display, and in what order. The field allows for numeric any-order listing of column numbers with expansion shorthand, and columns are allowed to be displayed more than once or not at all, for example, **1,3,2,5-7,2-3,9**.

A hyphen (-) between two numbers indicates a range of columns. You can use the asterisk character (*) to indicate the last available column. An asterisk alone means columns 1 through n where n is the last available column. You can also use 5* as a shorthand for 5-*. These mean the same thing: 1-*, *.

For Web projects, this setting defaults to *, or all columns. For Web projects optimized for mobile devices, this setting defaults to **1-2**, or columns 1 and 2.

**Note:** The numbers represent columns provided to the widget by the component. They do not represent the columns originally recognized by the component, which may or may not be excluded, before providing them to the widget.

**Detail columns**
Use this optional field to specify which detail columns to display when requested, and in what order. The field allows for numeric any-order listing with expansion shorthand.

For Web projects, this setting defaults to empty, or no detail columns section. For Web projects optimized for mobile devices, this setting defaults to **3***, or column 3 through the last column.

**Keep detail columns on the server**
Select this option to keep the detail columns section on the server until requested by the user. This reduces the amount of HTML output at initial rendering. The default is cleared.

**Strip underscores on input field**
Select this box if you want to strip the underscores from text when it is rendered.

**Trim spaces on input field**
Select this box to trim leading and trailing spaces from the input field.

**Enable cursor positioning option on input fields** `Web-only`
Select this box to allow the user to switch from data input mode to cursor positioning mode for input fields. When in data input mode, the user can enter data into the input field. When in cursor positioning mode, the user can tab to or otherwise position the cursor on any character in the input field. This setting is useful for Web projects that are optimized for mobile devices that do not have other cursor positioning capabilities. For more information about other settings for this option that do not appear in the HATS Toolkit GUI, see "Considerations and limitations for mobile devices" on page 44.

**Enable foreground colors**
If selected, host screen foreground colors are rendered.

For Web applications, colors are mapped to CSS stylesheet classes representing that color. For example, if a host screen field is marked as **RED**, the Field widget will enclose the generated HTML for that field in a tag whose class name attribute is set to **HRED**. This allows for you to remap host screen colors on your generated Web page.

**Note:** The blacktheme.css directly maps host screen field foreground colors (for example, red text on the host screen will appear as red text on the generated Web page). However, other stylesheet files like monochrome.css, map field colors differently in an attempt to create a consistent, modern style.

For rich client applications, colors are mapped by the rich client template.

**Note:** This setting is overridden by the **Override color > Foreground color** setting.

**Enable extended attributes**
If selected, extended field attributes (blink, reverse video, underline, and column separator) are rendered. Also, for 3270 Web applications, extended field colors are mapped (see the preceding setting for more information). See "Using style sheets" on page 317 for more information.

For rich client applications, extended field attributes are mapped by the rich client template.

**Blink style** `Web-only`
The CSS style setting you want to use to render blinking text from the host screen.

**Reverse video style** `Web-only`
The CSS style setting you want to use to render reverse video text from the host screen.

**Underline style** `Web-only`
The CSS style setting you want to use to render underlined text from the host screen.

**Column separator style** `Web-only`
The CSS style setting you want to use to render column separators from the host screen.

**Provide spreadsheet file**
If selected, a launcher is provided that will retrieve the table data in spreadsheet file format.

**File name prefix**
Specifies a prefix for the spreadsheet file name. The default is **spreadsheet**. The complete file name will be in the form of `FileNamePrefix_TimeStamp`, where the timestamp format is MMDDYY_HHMMSS, for example, spreadsheet_041506_113020.

**File name extension**
The extension name for the type of spreadsheet format. Options are **.csv** and **.xls**. The .csv file are pure text files, with data columns separated by commas. The .xls files are Microsoft Excel Biff3 files. The default is **.csv**.

**Launcher type**
The type of launcher to display to the user. Options are **Link**, **Image**, and **Button**. The default is **Link** (Web projects) or **Button** (rich client projects).

**Launcher caption**
The caption for the launcher to display to the user. The default is **Download** (Web projects) or **Export** (rich client projects).

**Button style class** `Web-only`
The CSS style class associated with the generated launcher button. The value of the class attribute of the HTML button tag will be set to this value. The default value is **HATSBUTTON**. See "Using style sheets" on page 317 for more information.

**Link style class** `Web-only`
Optional. The CSS style class associated with the generated launcher link. The value of the class attribute of the HTML link tag will be set to this value. The default value is **HATSLINK**. See "Using style sheets" on page 317 for more information.

**Table style class** `Web-only`
Optional. The CSS style class associated with the generated table. The value of the class attribute of the HTML table tag will be set to this value. The default value is **HATSTABLE**. See "Using style sheets" on page 317 for more information.

**Table cell style class** `Web-only`
Optional. The CSS style class associated with each cell in the generated table. The value of the class attribute of the HTML table cell (td) tag will be set to this value. The default value is **HATSTABLECELL**. See "Using style sheets" on page 317 for more information.

**Header row style class** `Web-only`
Optional. The CSS style class associated with each header row or column in the generated table. The value of the class attribute of the HTML table row or cell tag will be set to this value. The default value is **HATSTABLEHEADER**. See "Using style sheets" on page 317 for more information.

**Odd row style class** `Web-only`
Optional. The CSS style class associated with each odd numbered row in

the table. The value of the class attribute of each HTML table row tag in an odd row number will be set to this value. The default value is **HATSTABLEODDROW**. See "Using style sheets" on page 317 for more information.

**Even row style class** `Web-only`
Optional. The CSS style class associated with each even numbered row in the table. The value of the class attribute of each HTML table row tag in an even row number will be set to this value. The default value is **HATSTABLEEVENROW**. See "Using style sheets" on page 317 for more information.

**Style** `Web-only`
Optional.Use the launcher button next to this field to open a style properties dialog. This dialog frees you from the need to understand CSS to change the font, color, or other style settings for the widget. See "Using style sheets" on page 317 for more information.

**Override font** `RCP-only`
Select this option and use the supplied **Font** setting controls to override the default font name, font style, and font size.

**Override color** `RCP-only`
Select this option and use the supplied **Foreground color** and **Background color** setting controls to override foreground and background colors.

**Note:** For rich client projects, the various color settings for this widget will be applied in the following order (from highest priority to lowest priority):

1. Highlighted rows / columns
2. Override color
3. Enable extended attributes / Enable foreground colors
4. Show alternating row colors

---

## Text input

The Text input widget is responsible for rendering HTML input fields.

This widget renders data supplied by the following components:

- Command line
- Input field

**Note:** For DBCS considerations when using this widget see "SBCS eliminate maximum length" on page 462.

The following figure shows how a text input widget appears on a transformation, using the data from the Command line component example as input:

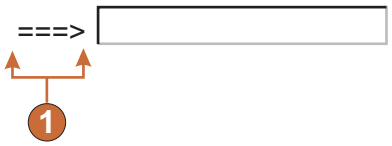



*Figure 57. Text input widget example*

1. Caption (from component)

The following settings can be configured for this widget:

**Override size**
If selected, the specified size value will be set on each generated input field. This setting is useful for visually changing the size of a large host input field. If cleared, the size of the input field is determined by the recognizing component.

**Size** The size of each input field.

**Note:** This value is interpreted by the browser – no guarantee is made as to how an input field will be rendered on different browsers.

**Override maximum length**
If selected, the specified maximum length value will be set on each generate input field. This setting is useful for manually restricting the number of characters that can be entered into an input field. If cleared, the maximum number of allowable characters is determined by the recognizing component.

**Maximum length**
The maximum number of characters that can be entered into each input field.

**Add input attributes**
If selected, the specified attributes will be added on each input field. This setting is useful for adding any extra attributes which are useful for customization. If type attribute is specified, the default type=text will be overridden by specified attribute. Same way maxlength and size attributes also will be overridden by specified attributes.

**Input attributes**
Any valid attribute value pair or a set of attribute value pairs separated by space. Any double quotes entered will be converted to single quotes. If any attribute value contains spaces, specify it with in single quotes

**Note:** The attribute value pair is interpreted by the browser - no guarantee is made as to how an input field will be rendered on different browsers. Example input attributes: type=date type='image' alt='Click me!' src='my-img.png' width='80' height='30'

**Caption source**
Specifies how the caption for the generated input field is determined. Options: **From component** (use the extracted caption from the component) or **Custom** (use the specified caption).

**Custom caption**
Optional. Specifies the caption for the input field.

**Note:** Leave this setting blank to not generate a caption for the check box.

**Number of columns per row**
The number of input fields to display horizontally before wrapping to the next line.

**Notes:**

1. This setting is not applicable in default rendering.
2. The widget preview does not always match the full page preview. This happens because the widget preview simply renders the component's output. It does not try to preserve the alignment of the screen when it renders, as is done in the full page preview.

**Read only**
If selected, the rendered input field is read only. A read only input field appears as a regular input field, but does not allow the user to modify its contents. This is useful in cases where you want to display the contents of a non-protected field to a user, but you do not want the user to modify the contents.

**Strip underscores on input field**
Select this box if you want to strip the underscores from text when it is rendered.

**Trim spaces on input field**
Selecting this trims leading and trailing spaces from the input field.

**Enable cursor positioning option on input fields** `Web-only`
Select this box to allow the user to switch from data input mode to cursor positioning mode for input fields. When in data input mode, the user can enter data into the input field. When in cursor positioning mode, the user can tab to or otherwise position the cursor on any character in the input field. This setting is useful for Web projects that are optimized for mobile devices that do not have other cursor positioning capabilities. For more information about other settings for this option that do not appear in the HATS Toolkit GUI, see "Considerations and limitations for mobile devices" on page 44.

**Enable foreground colors**
If selected, host screen foreground colors are rendered.

For Web applications, colors are mapped to CSS stylesheet classes representing that color. For example, if a host screen field is marked as **RED**, the Field widget will enclose the generated HTML for that field in a tag whose class name attribute is set to **HRED**. This allows for you to remap host screen colors on your generated Web page.

**Note:** The blacktheme.css directly maps host screen field foreground colors (for example, red text on the host screen will appear as red text on the generated Web page). However, other stylesheet files like monochrome.css, map field colors differently in an attempt to create a consistent, modern style.

For rich client applications, colors are mapped by the rich client template.

**Note:** This setting is overridden by the **Override color > Foreground color** setting.

**Enable 3270 numeric lock**
If selected, the user can enter only numbers 0 through 9, plus sign, minus sign, period, comma, and Hindi numerals into 3270 numeric fields. If cleared, any characters can be entered. The default is cleared.

**Note:** For Web applications, you can customize the list of acceptable keys in the lxgwfunctions.js file in the allowNumLockOnly() and allowNumLockOnlyForIEMobile() functions, as appropriate, using character code values.

**Enable extended attributes**
If selected, extended field attributes (blink, reverse video, underline, and column separator) are rendered. Also, for 3270 Web applications, extended field colors are mapped (see the preceding setting for more information). See "Using style sheets" on page 317 for more information.

For rich client applications, extended field attributes are mapped by the rich client template.

**Blink style** `Web-only`
The CSS style setting you want to use to render blinking text from the host screen.

**Reverse video style** `Web-only`
The CSS style setting you want to use to render reverse video text from the host screen.

**Underline style** `Web-only`
The CSS style setting you want to use to render underlined text from the host screen.

**Column separator style** `Web-only`
The CSS style setting you want to use to render column separators from the host screen.

**Input field style class** `Web-only`
Optional. The CSS style class associated with the generated input field. The value of the class attribute of the HTML input tag will be set to this value. The default value is **HATSINPUT**. See "Using style sheets" on page 317 for more information.

**Caption style class** `Web-only`
Optional. The CSS style class associated with the generated input field's caption. The default value is **HATSCAPTION**. See "Using style sheets" on page 317 for more information.

**Table style class** `Web-only`
Optional. The CSS style class associated with the generated table. If more than one input field is rendered, an HTML table will be generated to enclose these fields. The value of the class attribute of the HTML table tag will be set to this value. The default value is **HATSTABLE**. See "Using style sheets" on page 317 for more information.

**Style** `Web-only`
Optional.Use the launcher button next to this field to open a style properties dialog. This dialog frees you from the need to understand CSS to change the font, color, or other style settings for the widget. See "Using style sheets" on page 317 for more information.

**Override font** `RCP-only`
Select this option and use the supplied **Font** setting controls to override the default font name, font style, and font size.

**Override color** `RCP-only`
Select this option and use the supplied **Foreground color** and **Background color** setting controls to override foreground and background colors.

## Toolbar RCP-only

The Toolbar widget enables you to render function key and selection list items as buttons on a toolbar. The items can be rendered either on the main Transformation view toolbar, or on a stand-alone toolbar, which is useful when creating sections on a panel and providing a toolbar for a particular section. When items are rendered on the Transformation view toolbar, the style of button (text, image, or both text and image) is determined by the project-level toolbar settings. For more information about the project-level toolbar settings see "Toolbar RCP-only " on page 98. When items are rendered on a stand-alone toolbar, you control the style of the toolbar using the settings for this widget.

This widget renders data supplied by the following components:

- Function key
- Selection list

The following figure shows how a Toolbar widget appears on the Transformation view toolbar using data provided by a function key component as input:

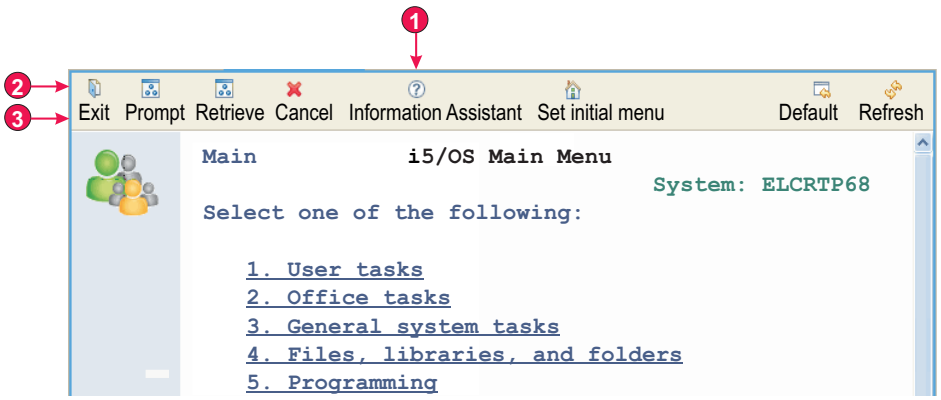



*Figure 58. Toolbar widget example*

1. Contribute items to main toolbar
2. Display toolbar items as (both text and image)

   **Note:** This setting (**Display toolbar items as**) is configured on the main **Toolbar** setting on the **Rendering** tab of the Project Settings editor. Its value applies when **Contribute items to main toolbar** is selected for the Toolbar widget.
3. Caption type (show description)

The following settings can be configured for this widget:

**Contribute items to main toolbar**
Select this box to render the items on the main Transformation view toolbar. If selected, toolbar items are rendered based on the project-level **Toolbar** settings on the **Rendering** tab of the Project Settings editor. For example, if the **Render items under a single toolbar item** setting is selected for this widget, its **Caption** and **Image** settings apply only if allowed by the project-level **Display toolbar items as** setting. For more information see "Toolbar RCP-only " on page 98.

Clear this box to render the items on a stand-alone toolbar. If using this option, you control how the items are rendered using the **Display item as** settings below.

**Display item as**
When rendering items on a stand-alone toolbar, use this setting to control how to display each item. Options are to display each item as **Text**, **Image**, or **Both text and image**.

**Render items under a single toolbar item**
Select this box to specify that all items rendered by this widget are displayed as menu items under a single item on the toolbar.

**Caption**
Specifies the text caption to use on the single toolbar item. Depending on the **Contribute items to main toolbar** setting, this setting has no effect if the project-level **Display toolbar items as** setting or the widget-level **Display item as** settings is set to **Image**. For more information see the description of the **Contribute items to main toolbar setting** above. The default is **Actions**.

**Image** Specifies the image to use on the single toolbar item. Select an image from the drop-down, or use the **Import** button to import an image. Depending on the **Contribute items to main toolbar** setting, this setting has no effect if the project-level **Display toolbar items as** setting or the widget-level **Display item as** settings is set to **Text**. For more information see the description of the **Contribute items to main toolbar setting** above.

**Caption type**
Use this setting to control what text caption is displayed for each item. This setting has no effect if **Display item as** is set to **Image**. Options are **Show leading token**, **Show description**, or **Show both**. For example, if the host text is `F3=Exit`, then **Show leading token** displays **F3**, **Show description** displays **Exit**, and **Show both** displays **F3=Exit**.

**Hover help source**
Controls what text is displayed as a tool tip when you hover over a toolbar item. Options are **Show leading token**, **Show description**, or **Show both**. For example, if the host text is `F3=Exit`, then **Show leading token** displays **F3**, **Show description** displays **Exit**, and **Show both** displays **F3=Exit**.

**Use project-level image mappings**
Select this box to specify that project-level image mappings are searched for an image to display for a toolbar item. Project-level mappings are searched after first searching component-level or default rendering-level mappings.

**Project-level image mappings**
When you render images for toolbar items, you can map source text from the function key or selection list to a specific image. For example, for the function key source text, `F3=Exit`, you can map to an image based on the leading token (`F3`), the description (`Exit`), or the full caption (`F3=Exit`). Unlike with other widget settings, with this setting you can configure project-level image mappings in the Project Settings editor and extend (not just overwrite) these mappings at the component or default rendering level. This allows you to configure generic project-level mappings, and make small changes at the component or default rendering level.

Use the **Add** button to add a new mapping. The **Add Condition** dialog appears with the following settings:

**Text** Defines which part of the function key or selection list source text to search for the value specified in the **Value** field. Options are **Leading token**, **Description**, or **Both**. For example, if the source text is `F3=Exit`, then specifying **Leading token** searches `F3`, specifying **Description** searches `Exit`, and specifying **Both** searches `F3=Exit` to find a match with the value specified in the **Value** field.

**Operator**
Defines what is considered a match when searching the text defined by the **Text** field for the value specified in the **Value** field. Options are **Contains** or **Equals**. **Contains** means there is a match if the value is found anywhere in the text. In this case the search is not case-sensitive. **Equals** means there is a match only if the text is equal to the value. In this case the search is case-sensitive.

**Value** Defines the value to search for in the text defined by the **Text** field. Multiple values can be specified using a vertical bar (`|`) symbol, for example `Exit|Quit|Leave`.

**Image** Defines the image to use if a match is found. Select an image from the drop-down list or click the **Import** button to import an image.

**Preview**
Displays a view of the selected image.

Use the **Edit**, **Remove**, **Up** and **Down** buttons to further define your image mappings.

**Note:** Text replacement is performed prior to the Toolbar widget's determination of which image to use. Mappings you define must assume that text replacement may have changed the leading token, description, or both for the item.

**defaultImage**
Use this setting to specify a default image to use when no image is mapped for a specific function key or selection list item.

**Note:** This setting does not appear in the GUI as a configurable setting. It must be set in the source for the Toolbar widget settings. For example, if you want to use as the default the details.gif image in the Rich Client Content/Images folder in your project, do either of the following:

To specify the default at the project level, in the HATS Projects view double-click the **Project Settings** file in the project folder to launch the editor for the application.hap file. On the Source tab locate the class, name="com.ibm.hats.rcp.transform.widgets.SwtToolbarWidget", in the <classSettings> tag and change the setting for defaultImage to `<setting name="defaultImage" value="images/details.gif"/>`.

To set the default image for a ComponentRendering control on a transformation, update the defaultImage string passed to the setWidgetSettings method of the ComponentRendering object. For example change `defaultImage:` to `defaultImage:images/details.gif`.

To use an image contained in another plug-in, specify the full path to the image within that plugin. For example change `defaultImage:` to `defaultImage:org.eclipse.ui/icons/full/obj16/delete_obj.gif`.

# Widget settings (Dojo)

## HATS Dojo widget considerations and limitations

Dojo is an open-source technology and is under continual development. To see an updated list of known considerations and limitations, including supported browsers, when using HATS Dojo widgets, see "Host Access Transformation Services 9.7 - Known issues and workarounds" at http://www.ibm.com/support/docview.wss?uid=ibm10876092. Also refer to Dojo support pages found at http://dojotoolkit.org/.

Following is a list of considerations and limitations when working with HATS Dojo widgets:

- Dojo widgets are supported only in HATS Web applications. They are not supported in the following HATS applications:
  - Bidirectional
  - Mobile device
  - Rich client
- Dojo widgets are not supported in the following HATS implementations:
  - Alternate rendering
  - Default rendering
  - Global rules
  - Screen combinations
  - HATS tabbed folders
- Dojo elements encompassing large areas of a host screen may not fit into desired layouts. If multiple component elements within a single component creates a problem for your layout, consider specifying smaller host screen regions to layout individual component elements.
- For information about customizing HATS Dojo widgets, see Customizing a HATS Dojo widget in the *HATS Web Application Programmer's Guide*.

---

## Combo box (Dojo) Web-only

The Combo box widget is responsible for rendering a Dojo combo box.

Characteristics of the Combo box widget are:

- A list of value pairs consisting of a list item caption and a corresponding list item value to be submitted to the host is displayed in a drop-down combo box. Depending on the component being rendered, the value pairs can be supplied by the component or manually by you in the widget settings.

  **Note:** Both the caption and value are displayed in the drop-down list.
- The Combo box widget is editable, so users can type in any value at runtime, not just what is provided in the list.

**Note:** The main difference between the Combo box and Filtering select widgets is that the Combo box widget allows users to type in and submit values not already in the list.

- As users type into the combo box, partially matched values from the list are displayed in a drop-down list.
- The widget is customizable with many API options, which include validation and constraint options, error messages, icons, auto-complete, and more. For example, by default there is no client-side validation for the Dojo Combo box. To see an example of how to add client-side validation, along with other widget customizations, refer to Customizing a HATS Dojo widget, in the *HATS Web Application Programmer's Guide*.

This widget renders data supplied by the following components:

- Command line
- Input field
- Input field with hints
- Item selection
- Selection list

The following figure shows how a Combo box widget appears on a transformation, using the data supplied using the **Fill from string** setting:

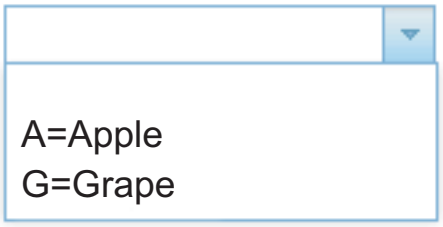



*Figure 59. Dojo Combo box widget example*

The figure below shows an example of a partially-matched value displayed in a pop-up list. In this example, the user typed a `G` into the input text box.

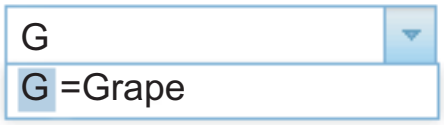



*Figure 60. Dojo Combo box widget filtering example*

As shown in the following figure, the Combo box widget, unlike the Filtering select widget, enables the user to type and submit values not in the supplied list.


*Figure 61. Dojo Combo box widget user supplied input example*

The following settings can be configured for this widget:

**Fill from global variable**
If selected, fill the drop-down list from the specified global variable.

**Global variable containing list values**
Specifies the name of the indexed global variable containing the set of values. An item for each index in the global variable will be created in the drop-down list.

**Shared**
HATS local and shared global variables can have the same name. Select this box if you want to use the shared global variable to populate the list items. When this box is cleared, the local global variable is used.

**Global variable containing list captions**
Optional. Specifies the name of the indexed global variable containing the set of captions. The size of the global variable specified by this value should be greater than or equal to the size specified in the preceding setting. The indexes in this indexed global variable should also match up with the indexes in the values global variable (so that the actual value and caption shown to the user are in sync). If this value is not specified, the caption for each item in the drop-down list will be its value.

**Fill from string**
If selected, fill the drop-down list from the specified string.

**List items**
Optional. Specifies the string of items to include in the drop-down list. Items should be separated with a semicolon (;). To have the list item caption be different than the list item value, enter both separated by an equal sign (=). For example, a value of **Apple=A;Grape=G** renders a drop-down list with two items: **A=Apple** and **G=Grape**. Selecting the first item causes an **A** to be inserted in the associated host screen input field.

If you want both the item in the drop-down list and the value inserted in the host screen to be the same, you only need to enter the item. For example, **Apple=A;G**. In this example, a **G** appears in the drop-down list and in the host screen input field.

**Auto submit on select**
If selected, once a selection is made in the drop-down list, it is submitted.

---

## Date text box (Dojo) Web-only

The Date text box widget is responsible for rendering a Dojo date text box with a pop-up calendar date picker.

This widget renders data supplied by the following component:

- Input field

The following figure shows how a Date text box widget appears on a transformation.

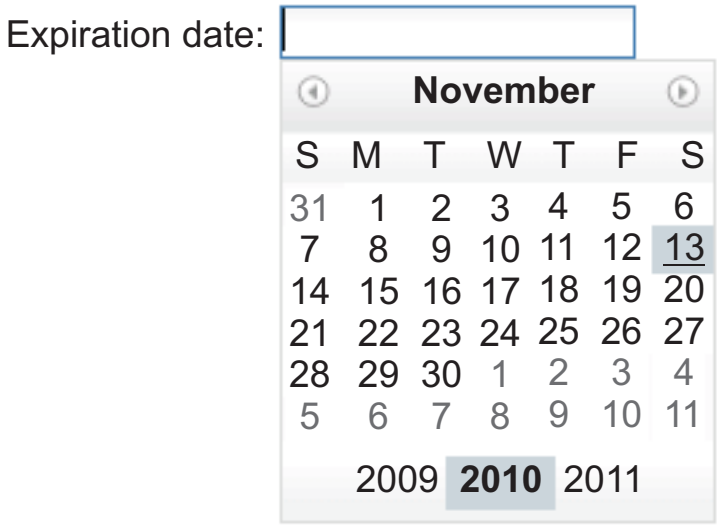



*Figure 62. Dojo Date text box widget example*

The following settings can be configured for this widget:

**Pattern**

Required. When the user selects a date from the pop-up calendar, the selected date must be formatted to map correctly to what is expected in the host application's input field. This setting specifies the pattern HATS uses to correctly format the selected date. For information about the meaning of symbols in the pattern, see http://docs.dojocampus.org/dojo/date/locale/format.

When you enter a pattern and save it, the HATS Toolkit performs the following conversions to ensure correct symbols are used:

- Uppercase `D` to lowercase `d`
- Uppercase `Y` to lowercase `y`
- Lowercase `m` to uppercase `M`
- Lowercase `e` to uppercase `E`

**Restrict earliest selectable date**

If selected, the value specified in the Date field is the earliest date the user can select from the pop-up calendar. The user is not prevented from manually entering an earlier date directly into the associated input field because of this setting.

**Note:** If the user enters a date outside this range, a warning symbol is displayed. If a range message (default or user-supplied) is defined, then it is displayed.

**Date** Enter the date in format MM/DD/YYYY for all locales.

**Restrict latest selectable date**

If selected, the value specified in the Date field is the latest date the user can select from the pop-up calendar. The user is not prevented from manually entering a later date directly into the associated input field because of this setting.

**Note:** If the user enters a date outside this range, a warning symbol is displayed. If a range message (default or user-supplied) is defined, then it is displayed.

**Date** Enter the date in format MM/DD/YYYY for all locales.

**Default value**

Optional. This field represents the initial date selected on the pop-up calendar when the host application does not pre-fill the associated input field with a valid date between the specified restricted dates. The interaction between this field and the associated host input field is handled as follows:

- If the host application pre-fills the input field with zeros or an incorrect date format, the default value does not overwrite the host field data. To update the associated input field, the user must either manually enter a date, or select a date with the pop-up calendar.
- If the default value is not specified or is in an incorrect format, today's date will be initially selected on the pop-up calendar.
- If you want to pre-fill the associated input field, add an **Insert Data** action to your screen customization event before applying the transformation.

**Note:** The date pattern is MM/DD/YYYY for all locales.

**Use default invalid message**

Select this box to display the default Dojo invalid message when the user enters an invalid date. Clear this box to supply your own invalid message to display.

**Invalid message**

Use this field to supply your own invalid message. If you leave this field empty, no message is displayed and only the input field is highlighted.

**Note:** If you supply your own invalid message, it is displayed to the user in the language you supply. No translation is performed.

**Use default range message**

Select this box to display the default Dojo range message when the user enters an out-of-range date. Clear this box to supply your own range message to display.

**Range message**

Use this field to supply your own range message. If you leave this field empty, no message is displayed and only the input field is highlighted.

**Note:** If you supply your own range message, it is displayed to the user in the language you supply. No translation is performed.

**Prompt message**

Use this field to supply a prompt message to display when the user selects the text box. For example, you could remind the user of the correct date format to use or to select a date from the pop-up calendar.

**Note:** If you supply your own prompt message, it is displayed to the user in the language you supply. No translation is performed.

**Re-validate on submit**
Select this box to prevent page submission when the user enters an invalid date. If not selected, an invalid date is flagged, but still can be submitted to the host.

---

## Enhanced grid (Dojo) Web-only

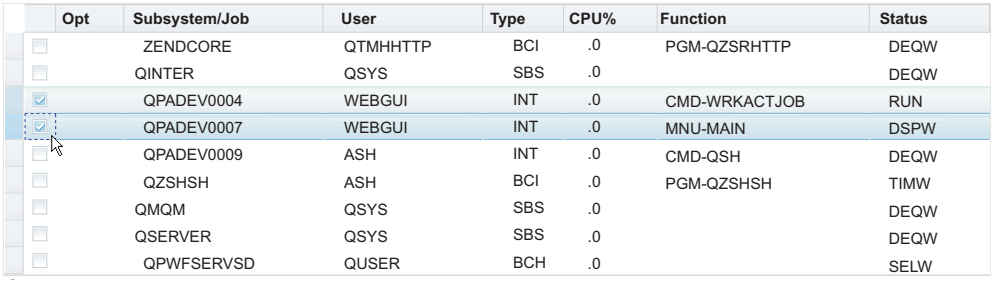
The Enhanced grid widget is responsible for rendering a Dojo table. The rendered table is read-only.

This widget renders data supplied by the following components:

- Table
- Table (field)
- Table (visual)

The following figure shows how an Enhanced grid widget appears on a transformation, using data similar to that shown in the Table component example as input.

**Note:** This figure shows the widget appearance when using the Dojo Claro theme. The widget might appear differently when using a different Dojo theme.



| Opt | Subsystem/Job | User | Type | CPU% | Function | Status |
|---|---|---|---|---|---|---|
| | ZENDCORE | QTMHHTTP | BCI | .0 | PGM-QZSRHTTP | DEQW |
| | QINTER | QSYS | SBS | .0 | | DEQW |
| | QPADEV0004 | WEBGUI | INT | .0 | CMD-WRKACTJOB | RUN |
| | QPADEV0007 | WEBGUI | INT | .0 | MNU-MAIN | DSPW |
| | QPADEV0009 | ASH | INT | .0 | CMD-QSH | DEQW |
| | QZSHSH | ASH | BCI | .0 | PGM-QZSHSH | TIMW |
| | QMQM | QSYS | SBS | .0 | | DEQW |
| | QSERVER | QSYS | SBS | .0 | | DEQW |
| | QPWFSERVSD | QUSER | BCH | .0 | | SELW |

*Figure 63. Dojo Enhanced grid widget example*

When using the Dojo Claro theme, default rendering of the Enhanced grid widget includes a row selector column as the first column in the table. Users can click cells in this column to select rows to drag to another location in the table. To select multiple consecutive rows, click the first row, press Shift, and click the last row, or click the first row and holding down the mouse, swipe to the last row. To select multiple nonconsecutive rows, press Ctrl while clicking each row. To drag, click the selected row, or rows, and drag.

**Note:** If focus is on a cell or row of the table, pressing the Enter key is handled by the widget and not sent to the host application.

Another function of the default rendering of the Enhanced grid widget includes a sort function. You can sort rows by column contents in ascending or descending order and remove a sort. You can also perform a nested sort, that is, sorting a second column nested within a sorted first column.

**Note:** Sorting of columns is based on simple string comparison. Therefore some strings, for example date strings of format `mm/dd/yy`, do not sort as expected.

To perform a sort, click the column heading as shown in the example below.

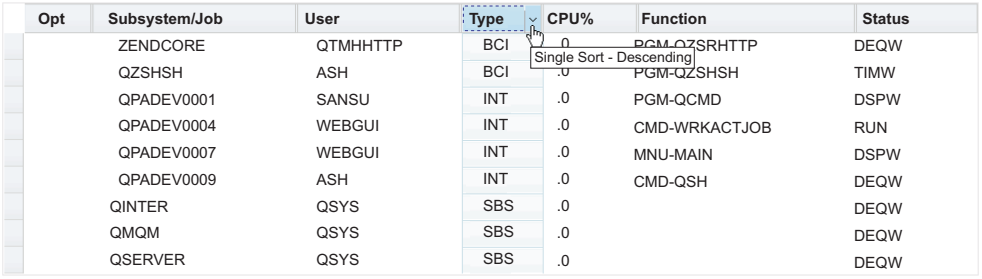



*Figure 64. Dojo Enhanced grid widget sort example*

---

## Filtering select (Dojo) `Web-only`

The Filtering select widget is responsible for rendering a Dojo drop-down input text box.

Characteristics of the Filtering select widget are:

- A list of value pairs consisting of a list item caption and a corresponding list item value to be submitted to the host is displayed in a drop-down input text box. Depending on the component being rendered, the value pairs can be supplied by the component or manually by you in the widget settings.

  **Note:** Both the caption and value are displayed in the drop-down list.
- The Filtering select widget is editable, so users can type in values at runtime.
- As users type into the input text box, partially matched values from the list are displayed in a drop-down list.
- User-supplied text that does not match a value in the list is flagged as an error. Invalid input results in a blank being submitted to the host.

  **Note:** The main difference between the Filtering select and Combo box widgets is that the Filtering select widget does not allow users to submit values not already in the list.
- The widget is customizable with many API options, which include validation and constraint options, error messages, icons, auto-complete, and more.

This widget renders data supplied by the following components:

- Command line
- Input field
- Input field with hints
- Item selection
- Selection list

The following figure shows how a Filtering select widget appears on a transformation, using the data from the Selection list component example as input:

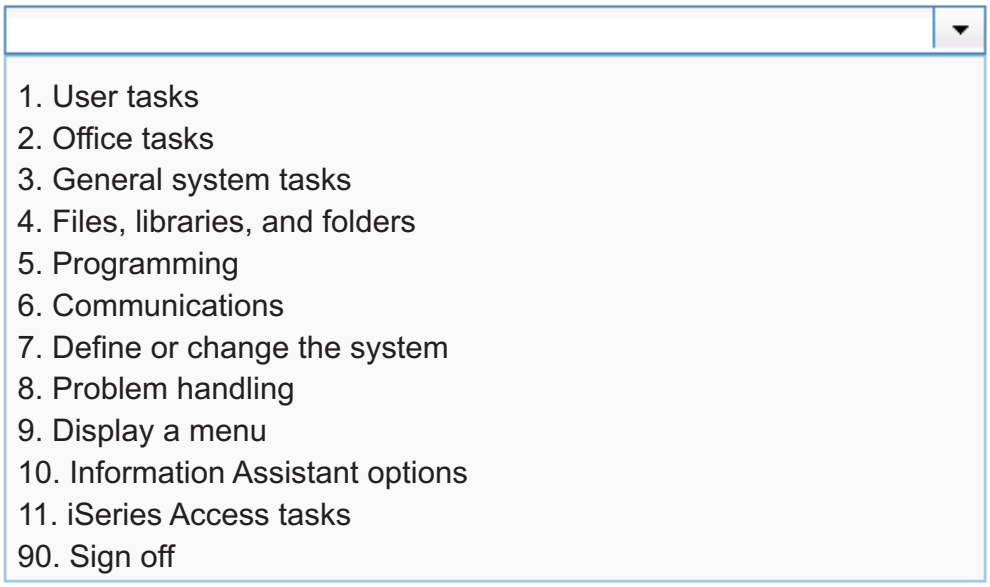



*Figure 65. Dojo Filtering select widget example*

The figure below shows an example of partially-matched values displayed in a pop-up list. In this example, the user typed a 1 into the input text box.

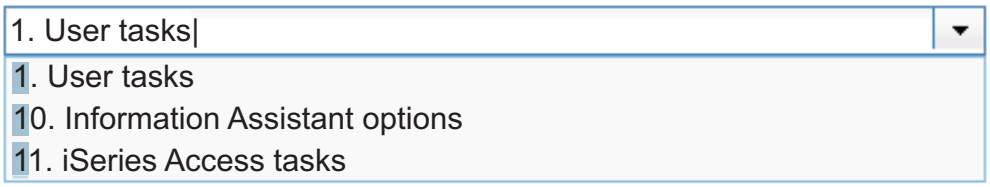



*Figure 66. Dojo Filtering select widget filtering example*

As shown in the following figure, the Filtering select widget, unlike the Combo box widget, prevents the user from typing and submitting values not in the supplied list.


*Figure 67. Dojo Filtering select widget invalid user supplied input example*

The following settings can be configured for this widget:

**Fill from global variable**
If selected, fill the drop-down list from the specified global variable.

**Global variable containing list values**
Specifies the name of the indexed global variable containing the set of values. An item for each index in the global variable will be created in the drop-down list.

**Shared**
HATS local and shared global variables can have the same name.

Select this box if you want to use the shared global variable to populate the list items. When this box is cleared, the local global variable is used.

**Global variable containing list captions**
Optional. Specifies the name of the indexed global variable containing the set of captions. The size of the global variable specified by this value should be greater than or equal to the size specified in the preceding setting. The indexes in this indexed global variable should also match up with the indexes in the values global variable (so that the actual value and caption shown to the user are in sync). If this value is not specified, the caption for each item in the drop-down list will be its value.

**Fill from string**
If selected, fill the drop-down list from the specified string.

**List items**
Optional. Specifies the string of items to include in the drop-down list. Items should be separated with a semicolon (;). To have the list item caption be different than the list item value, enter both separated by an equal sign (=). For example, a value of **Apple=A;Grape=G** renders a drop-down list with two items: **A=Apple** and **G=Grape**. Selecting the first item causes an **A** to be inserted in the associated host screen input field.

If you want both the item in the drop-down list and the value inserted in the host screen to be the same, you only need to enter the item. For example, **Apple=A;G**. In this example, a **G** appears in the drop-down list and in the host screen input field.

**Auto submit on select**
If selected, once a selection is made in the drop-down list, it is submitted.

---

## Text box (Dojo) Web-only

The Text box widget is responsible for rendering a Dojo text box.

This widget renders data supplied by the following components:

- Command line
- Input field

The following figure shows how a text box widget appears on a transformation, using the data from the Command line component example as input:


*Figure 68. Dojo Text box widget example*

**Note:** If you see a label displayed above the input field, you can adjust the width on the <td> tag containing the field in the source to get the desired rendering of the field.

---

## Validation text box (Dojo) Web-only

The Validation text box widget is responsible for rendering a Dojo text box that validates user input using a regular expression that you specify.

This widget renders data supplied by the following component:

- Input field

The following figure shows how a Validation text box widget appears on a transformation with an optional prompt message that you provide.


*Figure 69. Dojo Validation text box widget example with prompt message*

This figure shows a Validation text box widget with valid input entered.


*Figure 70. Dojo Validation text box widget example with valid input*

The figure below shows a Validation text box widget with the default invalid message.


*Figure 71. Dojo Validation text box widget example with invalid message*

The following settings can be configured for this widget:

**Regular expression**
Use this field to supply a regular expression to validate the format of the user-supplied data. If this field is empty, no validation is performed and users can enter anything.

**Validate**
Click this button to validate the syntax of the regular expression in the **Regular expression** field.

**Prompt message**
Use this field to supply a prompt message to display when the user selects the text box. For example, you could remind the user of the correct data format as defined by the regular expression field.

**Note:** If you supply your own prompt message, it is displayed to the user in the language you supply. No translation is performed.

**Use default invalid message**

Select this box to display the default Dojo invalid message when the user enters invalid data. Clear this box to supply your own invalid message to display.

**Invalid message**

Use this field to supply your own invalid message. If you leave this field empty, no message is displayed and only the input field is highlighted.

**Note:** If you supply your own invalid message, it is displayed to the user in the language you supply. No translation is performed.

**Re-validate on submit**

Select this box to prevent page submission when the user enters invalid data. If not selected, invalid data is flagged, but still can be submitted to the host.

## Component and widget mapping

The widgets supplied in HATS Toolkit for use in displaying host components in a GUI are mapped to those components. The following table lists the existing HATS host components and their corresponding widgets.

*Table 2. HATS host components and their corresponding widgets*

| Host component | Widget |
|---|---|
| Command line | Combo RCP-only<br>Drop-down (data entry)<br>List<br>Popup<br>Radio button (data entry)<br>Text input<br>Combo box (Dojo) Web-only<br>Filtering select (Dojo) Web-only<br>Text box (Dojo) Web-only |
| Field | Field |
| Function key | Button<br>Button table<br>Drop-down (selection)<br>Link<br>Radio button (selection)<br>Toolbar RCP-only |
| HTML DDS keyword | Label |

*Table 2. HATS host components and their corresponding widgets (continued)*

| Host component | Widget |
|---|---|
| Input field | Calendar **Web-only**<br>Check box<br>Combo **RCP-only**<br>Drop-down (data entry)<br>List<br>Popup<br>Radio button (data entry)<br>Text input<br>Combo box (Dojo) **Web-only**<br>Date text box (Dojo) **Web-only**<br>Filtering select (Dojo) **Web-only**<br>Text box (Dojo) **Web-only**<br>Validation text box (Dojo) **Web-only** |
| Input field with hints | Combo **RCP-only**<br>Drop-down (data entry)<br>List<br>Popup<br>Radio button (data entry)<br>Combo box (Dojo) **Web-only**<br>Filtering select (Dojo) **Web-only** |
| Item selection | Check box<br>Combo **RCP-only**<br>Drop-down (data entry)<br>Link (item selection)<br>List<br>Popup<br>Radio button (item selection)<br>Text input<br>Combo box (Dojo) **Web-only**<br>Filtering select (Dojo) **Web-only** |
| Light pen (attention) | Button<br>Link<br>Radio button (selection) |
| Light pen (selection) | Check box |
| Scrollbar field (ENPTUI) | Scrollbar (ENPTUI) |
| Selection field (ENPTUI) | Button<br>Check box<br>Link<br>Radio button (selection) |
| Selection list | Button<br>Button table<br>Drop-down (selection)<br>Link<br>Radio button (selection)<br>Toolbar **RCP-only**<br>Combo box (Dojo) **Web-only**<br>Filtering select (Dojo) **Web-only** |
| Subfile | Subfile (check box)<br>Subfile (drop-down)<br>Subfile (popup) |

*Table 2. HATS host components and their corresponding widgets (continued)*

| Host component | Widget |
|---|---|
| Table | Graph (horizontal bar)<br>Graph (line)<br>Graph (vertical bar)<br>Table<br>Enhanced grid (Dojo) **Web-only** |
| Table (field) | Graph (horizontal bar)<br>Graph (line)<br>Graph (vertical bar)<br>Table<br>Enhanced grid (Dojo) **Web-only** |
| Table (visual) | Graph (horizontal bar)<br>Graph (line)<br>Graph (vertical bar)<br>Table<br>Enhanced grid (Dojo) **Web-only** |
| Text | Label |
| URL | Link |

# Chapter 10. Using templates

Templates enable you to further control the appearance of your HATS application by customizing a template to match the style of your corporate Web site or applications, or other guidelines. A template is a JSP file in HATS Web projects or a Java SWT composite class in HATS rich client projects. Templates have an area reserved for the host screen that is rendered by a HATS transformation. They can contain company logos and information and, in the case of HATS Web templates, links to other Web pages. A template also defines the background color, image, or both, for the area where the transformed host screen appears.

Launching the HATS Template wizard enables you to create a blank template or a template based on one of those shipped with HATS. For HATS Web projects you can also base a template on a Web page (URL or file). You may want to import an existing Web page as a template based on your company's Web site. For HATS rich client projects you can import a template as a Java source file from another project by copying the source file into the project or by using the Eclipse import wizard.

HATS supplies templates that you can use in your projects. You can see the names of these templates by expanding the **Web Content** `Web-only` , or **Rich Client Content** `RCP-only` , folder then selecting the **Templates** folder in the **HATS Projects** view. The supplied templates include some or all of the following definitions:

- Default background colors
- Default foreground colors
- Default color mappings to map host colors to GUI colors
- Default fonts
- An area for the host screen rendered with a transformation
- For Web templates
  - Style attributes for widgets
  - Image definitions created using the .gif, .jpg, and .png files located in the **Web Content/Common/Images** folder
  - The HATS default application keypad

**Notes:**

1. Newly supplied templates are marked with an asterisk on the **Template** tab of the Project Settings editor.
2. When you create a HATS project and select **Optimize options for mobile devices**, only templates that are optimized for mobile devices are provided for use in the project.

## Template examples

This section describes some of the templates shipped with HATS.

Figure 72 on page 312 shows a host screen using the finance.jsp template that is shipped for use by HATS Web applications. This template provides an example that uses drop-down menus with links to other Web sites along with the application keypad at the bottom.


*Figure 72. HATS-supplied Web application finance template*

Figure 73 on page 313 shows a host screen using the industry.jsp template that is shipped for use by HATS Web applications. This template provides an example that uses menu bar links to other Web sites along with the application keypad at the bottom.

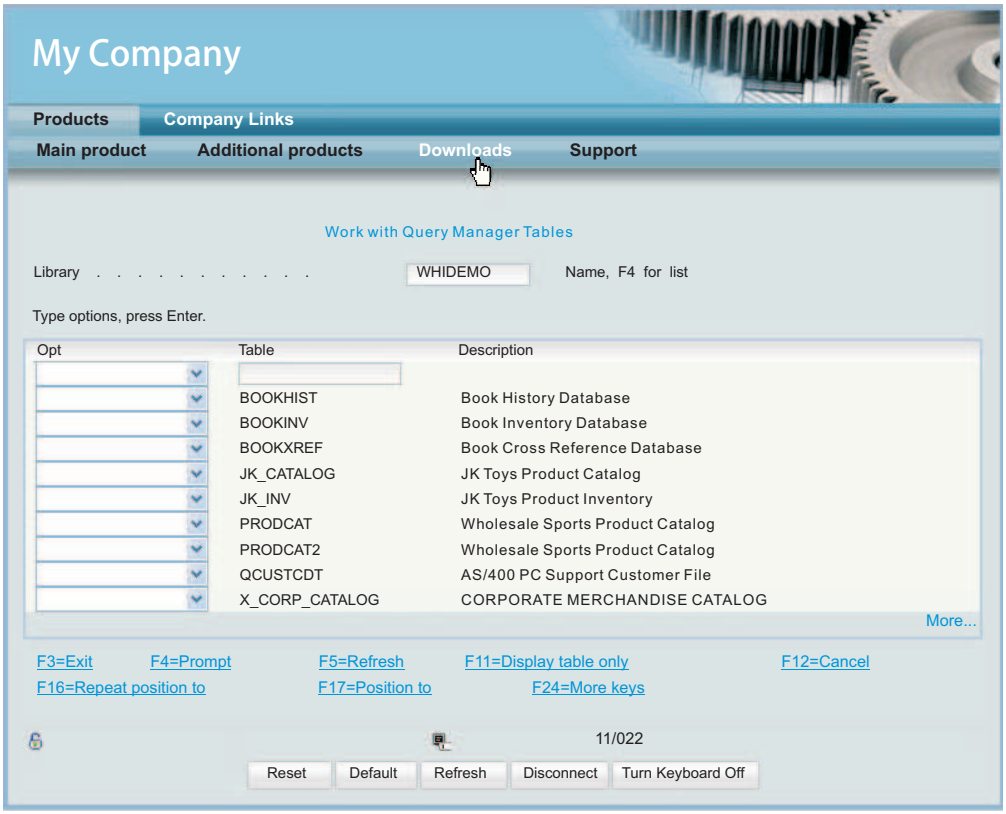



*Figure 73. HATS-supplied Web application industry template*

Figure 74 on page 314 shows a host screen using the modern.java template that is shipped for use by HATS rich client applications. This template is more simple, with just a sidebar, based on the assumption that the runtime environment (for example, Eclipse RCP, Lotus Notes, or Lotus Expeditor) will provide the overall common appearance including buttons, links, and so on.

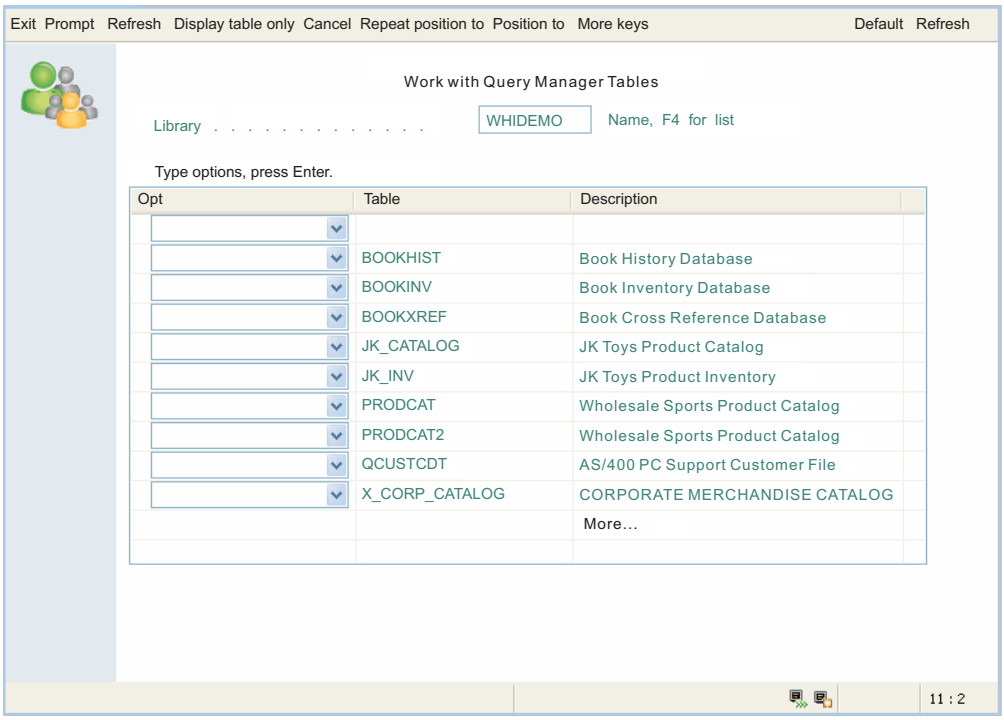



*Figure 74. HATS-supplied rich client application modern template*

You can make copies of HATS resources in the **HATS Projects** view by opening any folder and right-clicking on the resource and selecting **Copy**. Choose the folder where you want to **Paste** the resource and HATS will prompt you for a name. This can be useful as backup when you begin creating your own templates.

**Note:** You can not copy from the **HATS Projects** view into the **Navigator** view.

When you create your project in HATS Toolkit, you select a template to use as the default template for your project. This template is shown in the **HATS Projects** view with a different icon and the word **default** next to it. You can change the default template by going to the **HATS Projects** view and right-clicking on a template located in **Web Content/Templates** `Web-only` , or **Rich Client Content/Template** `RCP-only` , and select **Set as Default Template**. Another way is by double-clicking on the **Project Settings** of your HATS project in the **HATS Projects** view. Go to the **Template** tab and make your selection for the list.

When you create screen customizations, and the action is to apply a transformation or show URL, you can select the template to use when the transformation is applied. To make your host application consistent across all screens, you can use the same template for your transformation that you selected for your project default template. To do this, select the template that you want to use as the default template for this project.

## Create a Template wizard

You can create your own custom templates to meet functional needs and corporate style guidelines. You can design custom templates for your projects using the wizards and editors in HATS Toolkit. HATS automatically adds the necessary code to include color and fonts, the HATS default application keypad, and an area for the host screen rendered with a transformation to any templates that you create.

Use the Create a Template wizard to create a new template. You can access the wizard a number of ways, such as:

- In the **HATS Projects** view right-click on your project and select **New HATS > Template**.
- Click the **Create HATS Template** icon on the HATS toolbar.
- Select **HATS > New > Template** (or **File > New > HATS Template**) on the HATS menu bar.

In the wizard, specify the project and give your template a name. If the template is for a rich client project, the **Package** field is prefilled with a default package name as determined by the Default Java packages HATS preferences. See "Using HATS preferences" on page 126 for more information. You can use the **Browse** button to find and select a different package. Optionally, set a description for your template, then click **Next** and select one of the following options:

**Create a blank template**

Select this option to create a blank template.

**Prefill the template from an existing project template**

Select this option to prefill the new template from an existing template. Select the name of an existing template. What you are actually doing is making a copy of an existing template using the name you specified when you started the Create a Template wizard. A preview of the template you select appears in the window below.

**Prefill the template from an existing Web page (URL or file)** `Web-only`

Select this option to prefill the new template from an existing Web page (such as your company's existing Web site) or a specific file. Type in a URL or file in the text box or click the **Browse** button to search for one. Clicking **Preview** will display the template you have selected.

The Web page, as well as the linked files (for example, images, style sheets), will be downloaded from the Web. Links to the files will be changed to match the corresponding path in your HATS project.

**Note:** Custom JavaScript, applets, and so on that are sometimes embedded in an imported Web page may cause difficulties. If your Web site has complex tags, they may need to be modified in order for your template to work with HATS. The Create a Template wizard downloads the page, all images and style sheets but does not make any attempt to try to filter out possibly unsupported JavaScript calls. Web pages using frames are not supported

To make your new template the default template for the project, select the **Set as the default template for this project** check box.

Click **Finish.** This will launch the appropriate editor where you can make changes and modifications to your new template.

Keep in mind that any changes you make to objects in a project only affect that project. If you want to use the template you create for other projects, you need to copy that template to any new projects you create along with any style sheets or images that were modified.

For HATS Web projects, if you create your own template outside of the HATS Toolkit environment, ensure that the following statements are included:

**<LINK rel="stylesheet" href="xxx">**
One or more tags that refers to a style sheet that defines each of the following settings:

- Appearance of the buttons, links, and keypads
- Colors for the template background and widgets
- The font and size for text

**<HATS:Transform>**
This tag defines the location of the transformation that the template surrounds.

# Editing templates

Templates in HATS Web projects are JSP files and by default are edited using the Rich Page Editor. Templates in HATS rich client projects are Java SWT composites and by default, are edited using the Window Builder, or manually edited using Java Editor or any text editors.

## Editing templates for Web projects

You can see all of the templates, those supplied by HATS and any that you have created, by expanding the **Templates** folder under **Web Content** in the **HATS Projects** view. To edit a template using the Rich Page Editor that is built into Rational SDP, double-click on the name of the template. To see other available editors, right-click on the name of the template and select **Open With**. See Rational SDP documentation for the Rich Page Editor by selecting **Help > Help Contents** from the menu bar, and search for `Rich Page Editor`.

The following restrictions apply to editing templates:

- Custom templates must be UTF-8 encoded.
- Do not use any JSP variable, CSS class, HATSForm, or any other object whose name starts with HATS, hats, or Hats. These names are reserved for use by HATS.
- Do not create a form named HATS_form. HATS generates this form while assembling HTML outputs.
- Do not use a form that encloses the HATS:Transform tag. HATS generates a form in the place where the HATS:Transform tag is. HTML does not allow nested forms.

The following sections describe each tab of the Rich Page Editor.

### Design

The **Design** tab of the Rich Page Editor displays the current view of the template as you make changes to it. While on this tab, additional edit options are available from the HATS Tools context menu. For example, you can drag HATS components from the **Palette** and add HTML tags.

You can also insert global variables, macro keys, host and application keypads, or individual keys. You can insert these items using the **HATS Tools** menu on the HATS Toolkit menu bar.

If you want to add images to your project, you should import them into the Web Content/Common/Images directory of your project. To import images, click **File > Import > General > File System** to open the Import wizard. Select the location of the image source files you want to import in the **From directory** field. Select the

*project_name*/Web Content/common/images directory as the destination **Into Folder**. When your image source files are imported, right-click on the **Images** folder, and select **Show thumbnails** on the **Thumbnails** tab in the lower right window to see the images in the folder. You can use the drag-and-drop method to copy images into the **Design** tab view of your template.

**Note:** For considerations when using GB18030 national language characters, see "Using code page 1388 (GB18030)" on page 431.

When you click the **Insert Host Component**, **Insert Macro** or **Insert Global Variable** items in the **HATS Tools** menu, a wizard appears where you define those items, as you do in transformations. See "Insert Host Component" on page 178, "Insert Macro Key" on page 182 and "Insert Global Variable" on page 182 for more information about these wizards.

**Using style sheets:** A cascading style sheet is a file that defines a hierarchical set of style rules for controlling the rendering of HTML. It allows you to control elements of output such as font color, size, and background color in order to maintain the consistency of the area of the screen rendered by HATS with the style of the template.

*Modifying style sheets:* HATS provides style sheets to modify color schemes and font size. At least one of these style sheets is applied to the template. While viewing the template on the **Design** tab, you can apply these style sheets to your template.

By going to the **Web Content/Common/Stylesheets** folder in the **HATS Projects** view, you can then double-click the style sheet you want to edit. By default, this opens the style sheet in the CSS Designer editor window along with a preview window that displays samples of the styles in the style sheet.

When you click a particular style in the CSS Designer editor window, a sample of the style is displayed in the preview window.

*Determining which styles to edit:* It is common to want to change the color of a field on a host screen. The challenge is to know which style is being applied to that particular host field. In order to determine this, you can view the source of the Web page when the host field is displayed, and by examining the source, you can determine which style is being applied.

Suppose you want to change the text that it is being used for the User and Password text on the Sign On screen of a HATS project. To view the source of the Web page, run the HATS project in an actual Web browser until you can see the field in question. Do not use the Web browser built into Rational SDP. You can run the HATS project on the test server in Rational SDP, but you must access the project using an actual browser, for example, Internet Explorer.

When the field in question is visible, if, for example, you are using Internet Explorer, click **View** on the browser's menu bar and then click **Source** from the drop-down menu.

Search in the text file for the text that is in the field or near to the field that you want to modify, in this case, **User**. Just before the text in the source, you will find a `CLASS=` statement. That statement provides the CLASS or style that is being applied to the text. You can then go into the template or style sheet and edit the style to get the effect you want.

**Note:** Instead of viewing the source, if available with your browser, you can use new developer tools, for example Internet Explorer Developer Tools, to inspect particular page elements.

The style change will be applied throughout the HATS application where this style is used. If you create a default application, where the default template is applied to every host screen, this style will be used on every screen. If you want to have a style applied to just one host screen, then you must create a template that gets applied to that specific host screen.

For more information about cascading style sheets, go to http://www.w3.org/Style/CSS/.

**HATS style sheets:** The following style sheets are shipped with HATS and are included in each new project you create.

*Table 3. HATS style sheets*

| Name | Description |
|---|---|
| blacktheme | Black background; foreground color matches host screen color |
| calendar | Calendar popup styles |
| commontheme | Defines elements common to all the style sheets. It is imported by the main style sheet (blacktheme, graytheme, monochrometheme, tantheme, whitetheme) |
| finance | Style sheet for the Finance.jsp template |
| graytheme | Gray background; foreground colors are different shades of gray |
| industry | Style sheet for the Industry.jsp template |
| inlineCalendar | Style sheet for the inlineCalendar option of the Calendar widget |
| largeFont | Font size of all generated output relative to your browser setting and class function |
| medical | Style sheet for the Medical.jsp template |
| monochrometheme | White background; foreground colors are different shades of gray, links are blue |
| nonFixedFont | White background; font size varies |
| normalFont | Font size of all generated output relative to your browser setting and class function |
| PrintJobWindow | Print job window styles |
| research | Style sheet for the Research.jsp template |
| reverseVideoBlack | Reverses the foreground and background colors of the blacktheme style sheet |
| reverseVideoGray | Reverses the foreground and background colors of the graytheme style sheet |
| reverseVideoMono | Reverses the foreground and background colors of the monochrometheme style sheet |
| reverseVideoTan | Reverses the foreground and background colors of the tantheme style sheet |
| reverseVideoWhite | Reverses the foreground and background colors of the whitetheme style sheet |

*Table 3. HATS style sheets (continued)*

| Name | Description |
|---|---|
| scaleableFont | Font size of all generated output scaled according to the browser |
| smallFont | Font size of all generated output relative to your browser setting and class function |
| tantheme | Tan background; foreground colors are different shades of gray and blue |
| transport | Style sheet for the Transport.jsp template |
| whitetheme | White background; foreground color matches host screen color |
| xlargeFont | Font size of all generated output relative to your browser setting and class function |
| xsmallFont | Font size of all generated output relative to your browser setting and class function |

**Design tab considerations and limitations:** Following is a list of considerations and limitations when using the Design tab.

- Do not rely on the **Design** tab for an accurate browser preview of your template.
- The **Design** tab does not process JavaScript. So, if any visual manipulation of the page is done using JavaScript, it does not display.
- Some CSS files do not accurately display on the **Design** tab.

## Source

The **Source** tab displays the HTML and JSP tags in the JSP file for all the parts of the template. As you make changes on other tabs in the Rich Page Editor, the tags and attributes displayed in the source file change to match.

You can also make changes to the tags and attributes in the source file, and they are reflected on the other tabs of the Rich Page Editor.

## Making customized templates, style sheets, and images available

HATS enables you to share customized templates, style sheets and images.

- To share customized templates, copy the template .jsp files from

  `<workspace_directory>\project name\Web Content\templates`

  to

  `<shared_install_directory>\plugins\com.ibm.hats_nnn\predefined\projects\language\new\Web Content\templates`

  where *shared_install_directory* is the shared resources directory where you installed the HATS offering using IBM Installation Manager, *nnn* is the version and build level of HATS, and *language* is en for English.
- If you copy customized versions of the Finance.jsp, Industry.jsp, Research.jsp, or Transport.jsp templates, you must also copy the appropriate JavaScript file from

  `<workspace_directory>\project name\Web Content\common\scripts`

  to

  `<shared_install_directory>\plugins\com.ibm.hats_nnn\predefined\projects\new\Web Content\common\scripts`

- To make customized style sheets available to all new HATS projects, copy .css files from:

  `<workspace_directory>\project name\Web Content\common\stylesheets`

  to:

  `<shared_install_directory>\plugins\com.ibm.hats_nnn\predefined\projects\new\Web Content\common\stylesheets`
- To make customized images available to all new HATS projects, copy .jpg, .gif, or .png files from

  `<workspace_directory>\project name\Web Content\common\images`

  to

  `<shared_install_directory>\plugins\com.ibm.hats_nnn\predefined\projects\new\Web Content\common\images`

  **Note:** Some template images are found in subdirectories of the \images subdirectory.

After copying these files, the next time you create a new project you can select the new templates, style sheets, or images.

## Editing templates for rich client projects

A HATS rich client template is a specialized Java SWT composite class that implements a specific Java interface. This interface allows HATS to detect and use Java source files that are to be used as HATS templates in the rich client environment. HATS rich client templates are very basic with a simple header or sidebar and no footer. This is because it is assumed that the runtime environment (for example, Eclipse RCP, Lotus Notes, or Lotus Expeditor) will provide for the overall common appearance as well as areas for common buttons, links, and so on.

A template cannot contain host components (ComponentRendering composites), default rendering composites, or global variable controls, while a transformation can contain them.

You can see all of the templates, those supplied by HATS and any that you have created, by expanding the **Templates** folder under **Rich Client Content** in the **HATS Projects** view.

To edit a template using the default editor, double-click on the name of the template. To see the available editors, right-click on the name of the template and select **Open With**.

Refer to the chapter https://www.ibm.com/support/knowledgecenter/en/SSXKAY_9.7.0/com.ibm.hats.doc/rcppgd07.htm in the *HATS Rich Client Platform Programmer's Guide* for more information.

# Application keypad

The application keypad contains keys (such as **Refresh**, **Default**, and **View Print Jobs**) that represent application-level functions. These keys control functions within the HATS application. The application keypad keys include:

**Turn Keyboard On/Off**
: Toggles support for using the physical keys on the host keyboard. If keyboard support is not enabled for the project, this button does not

appear on the application keypad. For more information, see Chapter 16, “Enabling keyboard support,” on page 359.

**Reset** Clears all the fields on the browser page of any entries made by the user.

**Disconnect**
Disconnects from the host session. If this key is clicked, a link appears to let the user reconnect to the host.

**Screen Reverse (if applicable)**
Toggles the screen image from a left-to-right image to a right-to-left image or vice-versa, if the application is running on a host with an Arabic or Hebrew code page. If an Arabic or Hebrew code page is not selected in the project connection settings, this button does not appear on the application keypad.

**Default**
Turns off any customization of the host screen by a transformation and presents the entire host screen as basic HTML. The Default key can be modified by selecting a different template or style sheet for the transformation.

**Refresh**
Refreshes the current browser screen and performs the current action again.

**View Print Jobs (if applicable)**
Shows a list of print jobs that the user has created. If print support is not enabled for the project, this button does not appear on the application keypad.

By default, HATS displays the application keypad. To hide the keypad, go to the **HATS Projects** view, double click the **Project Settings** of your HATS project, select the **Rendering** tab, and click on **Application Keypad**. You then have the option of clearing the check box to hide the keypad.

**Note:** Custom keypads and individual keys are defined using individual tags or composites, as a result they are displayed in the template's editor even if the keypad is disabled in your project settings.

Refer to Chapter 5, “Modifying a HATS project,” on page 87 for more information about the settings for keyboard support and keypads.

# Chapter 11. Macros and host terminal

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

HATS supports macro-based customization to speed up and customize the host application process. Macros can be used for the following functions:

- As an action such as the play macro action or the perform macro transaction action. Playing a macro must be the last action performed for a screen event. For more information see "Actions" on page 154.
- To connect or disconnect a particular host connection. For more information see "Macros" on page 143.
- To generate an Integration Object. For more information, see "Creating an Integration Object" on page 341.
- In a transformation. See "Insert Macro Key" on page 182 for more information.

You can use macros to do any or all of the following functions:

**Skip-screen**
Skip-screen macros are navigational macros that move the user from one screen to another screen without displaying intervening screens.

**Prompt**
Prompt macros contain steps to request input from users during the host session. They can also set prompts from a user list. For example, you can use a prompt macro to ask a user for a user ID and password before logging into a host application.

**Note:** User list prompts can only be used in connect macros.

Prompt macros can also be used by developers to provide constant values or values obtained from global variables.

**Extract**
Extract macros contain events to extract host screen information as a string, as a list of strings, or as a table. For example, you can use an extract macro to connect to a directory-type host application and extract the results of doing a search in the directory.

Macros recorded or imported in HATS Toolkit are saved in a HATS macro (.hma) file. You can see the macros defined in your project by expanding the **Macros** node in the **HATS Projects** view of the HATS Toolkit.

HATS provides three different editors you can use to view and modify HATS macros, the Macro Editor, the Advanced Macro Editor, and the Visual Macro Editor. The Visual Macro Editor is the default editor for HATS macros. When you double-click on the name of a macro, the Visual Macro Editor is opened. For more information about the Visual Macro Editor, see the chapter, Visual Macro Editor, in the *HATS Advanced Macro Guide*.

To open the Macro Editor, right-click on the name of the macro and select **Open With > Macro Editor**.

The Advanced Macro Editor is opened from the Macro Editor Overview page. For more information about the Advanced Macro Editor, see the chapter, Advanced Macro Editor, in the *HATS Advanced Macro Guide*.

**Note:** Support for the Macro Editor and the Advanced Macro Editor is deprecated in HATS V9.7. While support continues for now, IBM reserves the right to remove these capabilities in a subsequent release of the product. This support is replaced by the Visual Macro Editor.

# Macro Editor

The following sections describe each tab of the macro editor.

## Overview

The **Overview** tab of the macro editor summarizes general information about the macro, such as the name, description and last modified date and time. The items you can modify on this tab are the description of the macro, the pausetime, timeout and connection.

**Pausetime** is the number of milliseconds of delay after each action is performed. **Timeout** is the number of milliseconds to wait before terminating the macro and **Connection** lets you select which connection to use from the drop-down menu.

The **Open Host Terminal** and **Advanced Editor** buttons are described in detail below.

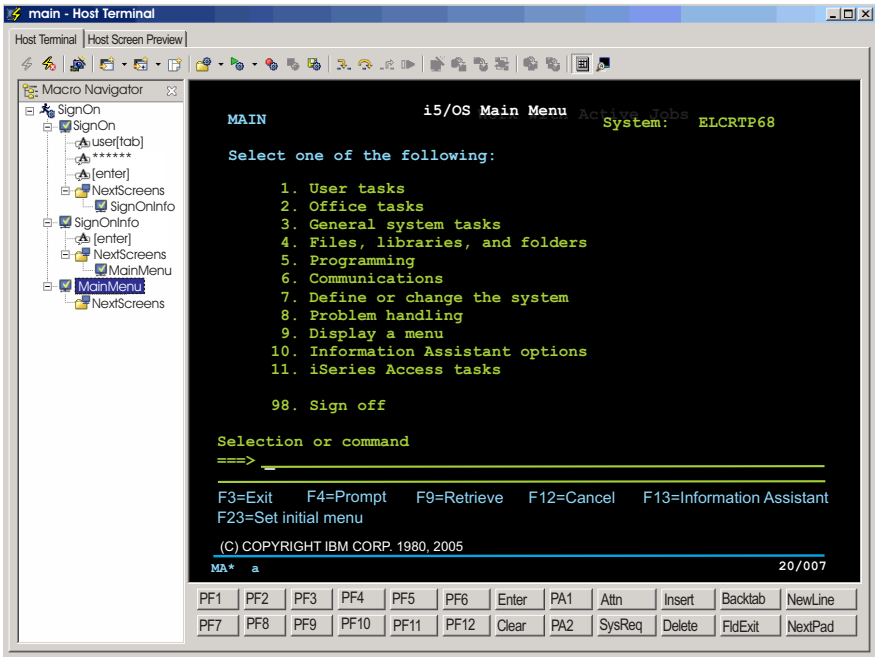



*Figure 75. HATS macro recording*

### Open Host Terminal

You can click **Open Host Terminal** to create or work with your macro using the host terminal. Once the terminal opens, you can use any of the icons on the toolbar to record and create your macro.

**Host Terminal:** You can also use the **Host Terminal** to edit a previously recorded macro by right-clicking on the nodes in the macro tree and selecting **Edit**. If you select a screen to edit, this will launch the screen recognition criteria editor which will allow you to modify your screen. If you edit any input that you entered in your screen, a window will open up and allow you to change or modify the input. If you edit the **Next Screens** node, you will be given a list of available screens to select for your next screen. Select from the list in the order you want your screens to appear.

To debug macro play errors, four icons exist on the **Host Terminal** toolbar. The icons are designed to help you step through a macro, where a step is defined as playing one action or recognizing one screen. The icons are **Step Into**, **Step Over**, **Step Return** and **Resume**.

Each of the icons have function keys associated with them. Depending on which view is active (either the **Macro Navigator** or the **Host Terminal**), the F5, F6, F7, and F8 keys will function differently. For instance, if the **Macro Navigator** is active, pressing F5 will do **Step Into**, and pressing F6 will do **Step Over**. If the **Host Terminal** is active, pressing F5 or F6 will send the AID key to the host. For more information about these and other macro related icons, see Macro related icons.

**Notes:**

1. HATS requires you to define screen recognition criteria for certain screens. These screens are:
   - First and last macro screen
   - Start and end of a loop
   - Macro extract
   - Macro prompt
2. Cut, copy, and paste functions are provided in the host terminal by using the following key combinations:
   - Cut: Ctrl-X or Shift-Delete.
   - Copy: Ctrl-C or Ctrl-Insert.
   - Paste: Ctrl-V or Shift-Insert.
3. Two new Video Terminal (VT) connection properties enable you to force the normal foreground or background of the VT Host Terminal to selected colors. This might be useful in a case where a VT application changes a foreground color to match the background color and thus renders the text unreadable on the Host Terminal.

   The settings can be added to the **Advanced** tab of the connection editor. The settings, which operate independently of each other, are **forceVTNormalForegroundToColor** and **forceVTNormalBackgroundToColor** .

   The format for the setting value is `redValue,greenValue,blueValue` where the values each range from 0 to 255. For example, to override the normal foreground color to be white, add this setting to the table of advanced settings:

   ```
   forceVTNormalForegroundToColor = 255,255,255
   ```

   These settings only affect the normal foreground and background colors. If any character attributes are set (bold, blink, underline, or reverse) or if any of the ANSI colors are set by the VT application, these settings are ignored.

   For more information about advanced connection settings, see “Configure optional, advanced connection settings” on page 135.

Table 4 on page 326 shows the macro icons that appear on the host terminal screen toolbar.

*Table 4. Macro icons*

| Icon | Description |
|---|---|
|  | Open a macro. Use the drop-down list to select a macro. |
|  | Play a macro. Use the drop-down list to select a macro. |
|  | Start recording a macro |
|  | Stop recording a macro |
|  | Save the recorded macro |
|  | Step Into (F5) the macro for debugging purposes |
|  | Step Over (F6) the macro for debugging purposes |
|  | Step Return (F7) in the macro for debugging purposes |
|  | Resume (F8) playback of the macro |
|  | Define the screen recognition criteria for a macro screen |
|  | Add a prompt action into the current macro (enabled only when recording a macro) |
|  | Add an extract action into the current macro (enabled only when recording a macro) |
|  | Record a loop into the current macro (enabled only when recording a macro) |
|  | Add prompt actions into the current macro for all fields on the screen (enabled only when recording a macro) |
|  | Add extract actions into the current macro for all fields on the screen (enabled only when recording a macro) |
|  | Show or hide the keypad in the host terminal window |
|  | Show or hide the textual Operator Information Area (OIA) |

*Macro related icons:* Below is a list of macro related icons on the **Host Terminal** toolbar:

**Record macro**

You can record macros in HATS Toolkit using the HATS host terminal. On the HATS host terminal screen, click the **Record macro** icon. The Record Macro wizard appears and enables you to select the target project, name the macro, give it a description, and see where the macro is saved. The option, **Create screen captures automatically while recording**, lets you specify that a screen capture automatically be created for every screen navigated while recording the macro. These screen captures can then be

used within the Visual Macro Editor if you must later make changes to the macro. This option is selected by default. Click **Finish** when you have specified these items.

Next the **Define Screen Recognition Criteria** window appears because HATS forces the first screen to have explicitly defined screen recognition. You can then use the HATS host terminal screen to navigate through the host application to any screen.

If you have a macro already open within the HATS host terminal, the **Record macro** icon will ask if you want to record a new macro or append to the open macro. If you append to the open macro, then macro recording will begin, and your actions will be appended to the current macro. The location within the macro where recording begins depends on what is selected in the **Macro Navigator** tree on the left, and whether the current terminal screen matches the screen recognition criteria of the selected node. Specifically:

- If a screen node or an action node is selected in the **Macro Navigator**, and the current host terminal screen matches the recognition criteria of the selected screen, the new actions will be added at the beginning of the selected node. If an action was selected within the matching screen node, then recording will begin immediately following the selected action within the matching screen.
- If a screen node or an action node is selected in the **Macro Navigator** window, and the host terminal screen does not match the recognition criteria of the selected screen, then a new screen node will be added to the **Next Screens** of the selected node, and recording will begin at the new screen node.
- If the high level node is selected in the **Macro Navigator** window which corresponds to the macro name, then HATS will search through all of the screen nodes and try to find a match for the current terminal screens. If a match is found, recording will begin at the beginning of the matching screen. If a match is not found, then a new screen node will be added to the **Next Screens** of the last screen and recording will begin at the new screen node.

Do not create a macro whose only step is to insert an extract or add a prompt. If you do this, when the user supplies a value, the page will not change, and there will be no way for the user to move to another page. If a macro doesn't go back to the host, it will sit and time out.

**Note:** If you intend to create an Integration Object from a macro, ensure that you follow these requirements when naming the macro:

- The name of the macro must consist of only letters, digits, or underscore characters.
- The first character of the name cannot be a numeral.
- Double byte (fullwidth) characters are not allowed.

**Stop Macro**

When you are finished recording your macro, click the **Stop Macro** icon.

**Save Macro**

Once you have stopped recording your macro, you will need to save it by clicking the **Save Macro** icon.

**Step Into (F5)**

Clicking the **Step Into (F5)** icon allows you to step from a screen to its first

enclosed action, and from action to action. Using the F5 key on your keyboard will perform the same function. This is used for debugging purposes.

**Step Over (F6)**
Clicking the **Step Over (F6)** allows you to go from screen to screen (performing all intervening actions). Using the F6 key on your keyboard will perform the same function. This is used for debugging proposes.

**Step Return (F7)**
Clicking the **Step Return (F7)** allows you to finish playing the actions under a screen and go to the next screen. Using the F7 key on your keyboard will perform the same function. This is used for debugging proposes.

**Resume (F8)**
To resume the playback of your macro, you can either click the **Resume (F8)** icon or press the F8 key on your keyboard. This can be used to run the rest of your macro if you are currently stepping through it.

**Define Screen Recognition Criteria**

You can set screen recognition criteria that HATS uses to match host screens. Host screens can be recognized by any combination of criteria including how many input fields or total fields are on the screen, the coordinates of the cursor's position, and text strings on the screen within a defined rectangle or anywhere on the screen. For more information see "Screen Recognition Criteria or Begin Screen" on page 148.

**Add Prompt Action**

If you want the macro to prompt the user for information, click **Add Prompt Action** to display the Add Prompt Action wizard. You can give the prompt a name and a default value. If the information the user provides, such as a password, should not be displayed on the host screen, click the **Password protect input** check box.

**Note:** Default values that you specify for prompts are stored in macro files unencrypted. The default values display in the clear when you edit prompts using the macro editors. Therefore, while using a prompt to specify a password is an appropriate thing to do, for security reasons you should not specify a default value for the password.

The **Row** and **Column** fields of the **Position** section of the wizard define where on the host screen the prompt information provided by the user is placed. If you place your cursor at a location on the host screen, such as the field for a password, before you begin recording the macro, the **Row** and **Column** fields are filled with those values. The **Handle Macro Prompt** section of the wizard enables you to determine how the prompt is processed. You can select one of the following radio buttons:

**Show handler**
For HATS Web projects, you can select a .jsp file to prompt the user for the necessary information, and include a button for the user to submit the information. A default macro handler, named default.jsp, is shipped with HATS. You can find this file by clicking the **HATS Projects** view of the HATS Toolkit, expanding the project name, and expanding **Web Content > Macro Event Handlers**. If you want to create your own handler, ensure that you return control to the HATS runtime.

**Notes:**

1. Integration Objects ignore the selected .jsp handler. Instead, an input page is created for the Integration Object, and a prompt for the value is placed in that input page. The generated output page copies the value supplied by the input page into the Integration Object before the Integration Object is run.
2. The **Show handler** option is disabled when the macro extracts from a plane other than the text plane.

**Note:**

For HATS rich client projects, you can specify a custom macro handler, or browse to select from the list of custom macro handlers defined in the rich client project, to prompt the user for the necessary information, and include a button for the user to submit the information. A default macro handler, named DefaultMacroHandler, is shipped with HATS. You can find this file in the **HATS Projects** view, expanding the project name, and expanding **Rich Client Content > Macro Event Handlers**. If you want to create your own handler, ensure that you return control to the HATS runtime by calling the createButtonArea() and createButtons() methods in the render() method of your custom macro handler. These methods are called in the default implementation of the RcpMacroHandler.render() method. For more information about RcpMacroHandler, see the HATS RCP API Reference section in the HATS Knowledge Center at http://www.ibm.com/support/knowledgecenter/SSXKAY_9.7.0?topic=/com.ibm.hats.doc/doc/rcpjavadoc/index.html.

**Set prompt to string**

If you know what value should be returned from a prompt, you can enter that string in the **String** field.

**Set prompt to global variable**

If you want the value of the prompt to be provided by a global variable, enter a name for the global variable in the **Name** field or select an existing variable using the drop-down menu next to the **Global variable** field. If you click the **Advanced** button, you can specify whether your variable is shared or indexed. If it is an indexed variable, you also need to specify whether to show all indexes or a single index. For more information, see Chapter 12, “Interacting with global variables,” on page 337.

**Note:**

If a prompt value is based on a global variable set by an extract, and **promptall** is set to `true`, the extract action is not run before the prompts values are retrieved. Because of this, the global variable used by the prompt does not contain a value. If you use global variables with extracts and prompts, you should set **promptall** to `false`.

Integration Objects do not access global variables directly. Instead, the input and output pages for the Integration Object retrieve the global variable value, and set it into the

Integration Object before it is run. Only shared global variables can be accessed by Integration Objects.

**Set prompt to property from User List**
If you want the prompt to access a user list, select the **User Profile** from the drop-down list. The user profile is the key as to determining whether to use the `userid` or password as the **User List property**. For more information, see "User List" on page 144.

**Note:** User list prompts can only be used in connect macros.

**Use Web Express Logon**
If you have configured your HATS application to use web express logon, enter the prompt type as either the user ID or password in the **Prompt type** drop-down list and enter the application ID in the **Application ID** field.

**Note:** The **Prompt type** should be prefilled with the correct value.

Click **OK** when you have made your selections.

**Note:** For considerations when using bidirectional language support, see "Macro prompt and extract bidirectional options" on page 445.

**Add Extract Action**

If you want the macro to extract information from the host screen, select an area on the host screen then click **Add Extract Action** to display the Add Extract Action wizard. You can specify a name for the extract. The **Start row**, **Start column**, **End row**, and **End column** fields of the **Position** section of the wizard define from where on the host screen the information is extracted. When you mark a region of the host screen with a rectangle and then click **Add Extract Action**, the **Position** section fields are filled with the values when the Add Extract Action wizard is displayed.

You can specify the **Extraction Format** as either:

- **Extract this region as one string**

  Specifies that the extracted text should be saved as a single horizontal string of characters. This option is supported for Integration Objects, global variables and macro handlers.
- **Extract this region as a list of strings**

  Specifies that the extracted text should be saved as a vertical list of strings. This option is supported for global variables and macro handlers. For an Integration Object, a list of strings is treated as one continuous string.
- **Extract this region as a table:**

  Specifies that the extracted text should be saved as a table of horizontal and vertical strings, with rows and columns. This option is supported for Integration Objects and macro handlers. For a global variable, strings extracted as a table become one continuous string.

  **Table Extract Configuration**
  Use this page to format the columns of the table.
  - **Column name**

    Use this field to change the heading of the selected column.
  - **Expand column**

Expanding a column moves characters between columns. When you highlight a column, the **Left** button moves the last character on each line of the column to the left of the highlighted column to become the first character on each line in the highlighted column. The **Right** button moves the first character from each line of the column to the right of the highlighted column to become the last character on each line in the highlighted column.

- **Reduce column**

  Reducing a column moves characters between columns. When you highlight a column, the **Left** button moves the first character from each line of the highlighted column to become the last character on each line in the column to the left of the highlighted column. The **Right** button moves the last character from each line of the highlighted column to become the first character on each line in the column to the right of the highlighted column.

- **Merge**

  Use this button to merge two highlighted columns into a single column. The characters in the highlighted columns are joined into one column.

- **Divide**

  Use this button to divide a highlighted column into two separate columns. The characters in the highlighted column are divided equally between the two new columns. If there are an uneven number of characters, the left column will contain one more character than the right column.

**Note:**

For considerations when using this function with DBCS support see "Recording a macro" on page 463.

The **Handle Macro Extract** section of the wizard enables you to determine how the prompt is processed. You can select the following check boxes:

**Show handler**

For HATS Web applications, you can select a .jsp file to display the extracted information to the user. A default macro handler is shipped with HATS, and it is named default.jsp. You can find this in the **HATS Projects** view, expanding the project name, and expanding **Web Content > Macro Event Handlers**. If you want to create your own handler, ensure that you return control to the HATS runtime.

**Note:** Integration Objects do not use this option. Instead, the output page will retrieve the extracted data from the Integration Object and display them.

For HATS rich client projects, you can specify a custom macro handler, or browse to select from the list of custom macro handlers defined in the rich client project, to prompt the user for the necessary information, and include a button for the user to submit the information. A default macro handler is shipped with HATS, and it is named DefaultMacroHandler. You can find this file in the **HATS Projects** view, expanding the project name, and expanding

**Rich Content > Macro Event Handlers**. If you want to create your own handler, ensure that you return control to the HATS runtime by calling the createButtonArea() and createButtons() methods in the render() method of your custom macro handler. These methods are called in the default implementation of the RcpMacroHandler.render() method. For more information about RcpMacroHandler, see the HATS RCP API Reference section in the HATS Knowledge Center at http://www.ibm.com/support/knowledgecenter/SSXKAY_9.7.0?topic=/com.ibm.hats.doc/doc/rcpjavadoc/index.html.

**Save as global variable**

You can enter a name for the global variable in the **Name** field or select an existing variable using the drop-down menu. If you select an existing global variable in the **Name** field, you must specify how to handle the existing variable by selecting one of the following radio buttons:

- Overwrite the existing value with this new value
- Overwrite the existing value with this new value, starting at the specific index.
- Append this new value to the end of the existing value.
- Insert this new value into the existing value, at the specific index.

You can also specify whether this variable is shared by selecting the **Shared** check box.

**Note:** If you extract a value and assign it to a global variable set by an extract, and you plan to use the global variable value for a prompt, you should set **promptall** to `false`. When **promptall** is set to `true`, the extract action is not run before the prompt values are retrieved. Because of this, the global variable used by the prompt does not contain a value.

Integration Objects do not directly extract to global variables. Instead, the output page for the Integration Object retrieves the data from the Integration Object after it has run, and then sets the global variables. Remember only shared global variables can be accessed by Integration Objects.

Click **OK** when you have made your selections.

**Note:** For considerations when using bidirectional language support, see "Macro prompt and extract bidirectional options" on page 445.

**Record a Loop**

If you want to record a loop in your macro, click **Record a Loop** then navigate to the screen where the loop will begin. Click **Next** to define the screen recognition criteria for the first screen in the Click **Next** to define additional criteria or **Finish** to return to the main terminal. At the main terminal, perform the actions which will be run during each cycle of the loop then press **Next**. Determine how your loop will end by selecting either **End when a unique screen is recognized** or **End after a fixed**

**number or iterations** and enter the number of loops below. Click **Next** then navigate to the final screen of the loop and press **Finish** to complete the loop.

**Note:** Significant emphasis should be placed on the screen recognition criteria for the starting and ending screens of the loop. In some cases, you may have a starting screen that is almost identical to the ending screen. You must be careful in distinguishing these screens from one another.

**Add Prompt Actions for All Fields**

If you select this icon, the **Insert All Fields from Global Variables** window will open. There you can select a **Stored Screen name** from the drop-down list. You also have the option of creating a screen capture by selecting the **Create a Screen Capture** check box. Selecting the **Shared** check box will take the global variable from the shared list.

**Notes:**

1. Default values that you specify for prompts are stored in macro files unencrypted. The default values display in the clear when you edit prompts in the host terminal macro editor or the advanced macro editor. Therefore, while using a prompt to specify a password is an appropriate thing to do, for security reasons you should not specify a default value for the password.
2. If you use this option with Integration Objects, HATS will add fields to the Integration Object for each defined global variable. You must write code to set the global variable values and copy them into the Integration Object before it is run. This can be done in the .jsp page that runs the Integration Object. For more information, see "Insert Integration Object Properties" on page 349. Remember only shared global variables can be accessed by Integration Objects.

**Add Extract Actions for All Fields**

If you select this icon, the **Extract All Fields from Global Variables** window will open. There you can select a **Stored Screen name** from the drop-down list. which will be used as a prefix for the global variables created for all fields. If the text of the protected fields should be grabbed at runtime, select the **All fields** option. If the text for the protected fields will be static in the transformation JSP, select the **Input fields** option. Selecting the **Shared** check box will take the global variable from the shared list.

**Note:** If you use this option with Integration Objects, HATS will add fields to the Integration Object for each defined global variable. You must write code to copy the data from the Integration Object into shared global variables after the Integration Object is run. This can be done in the .jsp page that runs the Integration Object. For more information, see "Insert Integration Object Properties" on page 349. Remember only shared global variables can be accessed by Integration Objects.

**Show/Hide Keypad**

This icon allows you to either show the terminal keypad or hide the terminal keypad.

**Show Textual OIA**

The **Show Textual OIA** (Operator Information Area) icon allows you to see the exact row and column position of the cursor on your terminal screen. It also tells you whether the input is inhibited or not inhibited.

**Host Screen Preview:** To see what the current host screen would look like if transformed by the server, click the **Host Screen Preview** tab at the top of the Host Terminal window. The preview screen will display the host screen you are on as a Web page.

The host screen preview first applies any screen customization defined for the present screen. If there are no screen customizations defined for the current screen, the project level template and default rendering rules are applied to the screen.

The host screen preview applies the order of screen customizations listed in the **Events** tab of your **Project Settings**. As soon as it finds a screen recognition match is shows you the screen transformed by the transformation defined in that screen customization (if any). If there is not match, then the default transformation is shown.

The host screen preview will not process any screen customizations defined in the next screen list. Next screens are not processed in the host screen preview. For more information, see "Next Screen" on page 167.

**Note:** The **Host Screen Preview** tab is only seen if your current host terminal is associated with the default connection. If you open a host terminal on a background connection, you will not see this preview.

### Advanced Editor

You can also click **Advanced Editor** on the **Overview** tab to launch the advanced features of the HATS macro editor, and modify settings for the macro. A separate window opens for the macro editor which has the following tabs:

- Macro
- Screens
- Links
- Variables

For more information about the advanced editor features of the macro editor, see *HATS Advanced Macro Guide*.

## Prompts and Extracts

The **Prompts and Extracts** tab of the macro editor lists the configured macro prompts and extracts, and how they are handled when the macro is played. You can edit the HATS-specific properties of a prompt or an extract by selecting it in the table and clicking **Edit**.

## Source

The **Source** tab displays the tags in the macro-name.hma file for all the attributes and values for the macro, where macro-name is the name you gave to the macro when you created it or imported it. As you make changes on other tabs in the project editor, the tags and attributes displayed in the source file change to match. You can also make changes to the tags and attributes in the source file, and they are reflected on the appropriate tabs of the macro editor.

Prompts and extracts appear in the macro source file in the order that you recorded them. In addition, when you open the macro in the editor, the cursor appears on the same line it was on when you last saved the macro, instead of appearing at the top of the macro file.

By default, content assistance is enabled for all macros in the project. While editing a macro on the **Source** tab, press Crtl+Space to invoke content assistance. You can configure which macros in the product provide content assistance. For instructions, see "Macro Content Assistance" on page 124.

## Working with macro errors

Errors in a macro are summarized in the **Problems** view, whether you are using the macro editor, text editor, or XML editor. The source code of the macro displays a marker, a red circle with a white X in it, which appears on any lines containing an error.

The macro can be saved even when it contains errors, enabling you to find and fix the errors at a later time, if necessary. However, macros containing errors cannot be played at runtime or in the host terminal.

## Importing a macro

You can also import macros recorded with other programs, such as the IBM WebSphere Host Publisher or IBM Host On-Demand. To import these macros, select **File > Import > HATS > Host Publisher/HOD Macro** and click **Next** to display the **Import a Host Publisher/HOD Macro** dialog. Click **Add** and navigate to the location of the macro on the file system.

**Notes:**

1. A Host On-Demand macro with an exit screen that performs an action that submits an AID key to the host may cause timing problems when running in the HATS runtime. When the macro finishes running, HATS runtime renders the exit screen. But the AID key causes the host to return another (trailing) screen which is not rendered. To correct this timing problem, do one of the following:
   - Use the automatic refresh function to display the trailing screen. See "Automatic Disconnect and Refresh `Web-only` " on page 109.
   - Use the Visual Macro Editor (VME) to change the exit screen for the macro from the original exit screen to the trailing screen. The integrated terminal of the VME works like Host On-Demand and can be used to play the macro and display the trailing screen. Use the VME to add the trailing screen to the macro, change its properties to indicate it is the exit screen, and change the properties of the original exit screen to remove the exit screen indication. For more information, see the Visual Macro Editor chapter in the *HATS Advanced Macro Guide*.
2. Host Publisher macros with fixed iteration loops continually recognize the same screen and perform different actions. In HATS, you cannot create a screen customization to recognize the same screen a second time and perform a different action than the first time it was recognized. If you attempt to use a Host Publisher macro with fixed iteration looping, your project might go into an infinite loop. Host Publisher macros with fixed iteration looping can be identified by looking at the source code for the macro. The macro contains `customreco` tags with ID attributes of `HPubFixedIterationLoop` and `custom` tags with ID attributes of `HPubIncrementLoop`.

## Exporting a macro

You can export HATS macros as well as IBM Host On-Demand macros. To export these macros, select **File > Export > HATS > HATS Macro** or **File > Export > HATS > HOD Macro** and click **Next**.

Select the **Macro files to export** along with the destination. You can elect to overwrite the existing resources by selecting the **Overwrite existing resources with warning** check box.

Click **Finish** when done.

**Note:**

For considerations when using DBCS support see "Exporting a macro" on page 463.

## Macro hints and tips

### Preventing an infinite loop

Running a macro is one of the actions that you can specify to be performed when a host screen matches a screen customization's recognition criteria. When you record a macro, be sure that the final screen is not the screen that is recognized by the screen customization that has the macro defined as the action. If the recognized screen is the final screen of the macro, a loop will be created.

### Handling transient screens

You can define transient screens along with non-transient screens for your macros. However at runtime, you may see one of your transient screens get recognized unexpectedly.

When comparing macro next screen descriptors, non-transient screen descriptors are compared first, then transient screen descriptors are compared. If the current screen information is updated after comparison of non-transient screen descriptors begins, the updated screen information is used for transient screen descriptor comparison. Therefore, it is possible that if a screen is locked (OIA is input-inhibited) when non-transient screen comparison begins, and then becomes unlocked, with no other changes, before transient screen comparison begins, this can cause non-transient screens to not be recognized, but then subsequently transient screens can be recognized, which is not the expected behavior.

To avoid this unexpected behavior, you can control whether the same screen information is used when comparing non-transient screen and transient screen descriptors. To do this, add the advanced connection setting, **screenRecoUseOIASnapshot**, and set its value to **true**. For how to set advanced connection settings, see "Configure optional, advanced connection settings" on page 135.

# Chapter 12. Interacting with global variables

A global variable contains a value that can be used to pass information from one HATS object to another. For example, you can extract information from several locations on a host screen, perform calculations, and insert the result on the current screen or a future one. You can build up an array of strings from one or more host screens and insert them into a transformation. You can extract a string that a user enters into a field on the GUI and use it elsewhere.

Global variables that are only visible within the scope of a single HATS application are called local global variables. Global variables that are visible to and can be used by all HATS Web applications in the same .ear file, or by all HATS rich client applications running in the same rich client environment, are called shared global variables. Any time you use or create a global variable, you must specify whether it is local or shared. The **Advanced** settings section allows you to specify sharing. The default setting for all global variables is **local**.

**Note:** Two global variables with the same name can coexist if one is local and the other is shared.

For HATS Web applications you can share global variables between applications within the same .ear file during a single HTTP session. For HATS rich client applications you can share global variables between applications running in the same rich client environment. If the rich client environment is closed and restarted, previously shared global variable values are not preserved.

A global variable can contain a numeric value, a string, or an indexed array of strings. If you use a global variable to contain an array of strings, you can specify for any action whether you want to use the entire array, a particular index, or all the values starting at a particular index. All operations on global variables are case-sensitive. Do not use names beginning with `HATS`, `hats` or `Hats` for global variables.

You can set the value of a global variable in these ways:

- With a **Set global variable** or **Extract global variable** action on a screen event or application event
- By prompting the user for a value in a screen transformation
- By prompting the user for a value while running a macro
- By setting a value in the macro source code
- By setting the value in a business logic program.

After a global variable has a value, you can use that value in the following ways:

- To calculate the value of another global variable, in a **Set global variable** action
- To write the value to a host screen, using an **Insert Data** action
- To insert the value into a transformation or a template, using the **Insert Global Variable** menu item
- To pass the value to a macro
- To use the value in business logic
- To use as a criterion in screen recognition
-

If you insert a global variable into a host screen, you must list this action before applying a transformation, to ensure that the global variable will appear on the GUI created from the host screen. It might make sense however to insert a global variable into the host screen after a transformation returns and before the host screen is sent back to the host. See “Actions” on page 154 for more information about specifying actions for screen events. For information about inserting global variables into transformations and templates, see “Insert Global Variable” on page 182.

Global variables can be used with prompt and extract macros either to provide a value for a prompt or to store a value extracted from the host screen. Global variables can be used in conjunction with macros to combine multiple host screens in a single GUI. See Chapter 11, “Macros and host terminal,” on page 323 and Chapter 14, “Combining screens,” on page 351 for more information about using global variables with macros.

You can override global variables to pass data from the GUI to the application. Local and shared global variables can be overridden. For more information, see “Global variable overrides” on page 115.

They may also be used for screen recognition. For more information, see “Screen Recognition Criteria or Begin Screen” on page 148.

For information about using global variables in business logic, see the *HATS Web Application Programmer's Guide* or the *HATS Rich Client Platform Programmer's Guide*, depending on your application environment.

If you want to use a global variable to accumulate strings or a numeric value from several screens, you can initialize it by adding a **Set global variable** action to the start or connect application event.

Double click on the **Project Settings** of your HATS project in the **HATS Projects** view. Go to the **Events** tab.

To set global variables in the connect event, double-click **Connect** to open the event editor. Then you can add the global variable on the **Actions** tab. This way, the global variable is linked with the action being performed.

The same can be done for the **Start** event.

You can remove local and shared global variables using the **Remove global variable** action on a screen event or application event. For more information, see “Remove global variable action” on page 159.

## Renaming global variables

You can rename each instance of a global variable. To get a list of all global variables found in a particular HATS project take the following steps:

1. Select **Window > Show View > HATS Global Variables** from the HATS menu bar. The **HATS Global Variables** view will be placed right below the **Palette** view.
2. Click on a project in the **HATS Projects** view to populate the **HATS Global Variables** view with all the global variables found in the selected project's macros and events. If you select a different project, the **HATS Global Variables**

view will be updated. The content is refreshed when a macro or event of the selected project is added, deleted, or edited.

3. You can rename a global variable through the **HATS Global Variables** view by right clicking the global variable to edit the name. You can also double-click on the macros or events listed under each global variable to open them. When you create or edit the macros or events, the **HATS Global Variables** view will automatically be updated.

**Note:** Renaming global variables is only supported for global variables defined in macros and events; therefore, the **HATS Global Variables** view will only display those global variables. Renaming global variables will not rename the global variables in business logic, transformations or elsewhere in your HATS project.

## Differences between global variables and macro variables

In HATS, there are two main types of variables; global variables and macro variables. The differences between them are outlined here.

### Global variables

Global variables are variables created in HATS Toolkit and used by HATS projects. Global variables are stored outside of the macro script in the HATS .hma source file. They are maintained and updated by HATS runtime. There are two types of global variables:

**Local** A local global variable is one that is created within a HATS project, and is only visible to the project.

**Shared**
A shared global variable is one that is visible to and can be used by all the Web applications in an .ear file or by all rich client applications in a rich client environment.

Whether a global variable is considered local or shared depends on whether the **Shared** check box in the GUI is selected when the global variable is created, or whether the value of the shared attribute of a **set**, **prompt**, or **extract** tag is specified as **yes** or **no** in the HATS .hma source file.

### Macro variables

Macro variables are variables created in the Visual Macro Editor or the Advanced Macro Editor. Macro variables are internal to macros. They are not used outside of macros. Macro variables are created, stored, and used by the macro engine, and listed in the macro script. For more information about macro variables, see the chapter, Variables and imported Java classes, in the *HATS Advanced Macro Guide*.

In the source of a HATS macro (.hma) file that uses HATS prompts and extracts for global variables, the prompts and extracts are contained in the source file before the macro script. The macro script syntax is enclosed by the <HAScript> and </HAScript> tags.

# Chapter 13. Using Integration Objects

This chapter introduces Integration Objects, which are Java beans that encapsulate interactions with a host application. Integration Objects are supported only by HATS Web applications.

**Note:** Global variables and templates are not supported for Integration Objects used in HATS portlets.

If you have used IBM WebSphere Host Publisher, you are already familiar with most aspects of Integration Objects, but you will need to learn about how Integration Objects are used in HATS. You do not need to be an experienced Host Publisher developer to create and use Integration Objects in HATS.

You may need to use Integration Objects if any of these statements are true:

- Your application connects to more than one host application.
- You want to encapsulate your host interactions into Enterprise JavaBeans (EJB) applications or Web services. These processes are described in the *HATS Web Application Programmer's Guide*.
- You want to build Web pages based on the inputs and outputs of a macro.

When you have created an Integration Object, you can use it in many ways:

- Run the Integration Object from business logic.
- Build Model 1, Struts or JSF Web pages based on the inputs and outputs of the Integration Object.
- Chain Integration Objects allowing them to be run in sequence.
- Run the Forward to URL action to pass control from a transformation-oriented HATS project to a JSP that invokes one or more chained Integration Objects.

Refer to the *HATS Web Application Programmer's Guide* for information about other advanced tasks that can be performed with Integration Objects. Some of the tasks discussed are:

- Using Integration Objects to create either traditional (WSDL-based) or RESTful Web services that can be used to create service-oriented architecture (SOA) assets that provide standard programming interfaces to business logic and transactions contained within host applications.
- Creating a HATS Enterprise JavaBeans (EJB) application by creating Integration Objects in a HATS EJB project.
- Modifying the Java code of the Integration Object.
- Running Integration Objects in standard portlets.

## Creating an Integration Object

An Integration Object is created from a macro. Follow the steps described in Chapter 11, "Macros and host terminal," on page 323 to record a macro.

Right click on the macro in the HATS Projects view and select Create Integration Object. The Integration Object will have the same name as the macro. You can also set a preference in HATS to automatically generate an Integration Object when a macro is recorded. Click **Window > Preferences**, expand **HATS** in the tree on the

left, and click **Integration Object**. Select the box labeled **Automatically generate Integration Object when saving macro**, then click **Apply**.

When naming a macro used to create an Integration Object, ensure that you follow these requirements:

- The name of the macro must consist of only letters, digits, or underscore characters.
- The first character of the name cannot be a numeral.

**Notes:**

1. Automatically generated Integration Objects are always non-chained.
2. For information about creating Integration Objects when using DBCS support see "Creating an Integration Object" on page 464.

Integration Objects cannot get created from chained macros. Integration Object chaining is quite different from macro chaining, in that each Integration Object in a chain is run to completion before the next Integration Object in the chain takes control. Macro chaining, using the HATS **play macro** action, terminates the current macro (the one in which the **play macro** action occurs) and begins to process the specified macro screen of the target macro. If you import a Host On-Demand macro that contains the `<playmacro name="chainedMacroName"/>` tag, you can only use this macro in the HATS **play macro** and **perform macro transaction** actions. See *HATS Advanced Macro Guide* for more information about macro chaining.

If a macro is changed using the macro editor, after an Integration Object has been created from that particular macro, the Integration Object has to be regenerated to accurately reflect the change. If the **Automatically generate Integration Object when saving macro** preference is selected, the Integration Object will be updated automatically.

To see a list of Integration Objects which you have created, click the **HATS Projects** view tab of the HATS Toolkit and expand the project name, then go to **Source > IntegrationObject** and double click the folder to view the contents.

**Note:** The **IntegrationObject** folder under **Source** is only visible after creating your first Integration Object.

## Integration Object chaining

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

When you right click on the macro to create the Integration Object, select Create Chained Integration Object. Then specify whether the Integration Object is the first, middle, or last in the chain. There can be more than one middle Integration Object in a chain. Enter the labels that identify the start and stop states for this Integration Object. There is no start state label for an Integration Object that is first in its chain, and no stop state label for an Integration Object that is last in its chain. State labels are required to ensure that the Integration Objects in a chain are processed in the correct order, with the correct data passed each time.

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

## Deciding when to use Integration Object chaining

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

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

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

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

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

## Using Integration Object chaining

An Integration Object in a chain leaves the connection in a state (at a particular screen). After the Integration Object finishes running, another Integration Object that begins in that state (that is, at that screen) can run.

You can use chaining to break up a complex application into multiple tasks, each task represented by an Integration Object. You must ensure that the order of the Integration Objects is correct.

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

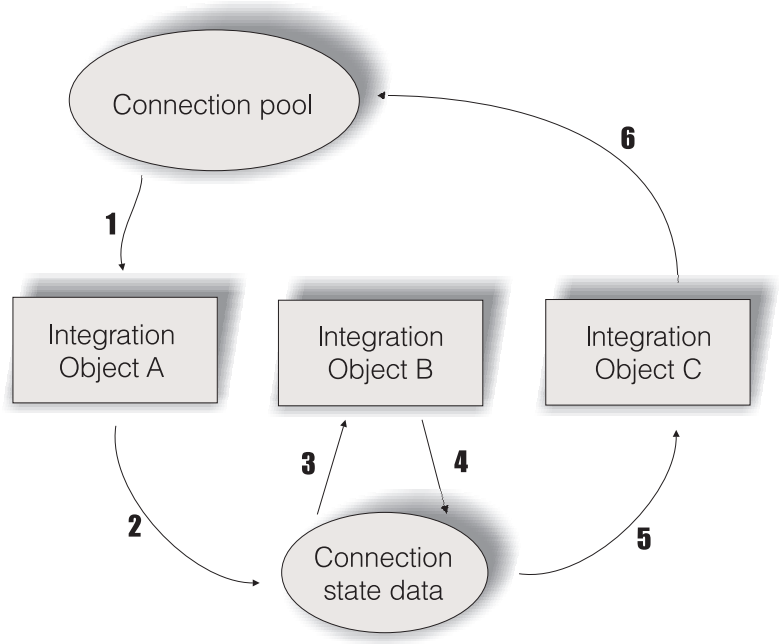



*Figure 76. Connection lifetime with chaining*

Figure 76 depicts the lifetime of a connection throughout the execution of three Integration Objects. Integration Object A is configured as first, Integration Object B as middle, and Integration Object C as last. Connection state data represents the connection and the state label of the last Integration Object run.

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

**Note:** Settings for the checkin screen are made by going to the **Macro** tab of your connection settings. You must enable pooling in order for the checkin screen setting to be available. For more information, see "Connection editor" on page 132.

To build an application using Integration Object chaining with HATS, you must first build the Integration Objects and then create your Web pages from the Integration Objects.

Follow these steps to create your application:

1. Record macros.
2. Create chained Integrations Objects from each macro.
3. Create Web pages for each Integration Object.

**Record a Macro**

You can record macros in HATS Toolkit using the HATS host terminal. On the HATS host terminal screen, click the **Record Macro** icon. For information about recording macros, see "Macro related icons" on page 326.

Repeat these steps for each Integration Object you want to create.

**Create Integrations Objects from each macro**

Once all of your macros are created, you can go to the **HATS Project View** tab of the HATS Toolkit and expanding the project name, and open the **Macros** folder.

Right-click on each individual macro and select **Create Chained Integration Object**. You will have the option of selecting the position in the object chain to either **First**, **Middle** or **Last**. If you select **First**, the state will be the stop state during Integration Object chaining configuration. If you select **Middle**, the state can either be the start state, or the stop state. The **Last** position of the object chain uses the start state.

This needs to be done for all the Integration Objects which are part of the chain.

**Create Web pages from each Integration Object.**

Once you have created your Integration Objects, you can then create your Web pages. For information about creating Web pages from Integration Objects, see "Building Web pages from an Integration Object" on page 346.

## Debugging applications that use Integration Object chaining

While Integration Object chaining is a powerful tool for modeling the most complex host applications, take care when assembling your applications. There is a risk of creating a Java EE application with Integration Objects that are invoked out of order; therefore, you should be aware of the errors you get on the server when this happens.

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

**HPS5075 Received STATE_PLAY_ERROR while playing macro in file zzz.macro**

This error occurs when the macro fails to play because it cannot match the current screen. This might happen if an Integration Object in a chain is invoked in the correct order (a connection in a specific chain was found for it to use), but the macro itself failed to play. It can be a problem if your

start and stop state labels are all the same, or at least are the same for two Integration Objects, and you invoke the Integration Objects out of logical order.

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

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

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

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

## Building Web pages from an Integration Object

You can create a JSP from any Integration Object There are three approaches to building these pages; they are called Model 1, JSF, and Struts. The following sections describe these three approaches and how to create pages using them.

### Create Model 1 Web pages

The traditional HATS approach to building a JSP is called Model 1.

**Note:** This is the only approach supported for HATS portlets.

In this model, a single JSP contains:

- The information to be presented to the user
- Formatting tags that specify how the information is displayed
- Control logic that controls which page is called next

In the **HATS Projects** view, right click on the name of an Integration Object and select **Create Model 1 Web Pages**. Specify the input and output page names along with the destination, or accept the default names created from the Integration Object name. You can also specify **Page generation options** by selecting **Append to existing files** or **Overwrite existing files**. Click **Next** after completing your selections.

When you create Model 1 Web pages, you define two pages: an input page, which gathers the data required by the Integration Object, and an output page, which presents the results after the Integration Object is run.

On the **Define inputs** page, you can define an input control for each input property. HTML controls are used as defaults. To change a control, select an input property and click **Edit**. You have the option to render input properties using either HTML controls or Dojo widgets. You can mix HTML controls and Dojo

widgets on the same Model 1 web page. You can add the leading text for the control and, where necessary, configure the selected control. Instead of using an input control, you can elect to set the input property to the value of a global variable. You also have the option to store the input property value as a global variable (global variables used by Integration Objects are shared). When the input page is run, it does not pass control directly to the output page; it returns control to the HATS application allowing global variables to be set and other actions to be taken as needed.

**Notes:**

1. If your macro has no prompts, there will be no input page.
2. You can use Dojo widgets only in HATS Web projects, not in HATS portlet projects.

On the **Define outputs** page, you select the output properties of the Integration Object that you want to display. To change a control, select an output property and click **Edit**. You have the option to render output properties using either HTML controls, Dojo widgets, or normal text. You can mix HTML controls and Dojo widgets on the same Model 1 Web page. You can add the leading text for the control. The output JSP contains code to initialize the Integration Object, to set the Integration Object properties from global variables, to run the Integration Object, and to display the Integration Object output.

The pages you create go in the Web Content/Model 1 Pages folder in the **HATS Projects** view. To edit a Web page and invoke the Rich Page Editor, double-click on the name of the page. To test your pages, right-click on the input page (or output page if there are no inputs) and select either **Debug on Server** or **Run on Server**.

**Note:** When using Model 1 Web pages for Integration Objects in HATS portlet projects, note the following considerations

- If you copy or rename a HATS standard portlet project, the Model 1 JSPs generated from a standalone or first in chain Integration Object refer to the wrong connection specification. You must modify the following statement:

  ```
  SignOn.setHPubStartPoolName("<new_project_name>/<default_connection>");
  ```

- When you generate a portlet project from a Web project, the Model 1 JSP pages previously generated from Integration Objects do not change. You must regenerate the pages, or make the changes manually.
- You can use Dojo widgets only in HATS Web projects, not in HATS portlet projects.

## Create Struts Web pages

HATS enables you to use Struts 2 to build Web pages based on Integration Objects.

**Note:** This approach is not supported for HATS portlets.

In the **HATS Projects** view, right click on the name of an Integration Object and select **Create Struts Web Pages**. Specify names for input and output pages, or accept the default names created from the Integration Object name. Click **Finish**.

When you create Struts Web pages, HATS generates following new files:

**Struts Action class**
This class is the main class. It instantiates the Integration Object, sets the Integration Objects properties using values in the ActionForm class, sets

the Integration Object's poolname, and runs the Integration Object. It then forwards to the output JSP, or if there is an error, then it returns to the HATS runtime and allows the HATS runtime to handle the error. The developer should not modify this class. This file is located in the Source/Struts/Actions folder in your project.

**ActionForm class**
This class contains setters and getters for each input in the input JSP. The input JSP submits to the Struts Servlet, which instantiates the ActionForm class and sets all of its properties based off the submitted input JSP form. This class is used by the Action class to set all of the Integration Objects properties. If you add or remove input properties from the input JSP, you must add or remove the matching setters and getters from this file. This file is located in the Source/Struts/ActionForms folder in your project.

**input JSP**
The input JSP contains a Struts HTML form that sends the request to the Struts Controller, which creates an ActionForm and forwards the request to appropriate Action class. If the Integration Object does not need any input, or where the inputs come from different sources, such as global variables, an empty input JSP is created containing only the HTML form. The input page is created in the Web Content/Struts Pages folder. To test your pages, run the link `http://<hostname>:<port>/<contextRoot>/<StrutsActionClassName>.action` in the browser.

**output JSP**
The output JSP displays the Integration Object's output properties. This page is similar to the Model 1 output JSP. This file is located in the Web Content/Struts Pages folder in your project.

HATS also adds or updates an action to the struts.xml file to associate the Action class with the input/output/error pages.

**Note:** When you create Struts pages, if Struts is not already installed in your project, HATS installs it and creates struts.xml file under source folder of your project.

## Create JSF Web pages

JavaServer Faces (JSF) is a framework for developing user interfaces (UI) for Web applications which run on a Java server. You can quickly build web applications by assembling reusable UI components in a page. Once you have created JSF Web pages, you can use tools such as the Rich Page Editor and the palette for modifications and editing.

**Note:** This approach is not supported for HATS portlets.

In the **HATS Projects** view, right click on the name of an Integration Object and select **Create JSF Web Pages**. Specify names for input and output pages as well as the destination, or accept the default names created from the Integration Object name. You can also **Overwrite existing files** by selecting the check box. Click **Next** after completing your selections.

When you create JSF Web pages, you define two pages: an input page, which gathers the data required by the Integration Object, and an output page, which presents the results after the Integration Object is run.

For the input page, you can define an input control for each input property. To change a control, select an input property and click **Edit**. You can add the leading text for the control and, where necessary, configure the selected control. Click **Next**.

For the output page, you select the output properties of the Integration Object that you want to display. To change a control, select an output property and click **Edit**. You can add the leading text for the control. Click **Finish**.

The pages you create go in the Web Content/JSF Pages folder in the **HATS Projects** view. To edit a Web page and invoke the Rich Page Editor, double-click on the name of the page. To test your pages, right-click on the input page (or output page if there are no inputs) and select either **Debug on Server** or **Run on Server**.

**Note:**

## BasicIOErrorPage.jsp and AdvancedIOErrorPage.jsp

The BasicIOErrorPage is the default error page used by JSPs that drive Integration Objects. If an error occurs, it will return basic information to the browser about the nature of the error. The backend connection is automatically discarded in this processing.

The AdvancedIOErrorPage.jsp allows the connection to be passed to the HATS entry servlet for a default transformation if the connection was created from the default connection definition. This processing allows you to interact with the backend connection and possibly determine why the error happened. The default transformation also gives you the ability to disconnect the connection. Basically, the AdvancedIOErrorPage.jsp gives you more control over the processing when an Integration Object encounters an error.

You can find these error pages in the **Navigator** view, in the Web Content folder of your project.

## Working with Integration Objects on JSPs

These HATS tools functions are applicable to working with Integration Objects on JSPs. You can select the items listed below using the **HATS Tools** menu on HATS Toolkit menu bar.

### Insert Integration Object Properties

You can add Integration Object input and output to your transformation JSP.

#### Input

Select **Insert Integration Object Properties > Input** when you want to add Integration Object input to a JSP. On the Select Integration Object class page, enter the class name of the Integration Object you want to use and the destination page that displays when the user submits the input form. Click **Next.** On the Define inputs page, to change a control, select an input property and click **Edit**. Here you can define the leading text, select an appropriate input control and configure it if necessary, and store the value as a global variable.

**Note:** Dojo widgets can only be used to render input properties when they are inserted into a Model 1 Web page. You can mix HTML controls and Dojo widgets on the same Model 1 Web page.

### Output

Select **Insert Integration Object Properties > Output** when you want to add Integration Object output to a JSP. On the Select Integration Object class page, enter the class name of the Integration Object you want to use and click the check box **Insert Integration Object transaction methods** to include the transaction methods that run Integration Objects. Click **Next.** On the Define outputs page, to change a control, select an output property and click **Edit**. Here you can define the leading text and select an appropriate output control.

**Note:** Dojo widgets can only be used to render output properties when they are inserted into a Model 1 Web page. You can mix HTML controls and Dojo widgets on the same Model 1 Web page.

A HATS global variable contains a value that can be used to pass information from one HATS object to another. If you want to pass Integration Object output data to another HATS object, you must add code to the output JSP used to drive the Integration Object. You should add the code after successfully running the Integration Object, for example:

```
<%
IOGV.setGlobalVariableString(session, "exampleGV",
exampleIO.getOutputData());
%>
```

In this example, the shared global variable named exampleGV is assigned the value of the OutputData property of the exampleIO Integration Object.

The global variable used in an Integration Object environment must be defined as a shared global variable. The code in the example creates the shared global variable if it does not already exist.

## Insert Forward to HATS Application

Select **Insert Forward to HATS Application** to add a button to your transformation that forwards JSP control information back to the HATS application, either Web application or standard portlet, that your user started. For example, you might want to return control to the HATS application after a JSP drives an Integration Object and performs a designated task. You can edit the attributes of this button in the **Design** view by highlighting the button, right-clicking, and selecting **Attributes**.

# Chapter 14. Combining screens

It is a common requirement to be able to gather output data from multiple host screens and present it in a single output page, or to provide a single input page, which then supplies multiple host screens with input data. HATS provides several methods for performing these operations. The different methods include the use of screen combinations, screen customizations, transformations, macros, global variables, and Integration Objects. Which method you use depends upon the operation to be performed and the location of the data.

The following sections summarize four operations and possible HATS methods for performing them. Combining operations and HATS methods for performing them are not limited to those summarized below.

## Combining contiguous output data

For this example, contiguous output data is defined as data all of which appears within one rectangular region on a host screen and appears at the same location on subsequent consecutive screens. An example of this would be a list of output data that does not all fit on one screen. In this example, using a terminal emulator a user must navigate forward through consecutive screens to view all of the data. No matter what screen the user views, the data appears within the same location on the display screen.

For this type of data, you can use a HATS screen combination to gather the whole multi-screen block of contiguous data into a single scrollable GUI for the user. A screen combination is one type of HATS screen event. For more information about screen events and creating and editing screen combinations see Chapter 7, “Working with screen events,” on page 147.

## Combining noncontiguous output data

For this example, noncontiguous output data is defined as output data that appears in multiple rectangular regions on a host screen and may appear at different locations on other, not necessarily consecutive, screens of the same host application. Examples of this would be any data that might be spread throughout multiple screens of a host application.

To combine this type of data, you can use HATS macros to automatically navigate through the host screens and use the macro **Add Extract Action** function either to gather data into global variables for later display in a single transformation or to use a HATS macro event handler transformation to display the data. For information about using HATS macros, see Chapter 11, “Macros and host terminal,” on page 323. For information about using HATS global variables, see Chapter 12, “Interacting with global variables,” on page 337.

## Combining output from multiple applications

This example assumes that output data, whether contiguous or noncontiguous as defined above, originates from multiple host applications. These host applications can be located on the same host system or different host systems.

To gather data from multiple host applications, you can use HATS Integration Objects and HATS support for background connections to multiple host applications. Integration Objects are Java beans that encapsulate interactions with a host application. They are generated from HATS macros and can be driven by Model 1 Web pages.

For each host application, create a background connection and connect, disconnect, and data macros. Create the data macros to accept the appropriate input, navigate through the screens of the application, and extract the data you want to display for the user. From each data macro, generate an Integration Object. From one of the Integration Objects, generate Model 1 Web pages that will pass input to the Integration Object and display the output extracted by the Integration Object. Then edit the output Web page and use the HATS tool to **Insert Integration Objects Properties** to run the other Integration Objects and add their output to the same Web page. While editing the output Web page you can organize the output from the multiple Integration Objects (that is, from the multiple host applications) in any way you like to combine the data for display to the user. For information about using HATS Integration Objects, see Chapter 13, “Using Integration Objects,” on page 341. For information about defining HATS connections, see Chapter 6, “Managing connections,” on page 131.

## Combining input for multiple screens

To provide a single input page, which then supplies multiple host screens in the same application with input data, you can use transformations, global variables, and macros. For example, use a transformation to collect the input data into global variables. One method is to use the **Global Variable Overrides** function, enabled on the **Other** tab of the Project Settings editor, to allow users to input data on the transformation that will initialize the global variables. See “Global variable overrides” on page 115 for more information. Then create a macro that navigates through multiple host screens using the values of the global variables as input, and put a button on the transformation the user can click to run the macro.

To supply input to multiple host applications, you can use Integration Objects running on multiple background connections similar to the method described in the section, “Combining output from multiple applications” on page 351. While building the macros from which the Integration Objects will be generated, you can use the **Add Prompt Action** function to create prompts for the input, or you can use the **Global Variable Overrides** function described above.

# Chapter 15. Enabling print support

As you develop a HATS project, you can establish a print session for the associated host application. When the HATS application is running in a production environment, a user of the application can print data or display data that is formatted for printing. Print support settings only apply to the default HATS connection.

When interacting directly with a host application, a user activates a physical printer to print data from the application. When interacting with a HATS application, the user does not activate a physical printer. Rather, a print file is generated, which can be displayed by the user . The print file can also be printed.

**Note:** If the print file is an Adobe PDF file and a PDF viewer (Adobe Acrobat Reader) is not installed, the user will be prompted to save the file to disk.

This chapter describes the process for enabling print support in your HATS project and for using print support in the HATS Toolkit environment and in the runtime environment for deployed applications.

## Configuring the host print session on 3270 hosts

Before setting up print support for your HATS project, make sure that you, or the system administrator, have performed the following configuration for the host print session:

- VTAM® configuration of a switched major node containing the display and printer logical units (LUs).
- HATS host print application connected to a TN3270E host port that provides access to the associated display and printer LUs used for printing. The display/printer LU association is specified and configured in the TN3270E server configuration.

Refer to the documentation for your 3270 host software for details on how to perform these steps.

## Defining print support for your project

For print support to be available for your project, you must obtain the host name and the port number for the Telnet server from the system administrator. Enter these values when you define the connection settings for your project.

### For 3270E connections

To define print support for 3270E connections, take the following steps:

- Verify with the system administrator that the print support on the host system is configured and active.

  **Note:** HATS only supports associated printing for 3270E connections. That is, the end user specifies a display LU or display LU pool name, and print jobs must be sent to the printer LU associated with the assigned display LU.
- In your project specify that the host type for your default connection is 3270E.

- Obtain the display LU or display LU pool name from the system administrator.
- In the editor for your 3270E default connection, click the **Advanced** tab and in the **LU or Pool Name** section, select one of the following options:
  - If using a display LU name, select **Prompt user** or select **Use a specified value** and enter the display LU name.

    **Note:** If you specify a display LU name, then in order for multiple users to use the application concurrently, they must override the display LU name at connection time. See "Connection parameter overrides" on page 106 for more information.
  - If using a display LU pool name, select **Use a specified value** and enter the display LU pool name.
- In the editor for your 3270E default connection, click the **Printing** tab and select the **Enable print support** check box .
- Select the type of printing scenario you want to configure. You can select from three predefined starter scenarios or configure your own using advanced Host On-Demand printer session properties. The three starter scenarios are:
  - Default printing (Adobe PDF format)
  - Basic text files (Plain text format)
  - Direct to server-attached default Windows printer **Web-only**

    or

    Direct to default Windows printer **RCP-only**
- For more information see "Print settings for 3270E connections" on page 135.
- If running on a VM system, identify the printer to use for print jobs.

When a user of the HATS application issues a command to print files, the host application sends a print job to the printer LU and the HATS runtime converts the print job to the format configured for the connection. Once the file is formatted, the user can click **View Print Jobs** to see a list of queued print jobs. For Web applications this is a button on the application keypad. For rich client applications it is an item on the pop-up menu in the Applications view.

**Note:** Each user has a separate list of available print jobs. Jobs printed by one user are not visible to another user.

Jobs in the queue can be opened or deleted by clicking the appropriate icons. For example, if open is selected for a PDF file and a viewer program is installed, such as Adobe Acrobat, the viewer program will open with the specified file. The file can be printed or saved at this point. If for example a PDF viewer is not installed, the user is prompted to save the file (as a .pdf file). The saved file can be opened from another system that has a PDF viewer program installed.

**Notes:**

1. For HATS Web applications, the print files are stored on the WebSphere Application Server system. If you terminate the HATS application, you will lose all print files in the queue.
2. For HATS rich client applications, the print files are stored on the rich client workstation. The user can select whether to delete the files when the rich client environment is closed. See "Print preferences" on page 81 for how this is done.
3. If the application has been configured to send the print jobs directly to a printer, the print files are not stored anywhere. The user cannot view or delete the jobs from this panel, but can see that they were received until they have finished printing.

4. Adobe Reader 9 is required to view PDFs created for 3270E connections that use Japanese host code pages 1390 or 1399. To create PDFs files that contain no JIS2004 characters and can be viewed with Adobe Reader 8, you must disable JIS2004 support for the 3270E connection. For how to do this, see "Disabling JIS2004 support for code pages 1390 and 1399" on page 434.

## Using the Host On-Demand PDT compiler

The Host On-Demand PDT compiler is supplied as a convenience for users who need to create new Host On-Demand Printer Definition Table (PDT) files for 3270E host printing. PDT files are not typically needed, but can be used if necessary to create binary print output files for specific non-Windows customized printer types. PDT files are created by first creating or modifying a source Printer Definition File (PDF) and then compiling it into a binary PDT file. PDT files are not used for printing to Windows printers.

To create a new PDT file, copy your source PDF into the <*RationalSDP_install_directory*>\hats\PDT\pdfpdt\usrpdf\ directory. A collection of sample source PDFs is provided in this directory. Unzip the sample PDFs into the usrpdf directory to modify or compile them. Open a command prompt to the <*RationalSDP_install_directory*>\hats\PDT\ directory and run the pdtcom.bat command. The Compile a Printer Definition Table window opens. In this window, select your source file from the drop-down box, enter a description, and click the **OK** button to compile it into a PDT file. The compiled PDT file is stored in the <*RationalSDP_install_directory*>\hats\PDT\pdfpdt\ directory. The file extension of the compiled PDT file is .hodpdt.

Copy the compiled PDT file into a folder, for example, MyPDT, in the root of the .ear project for your HATS Web application, or in the root of the HATS RCP Runtime Extension project for your HATS rich client project, or into a folder within the WEB-INF folder for your HATS portlet project.

The compiled PDT file is configured to HATS on the **Printing** tab of the connection editor for the default connection. In the Initialize section, select **Basic text files (Plain text format)** and click the **Initialize** button. In the Name/Value table, set the **PDTFile** property to have a value like **MyPDT/usribm3250.hodpdt**. Also set the **printMimeType** property to **application/octet-stream** and the **usePDT** property to **true**. For more information see "Printing" on page 135. Users will retrieve their 3270E print output as a binary print file suitable for copying to a particular type of non-Windows printer.

More information on Host On-Demand PDF and PDT files can be found in the Host On-Demand Knowledge Center at http://publib.boulder.ibm.com/infocenter/hodhelp/v11r0/index.jsp?topic=/com.ibm.hod.doc/doc/troubleshoot/pdtcompile.html.

**Notes:**

1. Printer Definition Files have no relation to Adobe Portable Document Format files.
2. If you have an existing PDT file from other HOD installations, you can use it for your custom printer without recompiling it. Simply copy the PDT file into place in the Web or rich client project as directed above, and set the default connection's print settings to use it.
3. For more information about Japanese JIS2004 support, see "JIS2004 support for PDT printing and Print-to-File for 3270E sessions" on page 433.

### For 5250 connections

To define print support for 5250 connections, in the editor for your 5250 default connection, click the **Printing** tab and select the **Enable print support** check box. Provide a URL for the System i Access for Web Printer Output window. The default URL is `http://hostname/webaccess/iWASpool`, where *hostname* is the name of the 5250 server. Print support for 5250 servers requires that System i Access for Web is installed on the 5250 host containing the print spool. All 5250 print functionality is provided by the System i Access for Web Printer Output servlet on the target 5250 host.

You do not need to perform any additional configuration. When a user of the HATS application issues a command to print files, System i Access for Web converts the host print jobs into PDF format and facilitates the download to the user. The user can click **View Print Jobs** on the application keypad to display the System i Access for Web Printer Output window. In this window, the user can select the following print options:

1. **PDF device type**
2. **Paper size**
3. **Destination**

Refer to your System i Access for Web documentation for more information about these print options.

## Providing documentation for users

To facilitate use of your HATS applications by users, it is recommend that you provide documentation on how to use the print support in HATS. Your documentation—provided either in the application's graphical user interface or in some other easily displayed form (perhaps a link in your template)—should describe:

- Using the functions in the Printer Output window, as described below
- If a PDF viewer (Adobe) is not installed, the user is prompted to save the file to disk.

  **Note:** You might consider adding a link to your application to where the user can download a free copy of the Acrobat Reader.
- When the HATS application closes, the printer output window closes automatically.
- Any application-specific information you want to include.

To use HATS print support, a user should follow these steps:

1. Start the HATS application.
2. Print the files.
3. Click **View Print Jobs**. For Web applications this is a button on the application keypad. For rich client applications it is an item on the pop-up menu in the Applications view. The Printer Output window displays a list of print jobs, if any exist.

   In the Printer Output window, the user can click options for a print job to either **Open** or **Delete** it. If the user clicks open, the print job is displayed as a PDF file in Acrobat Reader, if it is available. If not, the user is prompted to save the file to disk.

**Notes:**

a. While the print jobs are spooling, the user might see the file names for the print jobs in the printer output window, but **View** and **Delete** are disabled until the conversion to PDF format is complete.

b. Each user has a separate list of available print jobs. Jobs printed by one user are not visible to another user.

# Chapter 16. Enabling keyboard support

Users frequently interact with host applications using special keys on the physical keyboard, such as F1, Attn, Reset, and Clear. There are different ways in which the users of your HATS applications can send keystrokes to the host:

- By pressing keys on the physical keyboard. The term *keyboard support*, as used in this chapter, refers to this activity.
- By using buttons or links for function keys that are rendered by HATS default rendering. The user clicks on the button or link to send a host key to the host.
- By using the host keypad or buttons or links you individually insert. The user clicks on the button or link to send a host key to the host.

This chapter explains how to define keyboard support in your HATS project and contains tips for documenting keyboard support for your users.

To use keyboard support in HATS Web projects, you must have a supported Web browser. For a list of supported Web browsers and limitations, see "System Requirements for Host Access Transformation Services" at http://www.ibm.com/support/docview.wss?uid=swg27011794 and "Host Access Transformation Services 9.7 - Known issues and workarounds" at http://www.ibm.com/support/docview.wss?uid=ibm10876092.

## Defining keyboard support

If not enabled by default, you can enable keyboard support and select available options on the **Other** tab of your **Project Settings**. For more information, see “Keyboard support” on page 105.

You can also define a host keypad and an application keypad. See “Host keypad” on page 99 and “Application keypad” on page 99 for more information.

## Changing the appearance of the keypads

For HATS Web projects, use the **Host Keypad** and **Application Keypad** settings on the **Rendering** tab of your **Project Settings** to define which keys to display and whether to display them as buttons or links. See “Host keypad” on page 99 and “Application keypad” on page 99 for more information.

To change the style of a keypad for a specific HATS Web project, change the cascading style sheet (.css file) that corresponds to the keypad. This will be the style sheet you are using when you build your project. In the Rational SDP workbench, go to the **HATS Projects** view of the HATS Toolkit and expand the project name. Expand Web Content/Common/Stylesheets. For more information, see “Using style sheets” on page 317.

To modify the style sheet, double-click on the .css file to open the editor. Keep in mind that templates can override your style sheet changes. For more information about cascading style sheets, go to http://www.w3.org/Style/CSS/.

If not already defined, the style sheet can define any of the following HATS styles:

**table.HostKeypad**
Host keypad background

**table.ApplicationKeypad**
Application keypad background

**input.HostPFKey**
Host keypad PF buttons

**input.HostButton**
Host keypad buttons

**input.ApplicationButton**
Application keypad buttons

**a.HostKeyLink**
Host keypad links

**a.ApplicationKeyLink**
Application keypad links

For HATS rich client projects, use the **Host Keypad** settings on the **Rendering** tab of your **Project Settings** to define which keys to display and whether to display them as buttons or links. See “Host keypad” on page 99 for more information. Use the **Application Keypad** settings on the **Rendering** tab of your **Project Settings** to define which application-level buttons to display. Use the **Toolbar** settings on the **Rendering** tab to define whether the application-level buttons will display text, an image, or both. See “Application keypad” on page 99 and “Toolbar RCP-only ” on page 98 for more information.

## Providing documentation for users

To facilitate use of your HATS application, you may want to document the keyboard settings for your users—either in the application's graphical user interface or in some other easily displayed form (perhaps a link in your template). Your documentation should describe:

- How to turn keyboard support on and off, using **Turn Keyboard On/Off**.
- The HATS keyboard mapping, as shown in Table 5.
- How the Enter key works with HATS tabbed folders. Keyboard support must be off when the user presses the Enter key to display the contents of a different tab, after using the Tab key to move to a new tab. If keyboard support is not turned off, the Enter key goes to the host.
- For a list of the required level of Web browsers and limitations for using keyboard support with HATS Web applications, see "System Requirements for Host Access Transformation Services" at http://www.ibm.com/support/docview.wss?uid=swg27011794 and "Host Access Transformation Services 9.7 - Known issues and workarounds" at http://www.ibm.com/support/docview.wss?uid=ibm10876092.

*Table 5. Key mapping in HATS*

| Button on button bar | Default physical key mapping |
|---|---|
| F1 – F12 | F1 – F12 |
| F13 – F24 | Shift + F1 – F12 |
| ENTER | Enter |
| CLEAR | Esc |
| RESET | Ctrl + R |

*Table 5. Key mapping in HATS (continued)*

| Button on button bar | Default physical key mapping |
|---|---|
| SYSREQ | Shift + Esc |
| ATTN | Pause/Break |
| PAGEUP | Page Up |
| PAGEDN | Page Down |
| PA1 | Alt + Delete |
| PA2 | Alt + End |
| PA3 | Alt + Page Down |
| PRINT | Ctrl + P **Web-only**<br>Ctrl + Shift + P **RCP-only** |
| HELP | Ctrl + H |
| Enable/disable keyboard | Ctrl + K |
| Default | Alt + Insert |
| Refresh | Alt + Page Up |
| Reset HTML Form | Ctrl + S |
| Disconnect | Ctrl + D |
| View print jobs | Ctrl + J |
| Reverse | Alt + Enter |
| Field Exit | Shift + Enter **Web-only**<br>Numpad Enter **RCP-only**<br>**Note:** For more information, see "Special 5250 field key support." |
| Field+ | Ctrl + Numpad Plus **Web-only**<br>Numpad Plus **RCP-only** |
| Field- | Ctrl + Numpad Minus **Web-only**<br>Numpad Minus **RCP-only**<br>**Note:** For more information, see "Special 5250 field key support." |

**Note:** Ctrl+R is mapped to a host RESET for both bidirectional and non-bidirectional sessions, and Alt+Enter is mapped to reverse for bidirectional sessions.

## Special 5250 field key support

**Note:** For information about this support for mobile devices, see "Considerations and limitations for mobile devices" on page 44.

### Default field keypress combinations

The default field keypress combinations for HATS Web applications are listed below. For information about how to remap the keyboard, see "Remapping the keyboard in a HATS application" on page 363.

- Quick Field Exit: Ctrl + Enter.
- Field Exit: Shift + Enter.
- Quick Field Minus: Shift + Numpad minus.
- Field Minus: Ctrl + Numpad minus.

- Field Plus: Ctrl + Numpad plus.

## Quick Field Exit

The Quick Field Exit keypress can be used with HATS Web applications to speed the entry of numeric data for 5250 screens. This keypress clears the current field from the current caret position to end of the field, and advances the cursor position to the next field. Although this keypress can be used on any field, it is especially useful where users might otherwise use a Field Exit. A true Field Exit keypress sends the screen of data to the host and causes the HATS transformation to be redrawn. Therefore, screens requiring many Field Exit aids before being submitted may cause usability issues. The Quick Field Exit keypress, combined with the Field Exit Emulation feature, increases the speed of such applications considerably.

Because rich client applications have a different architecture than Web applications, a Quick Field Exit keypress in rich client applications is not necessary. There is no round-trip delay for a true Field Exit in rich client applications.

## Quick Field Minus

The Quick Field Minus keypress can be used with HATS Web applications to speed the entry of negative numeric data for 5250 screens. This keypress clears the current field from the current caret position to end of the field, prepends the field value with a minus sign character (-), and advances the cursor position to the next field. Although this keypress can be used on any field, it is only useful on 5250 signed numeric fields where users might otherwise use a Field Minus. A true Field Minus keypress sends the screen of data to the host and causes the transformation to be redrawn. Therefore, screens requiring many Field Minus aids before being submitted may cause usability issues. The Quick Field Minus keypress, combined with the Alternate way to input negative numbers feature, increases the speed of such applications considerably. Using the Quick Field Minus keypress on fields other than 5250 signed numeric fields may cause improper data entry.

Because rich client applications have a different architecture than Web applications, a Quick Field Minus keypress in rich client applications is not necessary. There is no round-trip delay for a true Field Minus in rich client applications.

## Alternate way to input negative numbers

To enter a negative number in a 5250 signed numeric field, users can enter a minus sign character (-) in the first position of the field, instead of using a Field Minus and incurring a round trip to the server. This allows for faster entry of large quantities of negative numbers into 5250 signed numeric fields. The HATS server emulates a Field Minus keypress for each 5250 signed numeric field entered with a leading minus sign character. Any unwanted prefilled data must be cleared from the field before tabbing to another field. Users should not enter a minus sign in the first position and also press Field Minus, Field Plus, or Field Exit.

## Field Exit Emulation

This feature automatically supplies a Field Exit aid key for the user, when required. A Field Exit aid key is required when certain types of 5250 fields are modified.

For example, if a user is using the Quick Field Exit keypress to advance from field to field, and submits the screen with an Enter key or some other non-field aid key, a Field Exit might be required for the field containing the cursor when the screen is submitted.

In this case, if a required Field Exit is not supplied, the host might respond with the error message, `Enter key not allowed in field`. With this HATS feature, the HATS server sends a Field Exit to the host before sending the Enter, avoiding the error message.

If you want to disable this feature, add the following statements to the application.hap file:

```
<class name="com.ibm.hats.common.RuntimeSettings">
        <setting name="emulateFieldExit" value="false"/>
</class>
```

If there is already a `<class>` section by this name, just add the `<setting>` line inside the section.

# Remapping the keyboard in a HATS application

## Concepts to understand before remapping HATS keys

Every HATS application embeds code from the Host On-Demand product to handle the connection to the Telnet server on behalf of the client. Therefore, this guide uses Host On-Demand terminology for mnemonic keywords which may be sent to the host computer to represent the user pressing keys in the host session.

### Mnemonic keywords

**HATS-supported 3270 mnemonic keyword table**

*Table 6. Mnemonic Keywords for SendKeys*

| Function | Mnemonic Keyword |
|---|---|
| Attention | [attn] |
| Alternate View | [altview] |
| Clear | [clear] |
| Enter | [enter] |
| F1 | [pf1] |
| F2 | [pf2] |
| F3 | [pf3] |
| F4 | [pf4] |
| F5 | [pf5] |
| F6 | [pf6] |
| F7 | [pf7] |
| F8 | [pf8] |
| F9 | [pf9] |
| F10 | [pf10] |
| F11 | [pf11] |
| F12 | [pf12] |
| F13 | [pf13] |
| F14 | [pf14] |
| F15 | [pf15] |
| F16 | [pf16] |

*Table 6. Mnemonic Keywords for SendKeys (continued)*

| Function | Mnemonic Keyword |
|---|---|
| F17 | [pf17] |
| F18 | [pf18] |
| F19 | [pf19] |
| F20 | [pf20] |
| F21 | [pf21] |
| F22 | [pf22] |
| F23 | [pf23] |
| F24 | [pf24] |
| Home | [home] |
| PA1 | [pa1] |
| PA2 | [pa2] |
| PA3 | [pa3] |
| Reset | [reset] |
| System Request | [sysreq] |

### HATS-supported 5250 mnemonic keyword table

*Table 7. Mnemonic Keywords for SendKeys*

| Function | Mnemonic Keyword |
|---|---|
| Attention | [attn] |
| Alternate View | [altview] |
| Clear | [clear] |
| Enter | [enter] |
| F1 | [pf1] |
| F2 | [pf2] |
| F3 | [pf3] |
| F4 | [pf4] |
| F5 | [pf5] |
| F6 | [pf6] |
| F7 | [pf7] |
| F8 | [pf8] |
| F9 | [pf9] |
| F10 | [pf10] |
| F11 | [pf11] |
| F12 | [pf12] |
| F13 | [pf13] |
| F14 | [pf14] |
| F15 | [pf15] |
| F16 | [pf16] |
| F17 | [pf17] |
| F18 | [pf18] |

*Table 7. Mnemonic Keywords for SendKeys (continued)*

| Function | Mnemonic Keyword |
|---|---|
| F19 | [pf19] |
| F20 | [pf20] |
| F21 | [pf21] |
| F22 | [pf22] |
| F23 | [pf23] |
| F24 | [pf24] |
| Field Exit | [fldext] |
| Field+ | [field+] |
| Field- | [field-] |
| Help | [help] |
| PA1 | [pa1] |
| PA2 | [pa2] |
| PA3 | [pa3] |
| System Request | [sysreq] |
| Page Up | [pageup] |
| Page Down | [pagedn] |
| Quick Field Exit | [qfldext] |
| Quick Field Minus | [qfld-] |
| Reset | [reset] |
| System Request | [sysreq] |
| Test Request | [test] |

### Determining keycode values

To determine the keycode value for a given key on the keyboard, save the following HTML code in a file and open it in an Internet Explorer browser window. The keycode value is displayed for any key pressed in that browser.

```
<html>
<head>
<script>
document.onkeydown = keydownhandler;
function keydownhandler()
{
 document.testform.displayField.value = "Keycode: " + window.event.keyCode;
}
</script>
</head>
<body>
 <center>
  <form name="testform">
   <input type="text" name="displayField" value="Hit any key">
  </form>
 </center>
</body>
</html>
```

## Remapping keys for HATS Web applications

In every HATS Web application is a file named KBS.js which determines how the keycode values generated by the users keyboard are mapped to key signals sent to

the host session. To remap the HATS keyboard, you must edit the KBS.js file. To locate KBS.js, go to the **Navigator** view then look in the *project_name*/Web Content/Common directory. There are three values in KBS.js which determine the relationship between any key on the keyboard being pressed and what signal is then sent to the host. These three values are:

1. The keycode value for the pressed key, represented as an integer variable in the KBS.js file. For example,

   ```
   var CODE_F2 = 113;
   ```

   for the F2 key,
2. The combination of the keycode value and the Alt, Ctrl, and Shift states of the keyboard, and
3. The Host On-Demand mnemonic keyword sent for the keycode combination value and Alt, Ctrl, and Shift states. For example,

   ```
   [CODE_F2, 0, 0, 0, '[pf2]']
   ```

   sends the pf2 key signal to the host when the F2 key is pressed on the user's keyboard.

To change key mappings in your HATS Web application, you can change or add combinations allowing new or existing keys to send the mnemonic keyword.

| Be aware that you cannot remap the Ctrl, Alt, and Shift modifier keys. HATS uses JavaScript, and JavaScript does not consider these keys to be keypresses, but rather modifiers. |
|---|

To remap keys in HATS Web applications, you only need to modify the keycode and defaultKeyMappings variables at the top of the KBS.js file.

**Note:** There are dependencies between the keys listed in the KBS.js file, the Keyboard Support state of the Project Settings, and the presence or absence of any key buttons listed on the screen. The rules governing these dependencies are listed below.

1. Keyboard support can be set using project-level settings to specify whether it should follow HATS V5, or earlier, rules or if it should support all mapped keys, regardless of what buttons or links are displayed. To modify these settings, go to the **HATS Projects** view and double click the **Project Settings** of your HATS project, select the **Other** tab and click **Keyboard Support**. For more information, see "Keyboard support" on page 105.
2. By default, HATS shows the application keypad but does not display the host keypad. In order for HATS to display the host keypad, you need to go to the **HATS Projects** view and double click the **Project Settings** of your HATS project, select the **Rendering** tab and click **Host Keypad**. You will then have the option of enabling the check box to display the host keypad as well as selecting which keys to display.

## Example

The following example shows how to map the Ctrl+Home key combination on the keyboard. To send the Home key to the host session and also place a button for this function in your transformation, two steps are required: Edit KBS.js and then modify your transformation as described in "Adding additional keypad buttons to a transformation" on page 370.

To specify the key combination that sends the host key, modify the KBS.js file by adding the lines specified in ***bold italic*** below:

```
var CODE_BACKSPACE = 8;
 var CODE_TAB       = 9;
 var CODE_ENTER     = 13;
 var CODE_PAUSE     = 19;
 var CODE_ESC       = 27;
 var CODE_PAGEUP    = 33;
 var CODE_PAGEDOWN  = 34;
 var CODE_END       = 35;
 var CODE_HOME      = 36;
 var CODE_INSERT    = 45;
 var CODE_DELETE    = 46;
 var CODE_A         = 65;
 var CODE_B         = 66;
 var CODE_C         = 67;
 var CODE_D         = 68;
 var CODE_E         = 69;
 var CODE_F         = 70;
 var CODE_G         = 71;
 var CODE_H         = 72;
 var CODE_I         = 73;
 var CODE_J         = 74;
 var CODE_K         = 75;
 var CODE_L         = 76;
 var CODE_M         = 77;
 var CODE_N         = 78;
 var CODE_O         = 79;
 var CODE_P         = 80;
 var CODE_Q         = 81;
 var CODE_R         = 82;
 var CODE_S         = 83;
 var CODE_T         = 84;
 var CODE_U         = 85;
 var CODE_V         = 86;
 var CODE_W         = 87;
 var CODE_X         = 88;
 var CODE_Y         = 89;
 var CODE_Z         = 90;
 var CODE_HOSTHOME = 36;
 var CODE_F1        = 112;
 var CODE_F2        = 113;
 var CODE_F3        = 114;
 var CODE_F4        = 115;
 var CODE_F5        = 116;
 var CODE_F6        = 117;
 var CODE_F7        = 118;
 var CODE_F8        = 119;
 var CODE_F9        = 120;
 var CODE_F10       = 121;
 var CODE_F11       = 122;
 var CODE_F12       = 123;
 var HostKey        =1;
 var ApplicationKey =2;
/************************************************************
 * NOTICE: DO NOT MODIFY THE ABOVE VARIABLES!!
 ************************************************************/

 var defaultKeyMappings = [
 // KEYCODE,       ALT, CTRL, SHIFT, MNEMONIC
 //============ command key mappings ================
   [CODE_ENTER,      0,    0,     0, '[enter]'     ],
   [CODE_PAUSE,      0,    0,     0, '[attn]'      ],
   [CODE_ESC,        0,    0,     0, '[clear]'     ],
   [CODE_ESC,        0,    0,     1, '[sysreq]'    ],
   [CODE_PAGEUP,     0,    0,     0, '[pageup]'    ],
```

```
  [CODE_PAGEUP,     1,    0,     0, 'refresh'     ],
  [CODE_PAGEDOWN,   0,    0,     0, '[pagedn]'    ],
  [CODE_HOSTHOME, 0,    1,     0, '[home]'    ],
  [CODE_PAGEDOWN,   1,    0,     0, '[pa3]'       ],
  [CODE_END,        1,    0,     0, '[pa2]'       ],
  [CODE_INSERT,     1,    0,     0, 'default'     ],
  [CODE_DELETE,     1,    0,     0, '[pa1]'       ],
  [CODE_D,          0,    1,     0, 'disconnect'  ],
  [CODE_H,          0,    1,     0, '[help]'      ],
  [CODE_P,          0,    1,     0, '[printhost]' ],
  [CODE_J,          0,    1,     0, 'printjobs'   ],
  [CODE_ENTER,      1,    0,     0, 'reverse'     ],
  [CODE_K,          0,    1,     0, 'toggle'      ],
  [CODE_S,          0,    1,     0, 'ResetButton' ],
//============ function key mappings ==============
  [CODE_F1,         0,    0,     0, '[pf1]'       ],
  [CODE_F1,         0,    0,     1, '[pf13]'      ],
  [CODE_F2,         0,    0,     0, '[pf2]'       ],
  [CODE_F2,         0,    0,     1, '[pf14]'      ],
  [CODE_F3,         0,    0,     0, '[pf3]'       ],
  [CODE_F3,         0,    0,     1, '[pf15]'      ],
  [CODE_F4,         0,    0,     0, '[pf4]'       ],
  [CODE_F4,         0,    0,     1, '[pf16]'      ],
  [CODE_F5,         0,    0,     0, '[pf5]'       ],
  [CODE_F5,         0,    0,     1, '[pf17]'      ],
  [CODE_F6,         0,    0,     0, '[pf6]'       ],
  [CODE_F6,         0,    0,     1, '[pf18]'      ],
  [CODE_F7,         0,    0,     0, '[pf7]'       ],
  [CODE_F7,         0,    0,     1, '[pf19]'      ],
  [CODE_F8,         0,    0,     0, '[pf8]'       ],
  [CODE_F8,         0,    0,     1, '[pf20]'      ],
  [CODE_F9,         0,    0,     0, '[pf9]'       ],
  [CODE_F9,         0,    0,     1, '[pf21]'      ],
  [CODE_F10,        0,    0,     0, '[pf10]'      ],
  [CODE_F10,        0,    0,     1, '[pf22]'      ],
  [CODE_F11,        0,    0,     0, '[pf11]'      ],
  [CODE_F11,        0,    0,     1, '[pf23]'      ],
  [CODE_F12,        0,    0,     0, '[pf12]'      ],
  [CODE_F12,        0,    0,     1, '[pf24]'      ]
];
```

**Note:** The name given to any new keycode variable (CODE_HOSTHOME in this example) is not important except that it be unique and match the corresponding entry in the defaultKeyMappings variable list. Also note in this example that adding a new keycode variable for the Home key is not actually necessary since the

```
var CODE_HOME = 36;
```

entry already exists for the Home key's keycode. However, if you are mapping a keyboard key that is not associated with a keycode variable already defined in KBS.js, such an entry would be necessary.

## Remapping keys for HATS rich client applications

In the rich client environment, default keyboard mappings are defined in the plugin.xml file of the HATS RCP Runtime Extension plug-in and apply to all HATS rich client applications running in the same environment.

An Eclipse keyboard context is used to influence what commands are available to the user at any given moment. When a user is using a HATS transformation view, the HATS keyboard context is used. This context is registered in the plugin.xml file of the HATS RCP Runtime Extension plug-in. The following example shows how this appears:

```
<extension
      point="org.eclipse.ui.contexts">
   <context
         name="%KEYBOARD_CONTEXT_NAME"
         description="%KEYBOARD_CONTEXT_NAME"
         id="com.ibm.hats.rcp.transformationContext"
         parentId="org.eclipse.ui.contexts.window">
   </context>
</extension>
```

The default keyboard mappings are also defined in the plugin.xml file. The following example shows how a key mapping for a HATS application running in Eclipse RCP or Lotus Expeditor appears in this file:

```
<extension
      point="org.eclipse.ui.commands">
   <category
         name="%COMMAND_CATEGORY_NAME"
         description"%COMMAND_CATEGORY_NAME"
         id="com.ibm.hats.rcp.transformationCategory"
   </category>
   <command
         name="[pf1]"
         category="com.ibm.hats.rcp.transformationCategory"
         id="com.ibm.hats.rcp.send_[pf1]">
   </command>
     ...
 </extension>

<extension
      point="org.eclipse.ui.bindings">
   <key
         commandId="com.ibm.hats.rcp.send_[pf1]"
         contextId="com.ibm.hats.rcp.transformationContext"
         configuration="org.eclipse.ui.defaultAcceleratorConfiguration"
         sequence="F1"
   </key>
     ...
 </extension>
```

The key bindings for running in Lotus Notes are separate from the other key bindings. The equivalent key binding for the previous example would be:

```
   <key
         commandId="com.ibm.hats.rcp.send_[pf1]"
         contextId="com.ibm.hats.rcp.transformationContext"
         schemeId="com.ibm.workplace.notes.hannoverConfiguration"
         sequence="F1"
   </key>
```

## Example

The following example shows how to map the Ctrl+Home key combination on the keyboard for a rich client application running in Eclipse RCP or Lotus Expeditor. First, add the command tag for [home], and key tag for Ctrl+Home, to the plugin.xml file in the HATS RCP Runtime Extension plug-in as shown below:

```
<extension
      point="org.eclipse.ui.commands">
     ...
   <command
         name="[home]"
         category="com.ibm.hats.rcp.transformationCategory"
         id="com.ibm.hats.rcp.send_[home]">
   </command>
     ...
 </extension>
```

```
<extension
      point="org.eclipse.ui.bindings">
     ...
   <key
         commandId="com.ibm.hats.rcp.send_[home]"
         contextId="com.ibm.hats.rcp.transformationContext"
         configuration="org.eclipse.ui.defaultAcceleratorConfiguration"
         sequence="Ctrl+Home"
   </key>
     ...
 </extension>
```

Next, update the transformation view code so that the view is aware that a new key mapping is registered in the plugin.xml file. To do this, edit the MainView.java file in your project. This file is located in the \Source*<project name>*\views folder under your project in the HATS Projects view. Double-click on the MainView.java file to open it in the editor. Add the createKeyboardActions method following the MainView method as shown below:

```
protected void createKeyboardActions() {
    super.createKeyboardActions();
    addKeyboardAction("[home]", "com.ibm.hats.rcp.send_[home]");
}
```

As with HATS Web applications, be aware that you cannot remap the Ctrl, Alt, and Shift modifier keys for HATS rich client applications.

You can provide a way for your users to configure the default Eclipse keyboard preferences. This is normally configured from the menu bar using **Window > Preferences > Workbench > Keys**. If you provide this for your users, they will be able to override the key mappings defined in the HATS RCP Runtime Extension plug-in.

## Adding additional keypad buttons to a transformation

You can add a new button or link into the default Host Keypad. The control for this is in **Project Settings** on the **Rendering** tab, **Host Keypad** settings. The new button or link will appear whenever the default Host Keypad is used. You can add a new button or link to a single transformation using the **Insert Host Keypad > Individual Key** function when editing the transformation. See “Individual Key” on page 183 for more information. The GUI for adding your own button or link requires a Caption and a Mnemonic. In this example, the Caption is **Home**, and the Mnemonic is **[home]**. If you want keyboard support for this new button, you must add it to the KBS.js defaultKeyMappings table (for Web applications) or the plugin.xml file of the HATS RCP runtime extension plug-in (for rich client applications).

# Chapter 17. Using host simulation

The HATS Toolkit includes a host simulation capability. With this capability, you can record host simulation trace files that can be saved and then used instead of a live host connection. Following are ways you can use host simulation files instead of using a live host connection:

- Create screen captures, screen events, and screen transformations using the host terminal function.
- Create and test macros using the host terminal function.
- Test HATS applications using the Rational SDP local test environment.
- Deploy HATS applications to a runtime environment to use as demonstrations.

In addition, the host simulation capability can be used in troubleshooting by allowing you to record a host simulation file in the runtime environment that you can use, along with other traces and logs, to document a failing scenario.

**Notes:**

1. Host simulation is not supported for SSL enabled connections.
2. Host simulation traces of ENPTUI-enabled devices recorded with the stand alone Host Simulator tool might not play back correctly if ENPTUI is not enabled in the HATS project.

## Host simulation wizard

To start the host simulation wizard take any of the following steps:

- From the menu bar select **File > New > HATS Host Simulation**.
- From the menu bar select **HATS > New > Host Simulation**.
- From the toolbar click the **Create a HATS Host Simulation file** item.
- Right-click a HATS project and select **New HATS > Host Simulation**.
- Right-click a HATS EJB project and select **New Host Simulation**.

The first page of the wizard is the **Create a Host Simulation Trace** page, which contains the following fields:

**Project**
Use this field to specify the target project for the host simulation trace file.

**Name** Use this field to supply a name for the host simulation trace file. The default name is `ProjectNameTrace_num`. Where`ProjectName` is the name of the target project, and `num` is a number appended if the base file name already exists.

**Location**
This field displays the destination location for the host simulation trace file. It cannot be modified.

**Description**
Use this field to add any optional description you want.

The second page of the wizard is the **Choose Connection** page. Use this page to select which connection within the project to use for recording the trace file. The project's default connection is the default for recording.

After clicking **Finish** in the wizard, the host simulation recorder finds an unused port from the HATS **Host Simulation** preferences **TCP/IP port range** settings to use and sets itself up as a proxy between the real host and the HATS host terminal. See "Using HATS preferences" on page 126 for more information. Next the host terminal is displayed.

Using the host terminal, interact with the host system to record the scenario for this trace file. You can use the **Stop Host Simulation Recording** toolbar icon to stop recording and the **Restart Host Simulation Recording** toolbar icon to restart recording.

When you stop recording you are given the options to save the trace file and whether to close the host terminal window. Saved trace files appear in the HATS Projects view under the target project in a folder named **Host Simulations**.

# Host simulation editor

To open the host simulation editor for a host simulation trace file, double-click the file in your project's **Host Simulations** folder. This editor allows you to browse trace information and modify playback settings. There are two tabs in the host simulation editor, the **Overview** tab and the **Source** tab.

## Overview tab

There are two sections on the **Overview** tab, the **General Information** section and the **Playback Settings** section.

### General Information section

The General Information section displays the trace file name and description, the date and time it was last modified, and the name of the connection it is associated with.

### Playback Settings section

In this section you can modify playback settings and click a button that will playback the trace on the host terminal. The following playback settings can be modified:

**TCP/IP Port Range**
: During trace playback, HATS host simulation function listens on a range of ports to simulate host communication with the host terminal or your HATS application. Multiple ports are used to allow trace playback on multiple sessions concurrently. Default values are prefilled using the HATS host simulation preferences settings. See "Using HATS preferences" on page 126 for more information.

**Time Delay Settings**
: Use these values, **Delay**, **Minimum (ms)**, and **Maximum (ms)** to set the time delay (in milliseconds) for host simulation to wait during playback before responding to requests from the host terminal or your HATS application. Default values are prefilled using the HATS host simulation preferences settings and can be modified here. See "Using HATS preferences" on page 126 for more information.

**Playback on Host Terminal button**
: Click this button to open a host terminal window and playback the host simulation trace file.

### Source tab

The **Source** tab displays the source of the host simulation trace file in an XML editor.

## Recording in the runtime environment

In the Web environment, the administrator can use the HATS administrative console to enable host simulation recording during runtime. See "Set Trace Options" on page 383 for how this is done.

In the rich client environment, the user can use the workbench preferences to enable host simulation recording during runtime. See "Troubleshooting preferences" on page 82 for how this is done.

## Playback options

The following table summarizes the various host simulation playback options.

*Table 8. Host simulation playback options*

| Environment | Launch action | Capabilities |
|---|---|---|
| Host terminal | In the HATS Projects view, right-click the host simulation file and select **Playback on Host Terminal**. | View playback |
| Host terminal | In the HATS Projects view, double-click the host simulation file. In the host simulation file editor, click the **Playback on Host Terminal** button. | View playback |
| Host terminal | In the HATS Projects view, double-click a connection. In the connection editor on the **Basic** tab, select **Use Host Simulation instead of the live connection** and select the trace from the drop-down.<br><br>In the HATS Projects view right-click the connection and select **Open Host Terminal**. | Create screen captures, screen events, and screen transformations.<br><br>Create and test macros. |
| Local test runtime<br><br>(Web, EJB, and rich client applications) | In the HATS Projects view, double-click a connection. In the connection editor on the **Basic** tab, select **Use Host Simulation instead of the live connection** and select the trace from the drop-down.<br><br>Run the application in the local test environment. | Test the HATS application in the local test environment. |
| Deployed runtime<br><br>(Web, EJB, and rich client applications) | In the HATS Projects view, double-click a connection. In the connection editor on the **Basic** tab, select **Use Host Simulation instead of the live connection** and select the trace from the drop-down.<br><br>Export and deploy the application. | Deploy the HATS application as a demonstration. |

## Importing and exporting trace files

A non-supported stand alone tool named, Host Simulator, has been available for download to allow users to perform some of the functions that are now available with the host simulation function built into HATS V9.7. You can import both the trace files created by the stand alone Host Simulator tool and the trace files created by the HATS host simulation function. To import the trace files created by the stand alone Host Simulator tool, from the menu bar select **File > Import > HATS > Host Simulator trace into HATS**. To import the trace files created by the HATS host simulation function, from the menu bar select **File > Import > HATS > HATS Host Simulation**. In the wizard, click the **Add** button to browse the file system and select the trace file to import. Next, select the Host Simulations folder within the destination project. If importing a trace file created by the stand alone Host Simulator tool, select the connection within the project with which to associate the trace file. Click **Finish**.

To export a HATS host simulation trace file, from the menu bar select **File > Export > HATS > HATS Host Simulation**. Next, select the trace file to export, browse to select the destination, and click **Finish**.

# Chapter 18. Using HATS administrative console

The HATS administrative interface can be used to manage connections and perform problem determination for HATS Web applications. This chapter describes how you can use these management functions. For information related to HATS rich client applications, see “Administering HATS rich client applications” on page 76.

You can include the HATS administrative console support files with the set of HATS applications contained within an Enterprise Archive (.ear) file deployed to WebSphere Application Server. The deployed .ear file provides a runtime environment, in which the HATS administrative console enables you to manage HATS applications. If you include the HATS administrative console files in multiple projects, more than one administrative console can be started using different application names; however, each administrative console provides the same views and functions.

**Note:** This option is not supported by HATS portlet projects.

Instead of including the HATS administrative console support files in each .ear file, you can create an .ear file that contains only HATS administrative console files. When you deploy this .ear file, you can start the HATS administrative console and change the management scope to remotely manage HATS applications in multiple .ear files with one instance of the administrative console. The deployed HATS administrative console application gives you a single point of administration for all HATS applications. When you create a new HATS project, you specify whether to include HATS administrative console files in the project. If you create a HATS administrative console project, you no longer need to add the administration support files to other projects you create.

To create a HATS administrative console project in HATS Toolkit, follow the path **File > New > Project > HATS > HATS Administration Project**. You should specify an Enterprise Application project name that uniquely identifies this project as the HATS administrative console project. The HATS administrative console project only appears in HATS Toolkit on the **Navigator** tab. Deploy the .ear file you created to WebSphere Application Server and start it (see “Starting HATS administrative console” on page 378). To change the scope of management for the HATS administrative console project, refer to “Selecting management scope” on page 379.

Other administration topics are described in “Display terminal functions” on page 384.

## HATS administrative console and WebSphere security

The HATS administrative console uses a Java Management Extensions (JMX) bean to perform remote administration. When WebSphere Application Server global security is enabled, these JMX calls are authenticated by WebSphere security and must have a valid level of authorization in WebSphere. Consequently, remote administration from the HATS administrative console does not work properly unless the user ID is defined as a WebSphere Application Server administrative console user with either Administrator or Operator authority.

The HATS administrative console allows you to view and change problem determination settings. It also allows you to view connection status or disconnect host connections. When you deploy a HATS application with HATS administrative capabilities, you can map each of three HATS roles to particular system user IDs for security. The three defined HATS roles (HATSAdministrator, HATSOperator, and HATSMonitor) each have different capabilities within the HATS administrative console. For more information see "HATS administrative console roles" on page 377.

Similarly, the WebSphere Application Server administrative console allows you to view and change the configuration of the WebSphere Application Server environment. It enables you to install, start, stop, and uninstall Web applications such as HATS applications. There are WebSphere console user roles (Administrator, Configurator, Operator, or Monitor) that provide different capabilities within the WebSphere Application Server administrative console. The following links from the WebSphere Application Server Knowledge Center provide additional detail on these security roles and policies:

- Administrative roles and naming service authorization http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp?topic=/com.ibm.websphere.base.doc/info/aes/ae/csec_adminconsole.html
- Default MBean security policy http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp?topic=/com.ibm.websphere.base.doc/info/aes/ae/cjmx_admin_defmbsec.html

If WebSphere global security is enabled, the users IDs that have a HATS role can connect to the HATS administrative console in a HATS application, and can perform administration tasks within that specific HATS application only. If the user tries to change the management scope to remotely administer another HATS application, WebSphere security will check the authority of the user ID. If the user ID is not a valid WebSphere Application Server administrative console user with Administrator or Operator authority, the JMX calls will be blocked and HATS remote administration will not function properly.

For example, if the user ID is either not defined as a WebSphere Application Server administrative console user at all, or defined with only Monitor authority, the Select Application list under Management Scope in the HATS administrative console will not display any other HATS applications. It will not be possible to change the management scope because no other applications are listed.

As another example, if the user ID is defined as a WebSphere Application Server administrative console user with Configurator authority, the list will show other HATS applications, and the user can change scope from one application to another. However, the information shown in the HATS administrative console may be incorrect, and any changes attempted through remote administration will not take effect.

Therefore, if WebSphere global security is enabled two options exist for using the HATS administrative console.

**Option 1: Keep HATS and WebSphere Application Server user roles separate**
If you do not want to give any WebSphere Application Server administration authority to any of the user IDs that are mapped to HATS roles, then you must include the HATS administrative console in every HATS application, and you will need to administer each HATS application separately using the HATS administrative console for that specific application. You will not be able to use remote administration.

**Option 2: Combine HATS and WebSphere Application Server user roles**
If you want to use the HATS administrative console in a single HATS application to remotely administer other HATS applications by changing the management scope, then the user IDs that are mapped to HATS roles must also be defined in WebSphere Application Server as WebSphere administrative console users with either Administrator role or Operator role. These users will be able to change the management scope in the HATS administrative console, and they will be able to perform the other HATS administrative tasks that are allowed for the specific HATS role to which they are mapped. In addition, the users will also be able to log into the WebSphere Application Server administrative console and perform WebSphere administrative tasks that are allowed for the WebSphere Application Server administrative console user role to which they are mapped.

For either option, if WebSphere global security is enabled, ensure the **Enable application security** option is selected. For WebSphere Application Server V6.x, from the WebSphere administrative console, select **Security > Secure administration, applications, and infrastructure > Application security > Enable application security**. For WebSphere Application Server V7.x and v8.x, from the WebSphere administrative console, select **Security > Global security > Application security > Enable application security**.

In summary, if WebSphere global security is enabled, the user IDs that are mapped to any HATS roles must also be mapped to WebSphere Application Server Administrator or Operator roles in order for HATS remote administration to function properly.

## HATS administrative console roles

HATS administrative console is bound to WebSphere security. This means that when WebSphere Application Server security is enabled, your system administrators must have proper authentication to perform HATS administrative tasks. The security functions used by HATS are based on the Java EE form-based authentication process. When WebSphere Application Server security is enabled, HATS administrative console operations are restricted based on the role defined for a user ID. There are three roles defined for use of the HATS administrative console. Each role has different capabilities.

**HATSAdministrator**
This role has full administration capability, able to do all of the following tasks:

- Manage license information
- Select the scope of management
- View active connections, connection pools, connection pool definitions, user lists, and logs
- Terminate active connections
- View log and trace files
- Set log and trace options
- View potentially sensitive data by seeing the display terminal, change the trace settings, terminate existing connections, and download trace files.

By default, the HATSAdministrator role is mapped to “All Authenticated” users.

**HATSOperator**
This role is equivalent to the HATSAdministrator role, but cannot access potentially sensitive data. A user with this role cannot enable display terminal tracing, nor download trace files.

**HATSMonitor**
This role only allows the user to view non-sensitive data, such as active connections, connection pools, user lists, and logs. A user with this role cannot change any settings, such as license management settings, log and trace file options, display terminal settings, nor download trace files.

The mapping of roles to users or groups of users is performed using the WebSphere administrative console.

For WebSphere Application Server V6.x, ensure the **All Authenticated** option on the **Map security roles to users/groups** panel is checked for the appropriate HATS role. For WebSphere Application Server V7.x and V8.x, ensure the **All Authenticated in Application Realm** option is selected using the **Map Special Subjects** drop-down on the **Map security roles to users/groups** panel for the appropriate HATS role.

## Starting HATS administrative console

The HATS administrative console is Web-based. You perform administration tasks using a Web browser, through a user interface provided by the HATS administrative console servlet, **HATSAdminServlet**.

To start the HATS administrative console, load this URL in your browser: **http://***host:port/appname***/hatsadmin/admin**, where ***host*** is your host name, ***port*** is the port number of HTTP transport of the application server on which the HATS application is running and ***appname*** is the name of a HATS application that includes administration support.

**Note:** The ***port*** specification is not needed if URL requests are directed to a Web server configured for WebSphere.

When you start the HATS administrative console, it uses the default language of the server. You can, however, change the language by selecting **Preferences** in the navigation frame, choosing the language you want, and restarting the browser. The new language choice remains in effect for that browser until you select another language in **Preferences**.

**Notes:**

1. If cookies are disabled for the browser accessing the HATS administrative console, the following changes occur:
   - The HATS administrative console does not keep track of the user's language preference.
   - The language of the browser is determined by the browser's preferred language, based on the Accept-Language header.
   - The language selection choice does not appear in the HATS administrative console preferences.
2. For a list of supported Web browsers and limitations, see "System Requirements for Host Access Transformation Services" at http://www.ibm.com/support/docview.wss?uid=swg27011794 and "Host Access Transformation Services 9.7 - Known issues and workarounds" at http://www.ibm.com/support/docview.wss?uid=ibm10876092.

## Starting the administrative console while in HATS Toolkit

The HATS administrative console can be started in HATS Toolkit, using either Debug on Server or Run on Server mode.

To launch the HATS administrative console for a particular project, right-click the project in the **HATS Projects** view, and click **Open Administrative Console**.

If the project is not already running in a local test environment, then it is launched using Debug on Server mode and a Web browser page is opened to the administrative console URL.

If the project is already running using Run on Server mode, then a prompt is displayed that informs you that the server is not running in Debug mode. If you select the option, **Continue in the current mode**, the administrative console will open in Run on Server mode.

For more information about running in different test modes see Testing modes for Web projects in *HATS Getting Started*.

The runtime settings you change when running the administrative console in the HATS Toolkit depends on which mode you are using. For more information, see "Administering problem determination components" on page 381.

# Using the functions in HATS administrative console

This section guides you through the various functions in the HATS administrative console.

## Selecting management scope

You can select the scope of management. HATS administrative console maintains a list of the management scopes for the life of the HttpSession. The initial scope of management is selected by default as the HATS enterprise application (.ear) of the HATS application used to access the HATS administrative console. You can change the scope of management to manage other HATS enterprise applications deployed within the same WebSphere cell.

The key to changing management scope is providing the host and port information for HATS administrative console to locate installed HATS enterprise applications. The port value specified must be an active WebSphere BOOTSTRAP_ADDRESS port. WebSphere application servers, nodes, and cells each have a BOOTSTRAP_ADDRESS port. Specifying the port value for one of these WebSphere resources enables administration for the HATS enterprise applications deployed to that resource. Specifying a host and port of a WebSphere node enables administration of all HATS enterprise applications installed on that node. Refer to WebSphere Application Server documentation for more information about the BOOTSTRAP_ADDRESS port.

In a secured environment, an administrator may see the following message in the HATS administrative console while trying to change the scope of administration:

```
There was a problem in trying to manage the scope you selected. There are
no HATS applications found. Your management scope has not been changed.
```

This may occur because in a WebSphere secured environment, the user is not authorized to carry out the communication required to populate the list of available applications to be administered.

If WebSphere is using the local operating system as the user registry, the username used to access the HATS administrative console must have Administrator rights in the local operating system registry. If WebSphere is using LDAP as the user registry, the username used to access the HATS administrative console must be the server ID used by WebSphere for accessing LDAP.

## Managing licenses

There are two panels under the License Management heading in the left navigation frame for managing license information:

- License Usage panel: tracks license usage for all application servers in a node. HATS tracks the number of HATS connections to host or database resources and logs a message when the value exceeds the number of licenses purchased.
- License Settings panel: specifies the type of license you have, **Authorized User** or **Value Unit**. If you have an Authorized User license, specify how many licenses you have. Valid values are from 1-50000. Also specify whether to collect license tracking information, and the template name of the file in which to write tracking information. The default template file name is license.txt. For more information about specifying license settings, see Enabling HATS runtime and license settings in *HATS Getting Started*.

## Monitoring connections

There are separate panels under the Connection Management heading in the left navigation frame with which you can monitor the different types of connections:

- Host Connections panel: connections to 3270, 5250, or VT applications running on a host, such as an IBM i server or other server accessible via Telnet.
- Database Connections panel: connections to database though the Java Database Connection (JDBC) interface from HATS applications.

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

## Monitoring connection pools

The Connection Pools panel under the Connection Management heading in the left navigation frame displays information about defined connection pools. A connection pool is not listed until at least one connection is allocated from the pool.

Connection pools are collections of communication links to back-end data sources, such as 3270 applications or databases. When an Integration Object or an application is run on behalf of a client request, it obtains an available connection from a pool, uses it for access to the data source, then returns the connection to the pool. When connection pooling is enabled, the processor usage of establishing a connection is absorbed in its first use. Each reuse of this connection benefits from the prior establishment of the connection and can run faster.

## Monitoring pool definitions

The Connection Pool Definitions panel under the Connection Management heading in the left navigation frame displays pool definition information for host and database connections. A pool definition is not listed until at least one connection is allocated from the defined pool.

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

## Monitoring user lists and user list members

The User Lists panel under the User Management heading in the left navigation frame displays information about user lists.

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

## Administering problem determination components

The HATS administrative console enables you to monitor logging and tracing to help you resolve problems. You can view the messages in the log, and change log and trace settings by selecting the links under the Troubleshooting heading in the left navigation frame. Properties files maintain settings for these problem determination aids.

When you change settings while accessing the HATS administrative console running in a runtime environment, the changes are saved in the runtime.properties file. Although not recommended, you can also edit the file directly to change settings. The file is located in the *was_dir*/installedApps/*ear_name* directory for a HATS enterprise application.

When you change settings while accessing the HATS administrative console running in a local test environment in the HATS Toolkit, the changes are saved in one of two different files. Changes are saved in the runtime.properties file, when running in Run on Server mode, and in the runtime-debug.properties file, when running in Debug on Server mode. For more information, see “Starting the administrative console while in HATS Toolkit” on page 379. You can also edit the files directly to change settings. The files are located in the root of the .ear project in the HATS Toolkit. Be aware that any changes made to the runtime.properties file in the HATS Toolkit are retained and become effective when you deploy the HATS application to a runtime environment. The runtime-debug.properties file is ignored in the runtime environment.

**Note:** When running on WebSphere Application Server for z/OS®, the runtime.properties file must be edited with an ASCII editor. This can be done by transferring the file to a workstation, for example using FTP, editing the file and transferring it back. Also, there is a copy of runtime.properties for every active Servant within an application server (distinguished with the

Address Space ID of the Servant). The base runtime.properties file should be changed for changes to persist across application server restarts.

For more information about log and trace settings in these files, see Appendix A, "Runtime properties files," on page 469.

## Log and trace file names

The base log and trace file names in runtime.properties are used as templates to generate unique sets of log and trace files for each application server. The default base trace file name is *trace.txt*; the default base log file name is *messages.txt*. The application server name is appended to the base file name to generate the template for the log and trace files for an application server. The application server name is the same as the fully qualified name of the application server, as defined by WebSphere. This name is *cell_node_server*, where:

- *cell* is the name of the WebSphere cell.
- *node* is the name of the WebSphere node.
- *server* is the name of the WebSphere server.

The application server running HATS is the concatenation of the underscore (_) character, followed by the application server name, followed by another underscore (_) character.

Using the trace file as an example, the name becomes *trace _cell_node_server_.txt*. Finally, an index (1, 2, 3, and so on) is added to this name to distinguish multiple files. With multiple trace files configured, the trace file names for this application server are *trace_cell_node_server_1.txt, trace_cell_node_server_2.txt*, and so on.

**Note:** When running on WebSphere Application Server for z/OS, an application server can have more than one Servant running, with each Servant generating unique log and trace files. To distinguish these, the server component of the name is prefixed with the Address Space ID (ASID) of the Servant (example: *trace_asid_1.txt*).

## View Log

Using the View Log panel, you can view the HATS log file, download the entire file, or clear the file. The log file records information about two kinds of events:

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

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

The log file is intended for you to read and use as a reference when troubleshooting problems. Most of the messages that appear in the log are documented in *HATS Messages*. That publication contains suggestions for actions you can take to correct problems, when necessary.

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

## Set Log Options

Using the Log Settings panel, you can control whether information and warning events are written to the log file, and define the file to which the log events are written. See “Log and trace file names” on page 382 for information about how log files are named. If the file already exists, new events are appended to it. Error events are always written to the log file. You can also define how many log files to keep and the maximum size of each log file.

## Set Trace Options

Using the Trace Settings panel, you can select one or more of the following types of trace options:

**Enable Runtime Tracing**
Enables tracing in HATS runtime for connections in your project. To trace these runtime connections at a different level, see Appendix A, “Runtime properties files,” on page 469 for information about the trace settings.

**Enable Widget Tracing**
Enables tracing in HATS runtime for widgets in your project. To trace widgets at a different level, see Appendix A, “Runtime properties files,” on page 469 for information about the trace settings.

**Enable Action Tracing**
Enables tracing in HATS runtime for event actions in your project. To trace events at a different level, see Appendix A, “Runtime properties files,” on page 469 for information about the trace settings.

**Enable Component Tracing**
Enables tracing in HATS runtime for components in your project. To trace components at a different level, see Appendix A, “Runtime properties files,” on page 469 for information about the trace settings.

**Enable Util Tracing**
Enables tracing in HATS runtime for runtime utilities in your project. To trace these runtime utilities at a different level, see Appendix A, “Runtime properties files,” on page 469 for information about the trace settings.

**Integration Object Tracing**
Enables tracing in specific Integration Objects in your project. Use the Class Name specification field to specify which Integration Objects to trace, using one or more patterns. Each pattern can contain one or more wildcard (*) character. For example, IntegrationObject.Callup* specifies that tracing is enabled for all Integration Objects that start with the letters `Callup`. To trace all Integration Objects, specify IntegrationObject.*. If multiple patterns are specified, they should be delimited with commas. HATS administrative console sets the Integration Object setting to the maximum trace level. To trace Integration Objects at a different level, see Appendix A, “Runtime properties files,” on page 469 for information about the trace settings.

**Applet Tracing**
Enables tracing of the applet. The higher level of trace will produce more tracing information. When applet tracing is turned on, it will output server side messages for troubleshooting. For more information about applet trace settings, see Appendix A, “Runtime properties files,” on page 469.

**User Macro Tracing**
Traces the playing of macros. Because it affects system performance, we recommend that you use this trace only for debugging macros.

**Display Terminal**

Enables you to view the host terminal screen of any connection as it runs in real time on the server. After you enable the display terminal, the terminal settings are shown only for newly created connections. For more information about the Display Terminal, see "Using display terminal for testing and debugging."

**Note:** You can decide whether HATS should display a dialog asking you if you want to turn on the display terminal when you Run on Server. For more information, see "Using HATS preferences" on page 126.

**Record Host Simulation Trace**

Select the **Enable Host Simulation Recording** box to enable host simulation recording during runtime. Recording starts the next time a host session starts and ends when the host session is closed, for example when the user clicks the disconnect button on the application keypad. If the user closes the Web browser while tracing is active, the trace will continue until the HTTP session times out. The trace file is saved in the host simulations folder on the server, for example, *<WebSphere install root>*\AppServer\profiles\AppSrv01\installedApps*<Node ID>**<ear file name>**<ApplicationName>*.war\WEB-INF\profiles\hostsimulations. Trace files are named using the following template, `ApplicationName_ConnectionName_Date(yyyymmdd)_Time(hhmmss)_Number`, for example, MyApplication_main_20060101_134543_1.

In order to trace multiple host sessions concurrently, host simulation uses multiple TCP/IP ports. Use the **TCP/IP Port Range for Recording** fields to set the range of port numbers for server runtime usage. The default starting port in the range is **7021**, and the default ending port is **7050**.

**IBM Service Tracing Options**

HATS administrative console also includes a **Display Service Tracing Options** check box with which you can turn on traces (if instructed to do so by IBM service). You should not use these traces unless you are asked to do so by IBM service.

**Trace Output**

Enables you to control whether trace information is written to the trace file, and define the file to which the log events are written. See "Log and trace file names" on page 382 for information about how trace files are named. If the file already exists, new trace information is appended to it. You can also define how many trace files to keep and the maximum size of each trace file.

# Display terminal functions

## Using display terminal for testing and debugging

When debugging applications on a test system, you can control whether a terminal window that displays host screens is created on the server display. You can use host screens displayed in the terminal window to observe interactions between the HATS applications and the host application at runtime. You can also interact with the host application using the host screen in the terminal window. Terminal screens are created only for connections established while this option is turned on. They are not created for existing connections or for connections that are idle. However, the Host Connections panel in HATS administrative console includes a **Toggle Display Status** button that turns display terminal on and off for selected connections.

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

You can turn display terminal on for any new host connections by selecting the **Enable Display Terminal** box on the **Trace Settings** panel under the **Troubleshooting** heading in the left navigation frame; however, you can use **Toggle Display Terminal** on the **Host Connections** panel to turn display terminal on and off for a particular connection at any time, regardless of whether you have selected the **Enable Display Terminal** box on the **Trace Settings** panel.

**Notes:**

1. For the display terminal to work properly when WebSphere Application Server is started as a service on the Windows platform, you must select **Allow service to interact with desktop** in the service settings for WebSphere Application Server.
2. For the display terminal to work properly with WebSphere Portal V7 on the Windows platform, the portal server must be started as a normal Windows process and not as a Windows service.
3. For information about displaying graphics, required for the display terminal function, using the Native Abstract Windowing Toolkit (NAWT) on the IBM i platform, see the section, Running your Java application on a host that does not have a graphical user interface, in one of the following documents as appropriate,
   - *System i Programming IBM Developer Kit for Java Version 5 Release 4*, at http://publib.boulder.ibm.com/infocenter/iseries/v5r4/topic/rzaha/rzaha.pdf
   - *System i Programming IBM Developer Kit for Java Version 6 Release 1*, at http://publib.boulder.ibm.com/infocenter/iseries/v6r1m0/topic/rzaha/rzaha.pdf
   - *IBM i Programming IBM Developer Kit for Java 7.1*, at http://publib.boulder.ibm.com/infocenter/iseries/v7r1m0/topic/rzaha/rzaha.pdf
4. For information about configuring the display terminal functions on other platforms, see the section, IBM i, z Systems, and Unix platform-specific information, in the HATS Knowledge Center at http://www.ibm.com/support/knowledgecenter/SSXKAY_9.7.0.

# Chapter 19. WebSphere Portal and HATS

IBM WebSphere Portal gives you a single point of access to Web content and applications, personalized to each user's needs.

HATS integration with WebSphere Portal lets data flow freely between HATS, WebSphere Portal, and other Portal applications. All of the relevant features of WebSphere Portal are also available for exploitation within HATS by built-in tooling or straightforward Java development. The majority of HATS functionality is preserved when running HATS in the WebSphere Portal environment. See "HATS portlet considerations and limitations" on page 396 for more information.

HATS applications can run directly within WebSphere Portal as portlets, enabling WebSphere Portal and HATS to interact directly. You can develop HATS portlets that comply with the standard Java Portlet Specification API (JSR 168 or JSR 286), hereafter referred to as the standard portlet API.

**Note:** Throughout this document, HATS portlets that comply with the standard portlet API (JSR 168 or JSR 286) are referred to as standard portlets. Where a distinction is required, JSR 168 or JSR 286 is noted.

HATS Business Logic classes are able to use the WebSphere Portal APIs, and HATS JSPs are able to include WebSphere Portal JSP tag libraries. For more information about programming portlets and advanced portal features, refer to the *HATS Web Application Programmer's Guide*.

A HATS portlet project is targeted at transformation oriented applications. For adapting Model 1 Integration Object based applications to run in WebSphere Portal, refer to the *HATS Web Application Programmer's Guide*. For converting Struts Integration Object based applications to run in WebSphere Portal, refer to documentation for your level of WebSphere Portal at http://www.ibm.com/developerworks/websphere/zones/portal/proddoc/index.html, and search on `Struts`.

For more information and help regarding the Portal Tools, select the **Help** menu on the Rational SDP toolbar and select **Help Contents**. Click **Developing portal and portlet applications** from the help list on the left to access documentation for portlet developers.

## Creating HATS portlets

There are two ways in which you can create a HATS portlet:

- You can create a new one using the HATS Create a Project wizard and designating that the **Target server** be one of the supported WebSphere Portal servers.
- You can generate one from an existing HATS Web project. This ensures that the time spent on creating your HATS Web application will not be wasted when you move it to WebSphere Portal.

### Creating a new HATS portlet project

To create a new HATS portlet project, launch the Create a Project wizard using one of the following actions:

- Select **HATS > New > Project** from the menu bar.
- Select **File > New > HATS Project** from the menu bar.
- Click the **Create HATS Project** icon on the toolbar.

In the Create a Project wizard, on the HATS Project panel:

1. Enter a name for the portlet project.
2. Optionally, enter a description.
3. Accept the default location.
4. Select **Web** to indicate this is a Web project.

   **Note:** If the Web deployment option is disabled, this indicates that no server runtimes are defined. To define server runtimes, go to **Window > Preferences > Server > Runtime Environments** and add at least one runtime definition.
5. For the Target server field, select one of the following target WebSphere Portal servers:
   - **WebSphere Portal v7.0**
   - **WebSphere Portal v8.0**.
6. For the Enterprise application project field, enter the name of the EAR project to use for testing in the local test environment.
7. For the Portlet API field, select **JSR 286 Portlet**, or **JSR 168 Portlet** to indicate with which API this portlet will comply.
8. Select whether to include administrative console support files.

   **Note:** This option is not supported for HATS standard portlets.
9. Click **Next** and continue creating your project as normal.

This will generate a single HATS portlet project including an EAR project that can be used for testing with a local test environment.

## Generating portlets from HATS Web projects

In order to allow an existing HATS Web project to run as a portlet, the structure and content of the HATS project must be altered to meet the portlet specification. The HATS Toolkit automates the process of converting a regular HATS Web project into a portlet. The Portal Tools feature of Rational SDP is a prerequisite for this conversion process.

To generate a HATS portlet project from an existing HATS Web project, in the **HATS Projects** view, right-click the Web project and select **Generate Portlet Project**.

In the Generate Portlet Project wizard, on the Portlet Project Generation Options panel:

1. Enter a name for the portlet project.
2. For the Target server field, select one of the following target WebSphere Portal servers:
   - **WebSphere Portal v7.0**
   - **WebSphere Portal v8.0**.

**Note:** When generating a HATS portlet project from a HATS Web project, the target server of the Web project determines the target server options for the portlet project.

If the target server of the Web project is WebSphere Application Server v7.0, the target server options for the generated portlet are:

- **WebSphere Portal v7.0**
- **WebSphere Portal v8.0**.

If the target server of the Web project is WebSphere Application Server v8.0 or v8.5, the target server option for the generated portlet is:

- **WebSphere Portal v8.0**.

3. For the Enterprise application project field, enter the name of the EAR project to use for testing in the local test environment.
4. For the Portlet API field, select **JSR 286 Portlet**, or **JSR 168 Portlet** to indicate with which API this portlet will comply.
5. Select from this list the types of files to be automatically converted:
   - Transformations
   - Transformation fragments
   - Templates
   - Macro handlers
   - Cascading style sheets
6. Click **Finish**.

A HATS portlet project will be created from the selected Web project. Once the portlet is created, additional customization can be done. For more information about modifying HATS projects, see Chapter 5, “Modifying a HATS project,” on page 87.

**Notes:**

1. If you convert a HATS Web project to a HATS portlet project, you may have to manually modify the Web Express Logon (WEL) configuration file (hatswelcfg.xml) if the plug-ins configured for the Web project are not supported for the portlet project. It is recommended that you use the WEL configuration user interface to add supported plug-ins so that the WEL configuration file (hatswelcfg.xml) is updated properly.
2. After generating a HATS portlet project from a HATS Web project, changes made to one project are not reflected in the other. You can always generate a new HATS portlet project from a changed HATS Web project to reflect changes made in the Web project.
3. You cannot convert a HATS portlet project back into a HATS Web project.
4. If you have an existing HATS V4, V5, V6, V7.0, V7.1, V7.5, V8.0, or V8.5 application, you must migrate your application to HATS V9.7V9.7 before generating a portlet project. For more information about migrating to HATS V9.7 see Chapter 2, “Migrating to HATS V9.7,” on page 13.

# Working with HATS portlets

A HATS portlet project is still a HATS project, meaning all the HATS wizards and editors can continue to be used on the project. A HATS portlet project is also a portlet project which also means all the Portal Tools editors and runtime configuration can be used on the project. A HATS portlet project appears in the tree with a different icon  .

## Connection parameter and global variable overrides

You can allow connection parameters and global variables for your portlets to be overridden by customizing the project settings. For more information, see "Connection parameter overrides" on page 106 and "Global variable overrides" on page 115.

### Standard portlets

For HATS standard portlets, if you allow overrides in the project settings, then the user can override connection parameters and global variables by selecting **Personalize** from the portlet menu on the toolbar and switching to edit mode.

**Note:** HATS does not support portlet edit mode on WebSphere Application Server. If you deploy your HATS standard portlet to run directly on WebSphere Application Server, instead of using portlet edit mode, the user can override connection parameters and global variables in the URL used to access the portlet.

Once in portlet edit mode, a settings page is displayed for the user. This page allows the user to override connection parameter and global variable settings. These functions are equivalent to the HATS functions for Web applications that allow a user to override settings using the URL when the application is first accessed.

On the settings page, in the connection parameter section, the connection parameter drop-down lists the Host On-Demand session parameters that you allow to be overridden in the project settings. The user sets an override by selecting a parameter from the drop-down list, entering a value in the **Value** field, and clicking **Add**. By default, the override setting is enabled when first added. It can be disabled by selecting **false** from the **Enabled** drop-down. An override can be deleted by selecting the check box next to the parameter name and clicking **Delete**.

In the global variable section, the user sets an override by entering a global variable name and value, and clicking **Add**. After adding an override, the global variable can be identified as a shared global variable by selecting **true** from the **Shared** drop-down. By default, the override setting is enabled when first added. It can be disabled by selecting **false** from the **Enabled** drop-down. An override can be deleted by selecting the check box next to the global variable name and clicking **Delete**.

When the user clicks **Save and Exit**, the settings are saved in the portlet preferences.

**Notes:**

1. The values entered by the user are not validated.
2. The user must disconnect and restart the portlet application before the new values take effect.

In addition to allowing users to set overrides, you can programmatically provide connection parameter and global variable overrides by extending the HATS JSR 286 or JSR 168 entry portlet, as appropriate. For more information, see the section, Extending the Entry portlet, in the *HATS Web Application Programmer's Guide*

**Note:** If you supply connection parameter or global variable overrides through the extended portlet class and the user provides the same overrides using portlet edit mode, then the values provided by the user in edit mode override the values you supply in the extended portlet class.

.

## Portlet communication

WebSphere Portal supports multiple ways for portlets to exchange or share information. For more information refer to documentation for your level of WebSphere Portal at http://www.ibm.com/developerworks/websphere/zones/portal/proddoc/index.html and search on `Portlet communication`.

This section describes the HATS support for WebSphere Portal portlet communication.

### JSR 168 portlets

HATS JSR 168 portlet support uses the WebSphere Portal cooperative portlet mechanism for providing a portlet communication solution.

JSR 168 portlets use properties to communicate with each other. A HATS JSR 168 portlet supports the ability to send properties to and receive properties from other JSR 168 portlets, HATS or non-HATS. Portlets must be wired in order to communicate.

**Sending a property from a HATS JSR 168 portlet:**

1. Create a global variable containing the value of the property to send. You can do this in several ways including: executing an **Extract global variable** action or a **Set global variable** action. Only global variables that can contain zero or more strings (no other type) can be used in JSR 168 portlet communication. Global variables can be thought of as wrappers for the values of JSR 168 properties.
2. Add a **Send global variable** action to a HATS screen event or to an unmatched screen application event to send the value of the global variable as a property. For information about how to do this, see "Send global variable action" on page 159.

   This action can be used in conjunction with the **Extract global variable** action to send input entered by the user from one portlet to another. For example, you can add an **Extract global variable** action between the **Apply transformation** action and the **Send global variable** action to capture data the user enters and send it to other portlets.
3. When the HATS JSR 168 portlet project is run and the **Send global variable** action is executed, the property name and the string contents of the specified global variable are passed to WebSphere Portal which then notifies the portlets wired to this HATS JSR 168 portlet to receive the property.

**Receiving a property in a HATS JSR 168 portlet:**

1. Use portlet settings to define property names and the corresponding global variables to contain the property values that can be received by this portlet.

Access portlet settings from the Project settings editor by clicking the **Other** tab then **Portlet**. For more information about how to do this, see "Portlet settings" on page 124.

2. Once WebSphere Portal has received a property from a source portlet, the target portlet is notified. The HATS JSR 168 portlet runtime service retrieves the property value from WebSphere Portal and stores it in the specified global variable. Again, only global variables that contain a string (or strings) can be used with JSR 168 communication.

**Communicating with non-HATS JSR 168 portlets:** Rational SDP provides GUI support to implement cooperative portlets. When configuring non-HATS JSR 168 portlets to communicate with HATS JSR 168 portlets, perform the following steps:

1. Open the Project Explorer view from the Rational SDP menu bar by selecting **Window > Show View > Other > General > Project Explorer**.
2. From the Project Explorer view, right-click the **Portlet Deployment Descriptor** node under the portlet project.
3. Select **Cooperative > Enable this Portlet to Send Data (Source)** or **Cooperative > Enable this Portlet to Receive Data (Target)** to start the wizard to enable cooperative portlet support for the selected portlet.
4. Ensure the following settings are correct in this wizard:
   - Data Type URI = `http://com.ibm.hats.portlet.pb#<property_name>Type`. The URI contains the namespace followed by the type of the property. Properties sent, or received, by HATS JSR 168 portlets use `com.ibm.hats.portlet.pb` as the namespace.

     **Note:** The word, `Type`, must begin with an uppercase `T`.
   - Java Type = `java.lang.String`. HATS portlets send, or receive, properties as String.

**Editing the Data Type URI:** When you configure a Send global variable action or a Portlet setting to receive a property, HATS defaults the Data Type URI for the property to be `http://com.ibm.hats.portlet.pb#<property_name>Type`. This URI is only used when you wire the source and target portlets. WebSphere Portal requires that the source and target portlets use the same URI in order to send and receive the property. If for any reason you need to configure the URI differently than the default, you can modify the URI for a property using the Portal toolkit by following these steps:

1. Open the Project Explorer view from the Rational SDP menu bar by selecting **Window > Show View > Other > General > Project Explorer**.
2. In the Project Explorer view, expand the project to the property that you want to modify. The properties can be found under the **Portlet Deployment Descriptor** node.
3. Right click on the property and select **Edit Data Type**.
4. Modify the **Data Type URI** in the Edit Cooperative Source or Edit Cooperative Target wizards.

## JSR 286 portlets

HATS JSR 286 portlet support uses the WebSphere Portal portlet events mechanism for providing a portlet communication solution.

JSR 286 portlets use events to communicate with each other. Events are defined with a name and value type. Event names within a single portlet must be unique. The portlet declares if it is going to publish (send) or process (receive) an event.

**Sending an event from a HATS JSR 286 portlet:**

1. Create a global variable containing the object, or objects, to send. You can do this in several ways including: executing an **Extract global variable** action or a **Set global variable** action. Any HATS global variable that contains one or more Serializable objects can be used in communication. Global variables can be thought of as wrappers for the objects of JSR 286 events.
2. Add a **Send global variable** action to a HATS screen event or to an unmatched screen application event to send the objects in the global variable as a portlet event. For information about how to do this, see "Send global variable action" on page 159.

   This action can be used in conjunction with the **Extract global variable** action to send input entered by the user from one portlet to another. For example, you can add an **Extract global variable** action between the **Apply transformation** action and the **Send global variable** action to capture data the user enters and send it to other portlets.
3. When the HATS JSR 286 portlet project is run and the **Send global variable** action is executed, the event name and the contents of the specified global variable are passed to WebSphere Portal which then notifies the portlets wired to this HATS JSR 286 portlet to receive the event.

**Receiving an event in a HATS JSR 286 portlet:**

1. Use portlet settings to define event names, the corresponding global variables to contain the events' objects, and the object types, that can be received by this portlet. Access portlet settings from the Project settings editor by clicking the **Other** tab then **Portlet**. For more information about how to do this, see "Portlet settings" on page 124.
2. Once WebSphere Portal has received an event's object from a source portlet, the target portlet is notified. The HATS JSR 286 portlet runtime service retrieves the event's object from WebSphere Portal and stores it in the specified global variable.

**Communicating with non-HATS JSR 286 portlets:** Rational SDP provides GUI support to implement JSR 286 Portlet events. When configuring non-HATS JSR 286 portlets to communicate with HATS JSR 286 portlets, perform the following steps:

1. Open the Project Explorer view from the Rational SDP menu bar by selecting **Window > Show View > Other > General > Project Explorer**.
2. From the Project Explorer view, right-click the **Portlet Deployment Descriptor** node under the portlet project.
3. Select **Event > Enable this Portlet to Publish Event** or **Event > Enable this Portlet to Process Event** to start the wizard to enable JSR 286 Portlet event support for the selected portlet.
4. Complete the wizard remembering that the event name and value type must match between publishing (sending) and processing (receiving) portlets.

## Wiring portlets for communication

Portlet wires are used to direct the information flow between portlets that communicate using JSR 286 Portlet events or the WebSphere Portal Cooperative portlets API. Portlet wiring refers to connecting two portlets using the portal server's wiring tool for the purpose of sending and receiving a portlet event or a cooperative portlet property. Portlets are wired for only a single event (or property) and both portlets must give this event (or property) the same name although the associated global variable names can be different. In other words, if two portlets

are going to send and receive two events (or properties), they need two wires connecting them, one for each event (or property). Without completing the wiring step, data is not transmitted.

### Custom PDTs and custom tables

Before deploying a portlet project with a custom printer definition table (PDT) or custom tables to a portal server, you must copy files or folders to the `\Web Content` directory of the HATS portlet project. The files or folders for each type of addition are shown below:

- Custom printer definition table (PDT): Compiled PDT files or the folders that contain compiled PDT files.
- Custom table: The customtable folder that contains the definition of custom tables.

### Web Services for Remote Portlets

Web Services for Remote Portlets (WSRP) is a standard that portals can use to provide portlets, applications, and content as WSRP services, and other portals can integrate the WSRP services as remote portlets for their users. Standard and IBM HATS portlets can be provided as remote portlets that can be consumed by other portals. For more information refer to documentation for your level of WebSphere Portal at http://www.ibm.com/developerworks/websphere/zones/portal/proddoc/index.html and search on `Using WSRP services`.

## Testing HATS portlets

If a WebSphere Portal Test Environment is configured in Rational SDP, you can use it to test your HATS standard before you deploy them to a production Portal server.

You can also test your HATS standard portlets using the WebSphere Application Server local test environment.

To test a HATS portlet project, use the same procedure used for testing HATS Web projects. For more information see “Overview” on page 87.

## Exporting HATS portlets

To export a HATS portlet for deployment, from the HATS Projects view right-click the portlet project and select **Export Project**. Enter the target file name for the exported .war file in the **Destination** drop-down, and click **Finish**. Only the .war file is exported. The .ear file is used only for testing in the local test environment.

Once you have exported your HATS portlet .war file, you install it into WebSphere Portal using the WebSphere Portal administration tools.

If installing a standard portlet into WebSphere Application Server, use the WebSphere Application Server administrative console and consider the following instructions:

- During the install, you must provide a context root name. This context root name is required in order to start the portlet .war file after installation is complete. During the install, deploy the .war file to server1, or the defined webserver instance.
- After installing the portlet .war file, you can check the context root name and portlet name using the WebSphere Application Server administrative console by

expanding **Applications** and clicking **Enterprise Applications**. Click your .war file and click **Context Root for Web Modules**. Note the value for Web module and ContextRoot. These values are used to start the HATS Application.

- To run the HATS application enter its URL into a browser. For example, enter `http://host:port/context_root_name/portlet_name` where `host` and `port` represent the host name and port for the WebSphere Application Server host where you installed your portlet. For `context_root_name`, use the value you provided for Context root when installing the portlet .war file, and for `portlet_name`, use the value you provided for the Web module.

# Administering HATS portlets

You can administer HATS portlets using the HATS administrative console. HATS standard portlets, whether installed on Portal Server or WebSphere Application Server, can be administered using a stand-alone HATS administrative application.

## Using a stand-alone HATS administrative application

To administer HATS standard using a stand-alone HATS administrative application, follow these steps:

1. Create a stand-alone HATS administrative project in the HATS Toolkit by selecting **File > New > Project > HATS > HATS Administration Project**. The HATS administrative project only appears in the HATS Toolkit on the **Navigator** tab.
2. Export the HATS administrative .ear project to an .ear file by right-clicking the .ear project and selecting **Export > Java EE > EAR**.
3. Deploy the .ear file for the stand-alone HATS administrative application to WebSphere Application Server using the WebSphere Application Server administrative console.

   **Notes:**

   a. Install the .ear file to the same WebSphere Application Server system as the WebSphere Portal server. Where a standard HATS portlet is installed directly to WebSphere Application Server, install the .ear file to the same server.

   b. Ensure the module is mapped to the WebSphere Application Server, for example, server1, and not WebSphere Portal Server.

   c. For WebSphere Application Server V6.x, ensure the **All Authenticated** option on the **Map security roles to users/groups** panel is checked for the HATS Administrator role. For WebSphere Application Server V7.x and V8.x, ensure the **All Authenticated in Application Realm** option is selected using the **Map Special Subjects** drop-down on the **Map security roles to users/groups** panel.
4. The user id used to administer the portlets must be defined as a WebSphere Application Server administrative console user with **Administrator** authority. To do so, follow these steps:

   a. From the WebSphere Application Server administrative console, select **Users and Groups > Administrative User Roles > Add**.

   b. Enter the user id used to administer the portlets and map the user id to the **Administrator** role.

   **Note:** For more information, see "HATS administrative console and WebSphere security" on page 375.

5. Start the HATS administrative application using the WebSphere Application Server administrative console.
6. Access the stand-alone HATS administrative console. For example, use the URL, `http://<washostname>/<hatsadminprojectname>/hatsadmin/admin`.

   **Note:** When accessing the HATS administrative console, you must use a different browser than the one being used to access the portlets.

   a. Select **Getting Started > Management Scope**.
   b. Click **New**.
   c. In the **Host** field enter the same host name as your target WebSphere Application Server host.
   d. In the **Port** field enter the BOOTSTRAP_ADDRESS port for your target server. To find the BOOTSTRAP_ADDRESS port for your target server, access the WebSphere Application Server administrative console and follow these steps:
      1) Select **Servers > Application Servers**.
      2) Click your target server (either WebSphere Portal server, or WebSphere Application Server, for example server1, in the case where a standard HATS portlet is installed directly to WebSphere Application Server.
      3) On the Configuration tab under Communications expand **Ports**.
      4) Note the BOOTSTRAP_ADDRESS port for your target server.
   e. Click **OK**.
   f. Select the newly added host.
   g. Do not select **Manage all applications within the scope selected above**.
   h. You can now see a list of HATS portlets. Use the HATS Administrative console to administer one portlet application at a time by selecting one from this list.

# HATS portlet considerations and limitations

## Common considerations and limitations

Following is a list of considerations and limitations when working with HATS standard portlets:

- Portlets can only be installed in production as Web application archives (.war files). HATS only has one project per .war file, therefore you cannot export a set of HATS applications as a single portlet application.
- Cookies are required to authenticate within WebSphere Portal, therefore you must run HATS in WebSphere Portal with cookies enabled.
- JSP files, such as templates, transformations, transformation fragments, and macro handlers, copied from HATS Web projects to HATS standard portlet projects, will not work without conversion. You can use the **Convert JSP for Portal** action to convert these JSP files.
- HATS options appear in the Rational SDP New Portlet wizard, but selecting these options only generates an empty portlet without HATS resources.
- When a HATS portlet is on the same portal page with other portlets (HATS or non-HATS), if you disconnect the HATS portlet, it is automatically restarted as a result of the page refresh that occurs when you interact with any other portlet on the same portal page. HATS standard portlets exhibit this behavior when the portal page theme is set to `PortalWeb2`.

- When HATS portlets, that use certain HATS predefined templates, run on the same portal page, the templates may interfere with one another. This is caused if all the templates use the same cascading style sheet (CSS) rule names. This only affects CSS rule names in the template.jsp file, because the names are not prefixed with `div#<portletName>`. To work around this limitation do one of the following:
  - Use any of the following HATS predefined templates which do not have this limitation:
    - Finance
    - Industry
    - Medical
    - Research
    - Transport
  - Edit the template.jsp file and add the prefix `div#<portletName>` to each CSS rule. For example, if you have this in your template.jsp file:

    ```
    A.BOTTOMBAR:visited,
    A.BOTTOMBAR:link,
    A.BOTTOMBAR:active {color: #55839A; font-size: 1em;}

    table.ApplicationKeypad {
     color: blue;
     white-space: nowrap;
    }
    ```

    Assuming that the portlet project in named `hatsPort914`, change the example as shown below:

    ```
    div#hatsport914 A.BOTTOMBAR:visited,
    div#hatsport914 A.BOTTOMBAR:link,
    div#hatsport914 A.BOTTOMBAR:active {color: #55839A; font-size: 1em;}

    div#hatsport914 table.ApplicationKeypad {
     color: blue;
     white-space: nowrap;
    }
    ```
- The following HATS functions, actions, and settings are not supported in HATS portlets:
  - Dojo widgets.
  - The **Enable HTTP Compression** setting.
  - The **asynchronous update** applet.
  - The **disconnectOnClose** connection parameter.
  - The screen combination option, **use dynamic, cached content loading**.
  - HATS interoperability with WebFacing applications.
  - HATS standard portlets can only run on WebSphere Portal or WebSphere Application Server platforms.
  - If you add Web Express Logon (WEL) support to a HATS standard portlet using the WebSphere Portal Credential Vault plug-in, it will run on supported WebSphere Portal servers. However, it will fail to log onto the host system when running directly on WebSphere Application Server which does not have Credential Vault Service support.
  - The following HATS actions, settings, and functions are not supported in HATS standard portlets:
    - The ability to add HATS administrative console support to the portlet project.

- The ability to generate sample code in the New Business Logic wizard for functions such as accessing and retrieving authentication data, portlet messaging, or using a property broker.
- The Portlet Prepresentation (Add Block Delimiter) action.

– When using the Send global variable action, the following actions are the only actions supported between the Apply transformation action and the Send global variable action:
- Execute business logic
- Extract global variable
- Insert data
- Set global variable
- Remove global variable
- Perform macro transaction
- Pause

– When using Integration Objects:
- Only Model 1 Web pages are supported. Struts and JSF Web pages are not supported.
- If you copy or rename a HATS portlet project, the Model 1 JSPs generated from a standalone or first in chain Integration Object refer to the wrong connection specification. You must modify the following statement:

  `SignOn.setHPubStartPoolName("<new_project_name>/<default_connection>");`
- When you generate a portlet project from a Web project, the Model 1 JSP pages previously generated from integration objects do not change. You must regenerate the pages, or make the changes manually.
- Global variables and templates are not supported.
- To run an Integration Object that is configured to use Web Express Logon in standard portlets, you must ensure PortletRequest is available in the Integration Object by calling setHPubPortletRequest(javax.portlet.PortletRequest) before executing the processRequest() method. For more detail, see Using Web Express Logon in the *HATS Web Application Programmer's Guide*.

– When viewing active host connections using the HATS administrative console, the client connection for a HATS standard portlet displays a blank IP address.

# Chapter 20. WebFacing and HATS

WebFacing technology provides the ability to convert IBM i data description specification (DDS) display file source members into a Web-based user interface for existing 5250 programs.

The WebFacing technology is provided by installable HATS features. You can install the IBM WebFacing Tool for IBM i feature as part of the HATS installation process.

The IBM WebFacing Tool for IBM i feature of the HATS Toolkit provides the ability to convert IBM i data description specifications (DDS) and user interface manager (UIM) help panel source members into a Web-based user interface for existing 5250 programs.

For more information about installing the WebFacing features, see the section, Installing HATS Toolkit in *HATS Getting Started*. For more information about the WebFacing capabilities, see the Developing WebFacing Applications topic in the product help under the Developing HATS Applications topic. WebFacing information is also available in the HATS Knowledge Center at http://www.ibm.com/support/knowledgecenter/SSXKAY_9.7.0.

## HATS interaction with WebFacing

HATS provides two methods for interacting with the WebFacing technology. One method allows a HATS Web application to use a connection to a WebFacing server instead of a Telnet 5250 server. The second method allows a HATS Web application and a WebFacing application to be linked together and interoperate as a single enterprise application that can use a single connection to a WebFacing server. Each method is described below.

### HATS connection to a WebFacing server

HATS provides two connectivity options for connecting to 5250 host screen applications:

- 5250 (Telnet)
- 5250W (WebFacing)

There is no difference in the data flow between HATS and a Telnet 5250 server or HATS and a WebFacing server. Both connections provide access to 5250 applications, but the connection to the WebFacing server does not require online transaction processing (OLTP) capacity (interactive CPW). This significantly reduces the overall cost of using HATS with 5250 applications by removing the interactive feature requirement associated with TN5250 session connectivity.

To enable a connection to a WebFacing server, select the **5250W** connection type from the **Type** field in the New Project wizard, the New Connection wizard, or the Connection editor. The default port for the WebFacing server connection is **4004**. See “Basic” on page 132 for more information.

You can enable existing projects to take advantage of the WebFacing connectivity option by selecting **5250W** from the Connection editor. The WebFacing server connection can serve as the default connection or as a background connection in the project.

Connection to the WebFacing server requires the server process to be running on the 5250 host system. To start the WebFacing server, enter the following command from the operating system command line: `STRTCPSVR SERVER(*WEBFACING)`.

To stop the WebFacing server, enter the following command from the operating command line: `ENDTCPSVR SERVER(*WEBFACING)`.

To ensure that the WebFacing server is running, enter the following command from the command line to list all WebFacing jobs: `WRKACTJOB JOB(QQF*)`.

The WebFacing server job, QQFWFSVR, and the WebFacing virtual terminal server job, QQFVTSVR, should both be displayed. It is normal to see more than one QQFVTSVR job in the list of jobs.

**Note:** Check the system value QAUTOVRT. The value for QAUTOVRT determines the number of VT jobs that can be auto started. The WebFacing server relies on VT jobs. If the value is 0, no browser sessions can be launched. If necessary, set QAUTOVRT to *NOMAX or some value greater than 0. To change the value for QAUTOVRT, use the WRKSYSVAL system command to work with system values. WebFacing Server connection jobs are still listed under the QINTER subsystem and marked as INT (interactive), but they are not counted as part of the interactive capacity of the server.

### Limitations

Secure Socket Layer (SSL) is not supported for 5250W connections between HATS and the host application.

Pre-naming of workstation IDs is not supported for 5250W connections. A specific workstation ID cannot be specified before connection.

## HATS interoperability with WebFacing applications

### Overview

HATS interoperability with WebFacing applications provides the ability to perform data description specification (DDS) display file source member transformation, as well as 5250 data stream transformation, within the same Web application. This function is particularly useful for those who have built a DDS source-based transformation application, using the WebFacing tool, that need to interact with a host application that has been transformed using the HATS tool. This function is also useful for HATS users who want to take advantage of transformation, based on DDS display file source members, provided by the WebFacing tool.

Interoperability, in this context, means that a single HATS Web application interoperates with, or works along side, a single WebFacing application. The combined Web application has the capabilities of both tools, with some limitations, but the capabilities are cleanly separated within the Web application. To create a combined Web application, a single HATS Web project is linked with a single WebFacing project and packaged into a single enterprise application, or .ear file. This enterprise application contains the runtimes of both products, along with a common runtime component that is used to create and interact with a shared connection to the 5250 backend host.

For details about how to implement interoperating HATS and WebFacing applications, see the *Create a Linked WebFacing and HATS Application* tutorial at the IBM Host Access Transformation Services Education Assistant Web site at http://publib.boulder.ibm.com/infocenter/ieduasst/rtnv1r0/topic/com.ibm.iea.hats/plugin_coverpage.html. For more information and a demonstration of this function, see the IBM Enterprise Modernization Demos at http://rational.dfw.ibm.com/atdemo/atdemo_hats_wf_recorded.html.

**Note:** The use of this function requires that the HATS WebFacing feature is installed.

# Chapter 21. Security and Web Express Logon

HATS offers security for your applications. Using Secure Sockets Layer (SSL), HATS provides a secure connection between your HATS application and the Telnet server it connects to.

HATS also supports Secure Shell (SSH) for VT connections.

Web Express Logon (WEL) gives you a mechanism to authenticate users (that run HATS Web applications and portlets) and provides them with single signon capability.

Using Kerberos services tickets can automate sign-on for rich client applications that use 5250 Telnet sessions and run on Windows domain clients.

By supporting Java 2 security, HATS helps you provide protection for system resources and APIs within your WebSphere Application Server system.

For information about the effects of WebSphere Application Server security on the use of the HATS administrative console see Chapter 18, "Using HATS administrative console," on page 375.

For WebSphere Application Server security information, visit the WebSphere Application Server online library at http://www.ibm.com/software/webservers/appserv/was/library/.

## Enabling SSL security

For Web applications, SSL security between the user's browser and the HATS application requires an HTTPS connection. This requires that both HTTP server and WebSphere Application Server be configured to support HTTPS. The HTTP server certificate is stored in the browser certificate store for the browser-HTTP server connection.

The HATS SSL configuration discussed in the remainder of this section is used to configure SSL between the HATS application and the Telnet server (which must be configured with an SSL port). This HATS SSL configuration is supported for HATS Web, rich client, and EJB applications, and for HATS portlets.

**Note:** SSL is not supported for 5250W connections between HATS and the WebFacing server.

To enable SSL between the HATS application and the Telnet server, select the **Enable SSL** check box on the **Security** tab of the connection editor. For more information see "Security" on page 139. By enabling SSL for a connection, you request that data flowing over the connection be encrypted to secure the connection.

Selecting the **Use JSSE** check box enables the use of TLS v1.0, TLS v1.1, or TLS v1.2 using the Java Secure Socket Extension (JSSE) security library, instead of SSLite, for the connection between HATS and the HOST system. If not selected (default option), SSLite library is used, and TLS v1.1 and TLS v1.2 are not available for the connection.

**Note:** The IETF Internet-Draft, TLS-based Telnet Security, defines the protocol for doing the SSL handshake over a TLS-based Telnet connection. If the Telnet server you are connecting to supports this protocol, you must add the SSLTelnetNegotiated property to the advanced connection settings of your connection definition. The advanced connection settings are found on the **Advanced** tab of the connection editor, see "Configure optional, advanced connection settings" on page 135. Set the value of the property to **true**.

HATS uses Host On-Demand technology to provide connection support from HATS applications to 3270 and 5250 applications using Telnet protocols. HATS uses the SSL support provided by Host On-Demand technology for securing these connections. Using a secure connection over SSL encrypts data flowing over the connection and thus protects it against observation by a third party.

For a connection to be secured, both the HATS application and the Telnet server it is connected to must support SSL. To secure the connection, the Telnet server must provide a certificate, which is used in encrypting the data.

When connection establishment is attempted, HATS receives the certificate from the Telnet server and determines whether to accept or reject the connection. HATS searches its built-in keystore file for a signer certificate that matches the Telnet server's personal certificate. The HATS keystore file contains a set of well-known certificates including Verisign, Thawte, and RSA. If the Telnet server is using a valid well-known certificate, it will be accepted because it will match one of the well-known certificates that are provided with HATS. In this case, there is no need to create a keystore file containing the certificate - the needed signer certificate is already in the HATS built-in keystore file.

If the Telnet server is not using a well-known certificate, a keystore file containing a valid signer certificate must be created and configured to HATS. This certificate can be obtained by opening the Telnet server's keystore file, extracting the certificate as a binary .der file, and using the Certificate Management tool (also known as the IBM Key Management tool), importing the .der file into the keystore file you use with HATS.

For example, if the Telnet server platform supports the IBM Key Management tool, to extract the certificate file from the Telnet server's keystore file, take the following steps:

1. Start the IBM Key Management tool.
2. Click **Key Database File** and select **Open**. For information about opening IBM CMS keystore files, see "Using IBM CMS keystore files" on page 425.
3. Select the **Key database type** for the Telnet server's keystore file and then **Browse** to the directory containing the file.
4. Click **OK**.
5. Under **Key database content**, select **Signer Certificates** from the drop-down list.
6. Select the certificate you want to extract and click **Extract**.
7. For **Data type** select **Binary DER data**. If the certificate is in ASCII format, select **Base64-encoded ASCII data**.
8. Give the certificate file a name and location and click **OK**.

To create a keystore file to use with HATS that includes the certificate file you extracted from the Telnet server's keystore file, take the following steps:

1. Copy the certificate extracted from the Telnet server's keystore file to your HATS development system.
2. Click **Start > All Programs >** *IBM Rational SDP package group* **> IBM Rational HATS V9.7 > Certificate Management** (where *IBM Rational SDP package group* is the name of the Rational SDP package group you have installed.
3. Click **Key Database File** and select **New**.
4. For the **Key database type**, select **PKCS12** or**JKS** (if you are using JSSE). Give the file a name with an extension of .p12 or .jks (if you are using JSSE) and a location, and click **OK**.
5. Type in a password, confirm it, and click **OK**. For the JKS file, the password must contain at least 6 characters.
6. Under **Key database content**, select **Signer Certificates** from the drop-down list and click **Add**.
7. For **Data type** select **Binary DER data**. If the certificate is in ASCII format, select **Base64-encoded ASCII data**.
8. **Browse** to find and select the certificate you extracted from the Telnet server's keystore file and click **OK**.
9. Enter a label for the certificate and click **OK**.
10. Exit the Certificate Management tool.

**Note:** For more information about Certificate Management, see "Using IBM Certificate Management for HATS applications" on page 420.

If a certificate is required by HATS, you can use options on the **Security** tab of the connection editor to configure how HATS finds the keystore file containing the certificate. Use the **Import** button to import the keystore file into your project. Or, use the **Use PKCS12 keystore at a specific path** option to specify that the keystore file will not be contained within your project but will exist elsewhere on the target runtime system. This option is useful if you want to update the keystore file without having to redeploy the HATS application. If you import the keystore file, HATS copies it to the root of the EAR project (for Web projects), to the \Web Content\WEB-INF directory (for portlet projects), or to the runtime extension plug-in (for rich client projects). If imported, the keystore file becomes a part of the HATS project, and it is packaged with the rest of the project files when you export the project. For more information see "Security" on page 139.

**Notes:**

1. Multiple certificates can be added to a single keystore file.
2. Multiple HATS projects can use the same keystore file, either by importing the same keystore file in the EAR project (for Web applications) or in the runtime extension plug-in (for rich client applications), or by referencing the same keystore file in the **Use PKCS12 keystore at a specific path** option.
3. Each connection within a single HATS project can reference the same or different keystore files.

For more information about WebSphere Application Server security, go to http://www.ibm.com/software/webservers/appserv/was/library/. For information about IBM HTTP Server security, go to http://www.ibm.com/software/webservers/httpservers/library.html.

## Enabling SSH security

To configure Secure Shell (SSH) for a VT connection:

1. In the HATS Projects view, expand the Connections folder and double-click the entry for the VT connection.
2. In the connection editor, click the **Advanced** tab.
3. Under Configure optional, advanced connection settings, click **Add**.
4. In the Add Parameter dialog, from the drop-down list for the Name field, select the **SecurityProtocol** parameter, set the Value field to SESSION_PROTOCOL_SSH, and click **OK**.
5. Click **Add**.
6. In the Add Parameter dialog, from the drop-down list for the Name field, select the **userID** parameter, set the Value field to your user id, and click **OK**.
7. Click **Add**.
8. In the Add Parameter dialog, from the drop-down list for the Name field, select the **userPassword** parameter, set the Value field to your password, and click **OK**.

# Using Web Express Logon (WEL)

## Overview

Web Express Logon (WEL) allows your users to access host applications using only their network security credentials. It provides a means for a HATS application to accept user network credential information, previously authenticated by a network security layer, and use it to generate host credentials to be used instead of requiring a HATS user to navigate host logon screens.

WEL works in conjunction with your company's network security application to maintain company security while allowing users to log on to host systems without having to re-enter their user IDs and passwords. It includes the following benefits:

- **Ease of use**: Users can log on to their network security application and access host applications without having to re-enter their IDs and passwords.
- **Reduced password-related support calls**: Users are less likely to call the company support line because of forgotten or misplaced passwords.
- **Increased productivity**: Users can log on only once in an environment that has multiple methodologies for defining user IDs, passwords, and authentications.

A variant called Certificate Express Logon is also supported which allows authentication by using X.509 certificates rather than by network security applications.

## Architecture

The following figure shows the WEL architecture using DCAS and RACF® as the example host credential mapper.

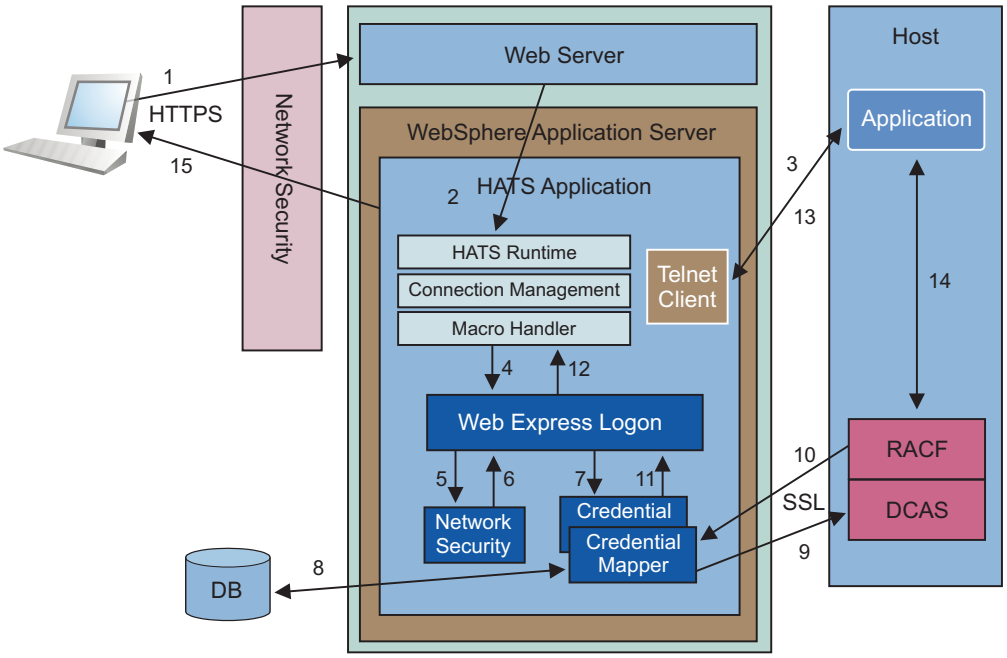



*Figure 77. WEL architecture*

1. The user submits a request to connect to a HATS application through a Network Security Application (NSA).

   The NSA authenticates the user using either an X.509 certificate or a user ID and password. After authentication the NSA authorizes access to the requested application based on policy information associated with the user. The NSA then passes the user's network credentials (network ID) through the Web server to the WebSphere Application Sever.

   IBM Tivoli Access Manager is an example of an NSA.
2. WebSphere Application Server routes the request to HATS.
3. The HATS runtime starts a Telnet connection to the host.
4. The HATS macro handler runs the WEL logon macro. See "How to create a WEL logon macro" on page 409. WEL receives a Java function call from the macro requesting that host credentials (host user ID and password or passticket) be returned.
5. WEL calls the appropriate Network Security plug-in to retrieve the user's network ID from where it was saved by the NSA.

   You must configure to HATS the Network Security plug-in that corresponds with the NSA being used. See "Network Security plug-in" on page 410.
6. The Network Security plug-in returns the user's network ID.
7. WEL calls the appropriate Credential Mapper plug-in to convert the user's network ID and host application ID to host credentials (host user ID and password or passticket).

   You must configure to HATS the Credential Mapper plug-ins that correspond with the credential mappers being used. See "Credential Mapper plug-ins" on page 410.
8. The Credential Mapper plug-in calls a function (in this case, a JDBC-accessible database such as IBM DB2®) to map the user's network ID to a host user ID.

   The Credential Mapper plug-ins provided with HATS are designed to use a JDBC-accessible database. Another possibility is to use an LDAP directory.

However, if you use LDAP, you must create your own custom plug-in. For more information see the chapter, Creating plug-ins for Web Express Logon, in the *HATS Web Application Programmer's Guide*.

9. The Credential Mapper plug-in calls the back-end credential mapper (in this case DCAS and RACF) with the host user ID and host application ID and requests a passticket.
10. RACF generates and returns a passticket.
11. The Credential Mapper plug-in returns the host user ID and passticket to WEL.
12. WEL returns the host user ID and passticket to the macro.
13. The macro inserts the host user ID and passticket onto the host logon screen.
14. The host application verifies the host user ID and passticket and allows the connection.
15. The host application screen is presented to the user.

## Planning for implementation

There are certain things you need to consider while you are planning your setup and configuration to use WEL in your HATS project. Following is a list of what information you'll need to obtain:

- What is your host type?
- What network security layer your users will enter through?
- What host applications will your users access using WEL?
- What host authentication is needed for those applications?
- Where the host authentication credentials will be stored and how to access them?
- Whether built-in Credential Mapper plug-ins are sufficient to do the job, or whether a custom plug-in needs to be written.

## Implementation

Once you have obtained the information to begin setting up WEL, you will need to take the following steps:

- Configure your network security application, if you are using one.
- Configure connections in your HATS project for WEL.

  When you select the **Use Web Express Logon** check box on the **Security** tab of the **Connection** editor and then click the **Configure ...** button, the Web Express Logon editor is opened. This editor has the following three tabs:

  – **Network Security Plugin**
  – **Credential Mapper Plugins**
  – **Source**
- Configure WEL by:
  – Identifying the network security plug-in to be used, and specifying any initialization parameters.
  – Identifying the Credential Mapper plug-ins to be used in the EAR project, the criteria for selecting each one, and the initialization parameters for each one. Order them optimally based on their selection criteria.
- Create and populate any back-end credential sources (such as DCAS).

- Configure any required keystore files needed to connect to back-end credential sources like DCAS. A Certificate Management tool (also known as the IBM Key Management tool) is provided, see "Create SSL keystore file (DCAS only)" on page 417 for more information.
- Record macros that include prompts for user ID and password to trigger WEL processing.
- Trigger these macros at appropriate times in the HATS project (connection's logon macro being one example).

## How to create a WEL logon macro

Perform the following steps to create a WEL logon macro:

1. From **HATS Projects** view, select the project connection from the **Connection** folder
2. From the HATS toolbar, click the **Open HATS Host Terminal** icon to start a session.
3. Click on the **Record Macro** icon
4. Navigate to the screen that contains the user ID input field.
5. Select **Add Prompt Action** icon from the toolbar, and the **Add Prompt Action** wizard appears. Fill in the fields. For more information, see "Add Prompt Action" on page 328.
6. Select **Use Web Express Logon** in the **Add Prompt Action** dialog. Select the prompt type of **User ID** and enter the Application ID in the Application ID field.
7. Navigate to the **Password** input field.
8. Select **Add Prompt Action** icon, select **Use Web Express Logon** with Prompt type of **Password** and enter it in the Application ID.
9. When you have completed the login process, click the **Stop Macro** icon and the **Save Macro**.

After the macro has been recorded and saved, you must configure HATS to invoke the macro. There are several methods which you can use:

- Define the WEL logon macro as the connect macro for the connection. Perform the following steps to configure WEL as the connect macro:
  1. Select the connection from the **HATS Projects** view.
  2. Open the **Macros** tab.
  3. Select the WEL macro using the **Connect macro** drop-down list.
- Invoke the WEL logon macro with the **Play Macro** option on the Connect event:
  1. In the Project Settings view, select the **Events** tab.
  2. Under Application Events, click **Connect**.
  3. On the Overview panel for the Connect event, click the **Actions** tab.
  4. On the Actions panel, click **Add**.
  5. On the Select an Action page, select **Play macro** and click **Next**.
  6. On the Define Action Properties page, select the WEL logon macro from the drop-down list and click **Finish**.
- Invoke the WEL logon macro with a **Play Macro** option on a screen customization. For more information, see "Play macro" on page 164.
- Create an Integration Object from the macro.

## Network Security plug-in

The following Network Security plug-ins can be selected for WEL. You can only have one security plug-in per .ear file. Select from the drop-down list beside **Plugin type** to display the following list:

- None (used when no network security package is being used, as with Certificate Express Logon)
- Custom (refer to the chapter, Creating plug-ins for Web Express Logon, in the *HATS Web Application Programmer's Guide*.)
- Access Manager Network Security
- WebSphere Portal Network Security Plugin

  **Note:** The WebSphere Portal Network Security Plugin will only appear if you are working with a HATS portlet project.

## Credential Mapper plug-ins

The following built-in Credential Mapper plug-ins can be selected for WEL. Click the **Add** button, select **Add built-in Credential Mapper plugin**, and then select from the following list:

- DCAS/RACF/JDBC Credential Mapper

  **Note:** DCAS and RACF are used with the zOS operating system to obtain passtickets. A JDBC-accessible repository is required to map the user's network ID to the user's host ID.
- Certificate-based DCAS/RACF Credential Mapper

  **Note:** DCAS and RACF are used with the zOS operating system to obtain passtickets. This plug-in does not require a JDBC-accessible repository because a certificate is passed directly to DCAS, and a host ID and passticket pair is returned.
- JDBC Vault Credential Mapper

  **Note:** Any JDBC/ODBC compliant repository, such as DB2, Oracle, even an Excel spreadsheet on Windows can be used. This repository is used to store host user IDs and passwords.
- WebSphere Portal Credential Vault Credential mapper

  **Notes:**

  1. This plug-in only appears in the **Add built-in Credential Mapper plugin** dialog for a portlet project.
  2. This mapper retrieves a user ID and password pair from a specified credential vault slot.
  3. Two settings are required for this credential mapper plug-in, the **SLOT_ID** (default **HATS**) and **SLOT_TYPE** (default **2**).
  4. The slot ID specifies the prefix for the slot name to use in the credential vault. The complete slot name is constructed as follows:
     - `Slot id + (space) + full host name + (space) + application id`
     - For example, `HATS zserveros.demos.ibm.com CICSA`
  5. If no application ID exists, there must be a trailing space after the host name when accessing the slot.
  6. The slot type specifies the type of credential vault slot:
     - 0 Private

- 1 Shared
- 2 Administrative (default)
- 3 System

7. The Portal administrator is responsible for setting up the slot ID and slot type.
8. For more information see the documentation for your level of WebSphere Portal at http://www.ibm.com/developerworks/websphere/zones/portal/proddoc/index.html.

- Test Credential Mapper

  **Note:** This plug-in is included to test WEL macros. It is only for testing in the HATS Toolkit because it uses hard coded host user IDs and passwords that you provide.

You also have the choice of adding a custom Credential Mapper plug-in by selecting **Add custom Credential Mapper plugin** and entering the name of the fully qualified plug-in in the text box. For information about creating a custom plug-in, see the chapter, Creating plug-ins for Web Express Logon, in the *HATS Web Application Programmer's Guide*.

Once you have selected your Credential Mapper plug-in, the details, such as class, name, description and author, are filled in the **Details** section. The **Initialization** section displays a set of parameters configured for the plug-in you selected. By clicking the **Add** button, you can specify additional parameters for your plug-in. You can also select **Remove** to remove selected parameters. Only parameters which are not required can be removed.

## Credential Mapper selection

Table 9 describes which Credential Mapper requests will be handled by this plug-in.

*Table 9. Authentication types and descriptions*

| Authentication type | Description |
|---|---|
| AuthType_All | Identifies the credentials to be used for all authentication types |
| AuthType_3270 | Identifies the credentials to be used with 3270 emulation |
| AuthType_5250 | Identifies the credentials to be used with 5250 emulation |
| AuthType_VTHost | Identifies the credentials to be used with VT emulation |

If multiple credential mappers are defined, you can use a host mask as a secondary selection criteria to identify the most appropriate credential mapper to use. Table 10 shows examples of how to specify a host mask.

*Table 10. Host masks and values matched*

| Host mask | Value matched |
|---|---|
| *.raleigh.ibm.com | Matches all addresses that end with .raleigh.ibm.com |
| ralvm* | Matches all addresses that start with ralvm |
| * | Matches all |

*Table 10. Host masks and values matched (continued)*

| Host mask | Value matched |
|---|---|
| *xyz* | Matches any host address that contains xyz |

## Initialization parameters

For solutions that use z/OS and DCAS, add the DCAS plug-in parameters. Adding these parameters allows the CMP to map a network ID to a host ID and then get a passticket from the DCAS application running on the host. A passticket is a credential that is similar to a password, however a passticket expires after a certain amount of time and is used only one time. DCAS requires a Security Access Facility (SAF)-compliant server product, such as an IBM Resource Access Control Facility (RACF) security server, that supports passticket generation.

**Note:** Java Secure Socket Extension (JSSE) is used by the HATS DCAS/RACF/JDBC and Certificate-based DCAS/RACF credential mapper plug-ins for secure connections to the DCAS server. For information about migrating this support from versions before HATS V7 see "Web Express Logon (WEL)" on page 21.

### DCAS parameters for DCAS/RACF/JDBC Credential Mapper plug-in

**Required DCAS parameters:** Some combination of the following parameters is required to allow the credential mapper plug-in to connect to the DCAS server securely:

**CMPI_DCAS_TRUSTSTORE**
This parameter contains the name of the keystore file to be used to look up the HATS DCAS client certificate and the DCAS server certificate. If CMPI_DCAS_USE_DEFAULT_TRUSTSTORE is set to **true**, the JSSE default keystore file is used instead of the keystore file specified by this parameter.

**CMPI_DCAS_TRUSTSTORE_PASSWORD**
This parameter contains the password of the keystore file specified by CMPI_DCAS_TRUSTSTORE.

**CMPI_DCAS_TRUSTSTORE_TYPE**
This parameter contains the type of the keystore file specified by CMPI_DCAS_TRUSTSTORE. Valid values are **pkcs12**, **jceks**, and **jks**.

**CMPI_DCAS_USE_DEFAULT_TRUSTSTORE**
This parameter indicates whether the JSSE default keystore file should be used to look up the HATS DCAS client certificate and the DCAS server certificate. The default is **false**. If specified as **true**, this keystore file is used instead of the keystore file specified by the CMPI_DCAS_TRUSTSTORE parameter.

**Note:** The search order to locate the JSSE default keystore file is:

- the location specified by the javax.net.ssl.trustStore system property, then
- <java-home>/lib/security/jssecacerts, then
- <java-home>/lib/security/cacerts

The following parameters are designed to work with your JDBC database credential mapper. Using this type of network-accessible database provides you with a flexible and secure means of associating user's network IDs to their host

IDs. By storing all the relevant access information, you can configure access to an existing database or point to a newly created database. The level of security for the database varies according to database vendor.

**CMPI_DCAS_DB_ADDRESS**
This is a URL string that provides the address of the database.

**CMPI_DCAS_DB_NET_DRIVER**
This string contains the name of the class that acts as the network database driver. An example of this string is COM.ibm.db2.jdbc.net.DB2Driver. The location of this class is assumed to be in the existing class path.

**CMPI_DCAS_DB_USERID**
This is the ID of the user account to use when accessing the database.

**CMPI_DCAS_DB_CASE_SENSITIVE**
This parameter specifies whether the DCAS plug-in converts the application ID and network ID of the user to lowercase characters and then uses the lcase() method to make SQL queries to the HCM database. This parameter should be set to true when using SQL applications that do not support the lcase() method.

**CMPI_DCAS_DB_PASSWORD**
This is the password of the user account to use when accessing the database.

**CMPI_DCAS_DB_TABLE**
This entry identifies the table to use for the needed query.

The following four parameter values should match the column names in your credential mapper database and should clearly indicate the contents of the columns. With some databases, such as IBM DB2, the four column headings in the database must be in all upper case, for example, NETWORKID, HOSTADDRESS, APPLICATIONID, and HOSTID.

**CMPI_DCAS_DB_NETID_COL_NAME**
This entry identifies the name of the column that contains the network ID value (NETWORKID).

**CMPI_DCAS_DB_HOSTADDR_COL_NAME**
This entry identifies the name of the column that contains the host address value (HOSTADDRESS).

**CMPI_DCAS_DB_HOSTAPP_COL_NAME**
This entry identifies the name of the column that contains the host application value (APPLICATIONID).

**Note:** Application ID is only used for 3270 host types.

**CMPI_DCAS_DB_HOSTID_COL_NAME**
This entry identifies the name of the column that contains the user's host identification value (HOSTID).

Based on the information provided by the parameters above, you can make an SQL query of the database to get the host ID. This query uses the network ID, the host address, and the host application as keys for the query. The result is identified in the Host Identification column. Assuming that the query is successful, a call is made to the DCAS server to request the passticket.

**Optional DCAS parameters**: The following DCAS parameters are optional:

**CMPI_DCAS_DB_PRESERVE_WHITESPACE**
This parameter indicates whether to trim white space from the credential request parameters or not. If true, the white space is not trimmed. The default is false.

**CMPI_DCAS_HOST_ADDRESS**
The default DCAS host address is determined based on the destination host specified for the HATS connection.

**CMPI_DCAS_HOST_PORT**
The DCAS host address is determined based on the destination host specified in the request. The default port address of 8990 is used, but you may override it using this parameter.

**CMPI_DCAS_NO_FIPS**
If set to true, this parameter indicates that the FIPS security provider should not be used. The default security provider will be used instead. The default is **false**.

**CMPI_DCAS_REQUEST_TIMEOUT**
This parameter specifies the passticket request timeout in milliseconds. It should be less than the macro timeout value. The default is **50000**.

**CMPI_DCAS_TRACE_LEVEL**
This parameter specifies the trace level for the DCAS plug-in. The trace messages are logged to the HATS trace file. Trace level values include the following settings:

- **0 = None**: No tracing. This is the default.
- **1 = Minimum**: Trace APIs and parameters, return values, and errors.
- **2 = Normal**: Trace Minimum plus internal APIs and parameters and informational messages.
- **3 = Maximum**: Trace Normal plus Java exceptions.

**CMPI_DCAS_USE_NETID_AS_HOSTID**
If set to true, the database lookup for the host ID is skipped. Use this if the network ID is also the RACF ID. The default is **false**.

**CMPI_DCAS_VERIFY_SERVER_NAME**
This parameter indicates if the server host name in the certificate must be verified in addition to the certificate validation. The default is **false**.

## DCAS parameters for Certificate-based DCAS/RACF Credential Mapper plug-in

**Required DCAS parameters:** Some combination of the following parameters is required to allow the credential mapper plug-in to connect to the DCAS server securely:

**CMPI_DCAS_TRUSTSTORE**
This parameter contains the name of the keystore file to be used to look up the HATS DCAS client certificate and the DCAS server certificate. If CMPI_DCAS_USE_DEFAULT_TRUSTSTORE is set to **true**, the JSSE default keystore file is used instead of the keystore file specified by this parameter.

**CMPI_DCAS_TRUSTSTORE_PASSWORD**
This parameter contains the password of the keystore file specified by CMPI_DCAS_TRUSTSTORE.

**CMPI_DCAS_TRUSTSTORE_TYPE**
This parameter contains the type of the keystore file specified by CMPI_DCAS_TRUSTSTORE. Valid values are **pkcs12**, **jceks**, and **jks**.

**CMPI_DCAS_USE_DEFAULT_TRUSTSTORE**
This parameter indicates whether the JSSE default keystore file should be used to look up the HATS DCAS client certificate and the DCAS server certificate. The default is **false**. If specified as **true**, this keystore file is used instead of the keystore file specified by the CMPI_DCAS_TRUSTSTORE parameter.

**Note:** The search order to locate the JSSE default keystore file is:

- the location specified by the javax.net.ssl.trustStore system property, then
- <java-home>/lib/security/jssecacerts, then
- <java-home>/lib/security/cacerts

**Optional DCAS parameters**: The following DCAS parameters are optional:

**CMPI_DCAS_HOST_ADDRESS**
The default DCAS host address is the destination host specified for the HATS connection.

**CMPI_DCAS_HOST_PORT**
The default port address of 8990 is used, but you may override it using this parameter.

**CMPI_DCAS_NO_FIPS**
If set to true, this parameter indicates that the FIPS security provider should not be used. The default security provider will be used instead. The default is **false**.

**CMPI_DCAS_REQUEST_TIMEOUT**
This parameter specifies the passticket request timeout in milliseconds. It should be less than the macro timeout value. The default is 50000.

**CMPI_DCAS_TRACE_LEVEL**
This parameter specifies the trace level for the DCAS plug-in. The trace messages are logged to the HATS trace file. Trace level values include the following settings:

- **0 = None**: No tracing. This is the default.
- **1 = Minimum**: Trace APIs and parameters, return values, and errors.
- **2 = Normal**: Trace Minimum plus internal APIs and parameters and informational messages.
- **3 = Maximum**: Trace Normal plus Java exceptions.

**CMPI_DCAS_VERIFY_SERVER_NAME**
This parameter indicates if the server host name in the certificate must be verified in addition to the certificate validation. The default is false.

## Vault parameters for JDBC Vault Credential Mapper plug-in

Add Vault parameters for CMPIVaultPlugin. For environments that use JDBC-based Vault host security, add the Vault plug-in parameters. This model is identical to the database mechanism used to associate network IDs and host IDs in the DCAS passticket environment. The only difference is that Vault-style authentication requires the CMPI_VAULT_DB_HOSTPW parameter

**Required Vault parameters**: The following Vault parameters are required:

**CMPI_VAULT_DB_ADDRESS**
This is a URL string that provides the address of the database. An example of this string is `jdbc:db2://dtagw:6789/CMTEST`.

**CMPI_VAULT_DB_NET_DRIVER**
This string contains the name of the class that acts as the network database driver. An example of this string is `COM.ibm.db2.jdbc.net.DB2Driver`. The location of this class is assumed to be in the existing class path.

**CMPI_VAULT_DB_USERID**
This is the ID of the user account to use when accessing the database.

**CMPI_VAULT_DB_CASE_SENSITIVE**
This parameter specifies whether the Vault plug-in converts the application ID and network ID of the user to lowercase characters and then uses the lcase() method to make SQL queries to the HCM database. This parameter should be set to true when using SQL applications that do not support the lcase() method.

**CMPI_VAULT_DB_PASSWORD**
This is the password of the user account to use when accessing the database.

**CMPI_VAULT_DB_TABLE**
This entry identifies the table to use for the needed query.

The following five parameter values exactly match the column names in your credential mapper database.

**CMPI_VAULT_DB_NETID_COL_NAME**
This entry identifies the name of the column that contains the network ID value (NETWORKID).

**CMPI_VAULT_DB_HOSTADDR_COL_NAME**
This entry identifies the name of the column that contains the host address value (HOSTADDRESS).

**CMPI_VAULT_DB_HOSTAPP_COL_NAME**
This entry identifies the name of the column that contains the host application value (APPLICATIONID).

**Note:** Application ID is only used for 3270 host types.

**CMPI_VAULT_DB_HOSTID_COL_NAME**
This entry identifies the name of the column that contains the user's host identification value (HOSTID).

**CMPI_VAULT_DB_HOSTPW_COL_NAME**
This entry identifies the name of the column that contains the user's host password (PASSWORD).

Based on the information provided by the parameters above, you can make an SQL query of the database to get the host ID. This query uses the network ID, the host address, and the host application as keys for the query. The result is identified in the Host Identification column. Assuming that the query is successful, the user ID and password are returned.

**Optional Vault parameters**: The following Vault parameters are optional:

**CMPI_VAULT_TRACE_LEVEL**
This parameter specifies the trace level for the Vault plug-in. The trace messages are logged to the HATS trace file. Trace level values include the following settings:

- **0 = None**: No tracing. This is the default.

- **1 = Minimum**: Trace APIs and parameters, return values, and errors.
- **2 = Normal**: Trace Minimum plus internal APIs and parameters and informational messages.
- **3 = Maximum**: Trace Normal plus Java exceptions.

**CMPI_VAULT_DB_PRESERVE_WHITESPACE**
This parameter indicates whether to trim white spaces from the credential request parameters or not. If true, the white spaces are not trimmed. The default is false.

### Vault parameters for the WebSphere Portal Credential Vault Credential mapper

**Note:** This initialization parameter will only appear if you are working with a HATS portlet project.

**SLOT_ID**
This is the vault slot ID that will retrieve a passive user-password credential from a vault slot.

**SLOT_TYPE**
This parameter indicates the type of credential slot you want to access. Valid values are:

- **0 = Private Slot**
- **1 = Shared Slot**
- **2 = Administrative Slot**: This is the default value.
- **3 = System Slot**

For descriptions of the different types of slots, refer to the WebSphere Portal product information Web site: http://www.ibm.com/developerworks/websphere/zones/portal/proddoc.html.

## Create SSL keystore file (DCAS only)

In order to communicate with a DCAS server, an SSL connection must be established using client authentication. This requires you to specify a keystore file. The supported keystore file types are PKCS12, JKS, or JCEKS (PKCS12 is not supported on Solaris). To create a keystore file to specify in the CMPI_DCAS_TRUSTSTORE parameter, use the Certificate Management tool (also known as the IBM Key Management tool). This keystore file must contain the HATS DCAS client's certificate and the DCAS server's certificate (public key) information.

**Notes:**

1. If you set the CMPI_DCAS_USE_DEFAULT_TRUSTSTORE parameter to **true**, the JSSE default keystore file is used instead of the keystore file specified by the CMPI_DCAS_TRUSTSTORE parameter, and must contain the HATS DCAS client's certificate and the DCAS server's certificate (public key) information.
2. The HATS DCAS client's certificate must also be added/imported to the DCAS server's keystore file for SSL client authentication.

If you already have an older certificate that was created using the IBM Key Management tool, you can import it. Personal server certificates that were created with an old system cannot be exported from the old and imported into the new. There is however a way in which you can do this:

1. Import the existing .kdb file into a new keystore file (PKCS12, JKS, or JCEKS).

2. Export the certificate (such as, the DCAS personal server certificate) to a .p12 format certificate.
3. Import the certificate (.p12 format) into a new keystore file (PKCS12, JKS, or JCEKS).

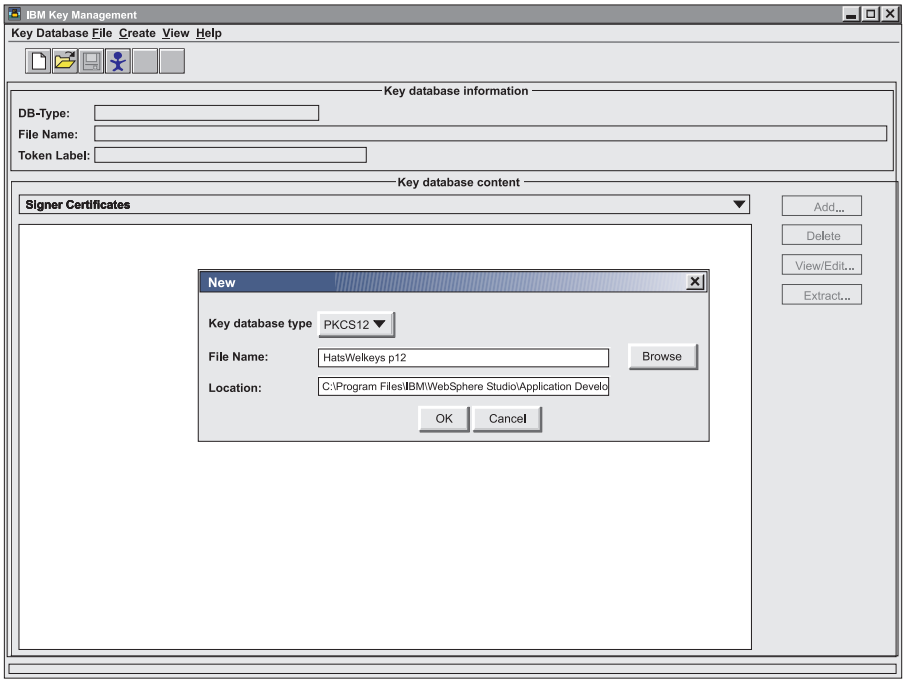



*Figure 78. HATS Certificate Management*

To create a new keystore file named HatsWelkeys.p12 that will be specified in the CMPI_DCAS_TRUSTSTORE parameter, take the following steps:

**Note:** These instructions show how to create a PKCS12 keystore file. If the target platform for your HATS application is Solaris, instead of using Key database type of **PKCS12** below, use either **JCEKS** or **JKS** instead.

1. Click **Start > All Programs >** *IBM Rational SDP package group* **> IBM Rational HATS 9.7 > Certificate Management** (where *IBM Rational SDP package group* is the name of the Rational SDP package group you have installed).
2. Click **Key Database File** and select **New**. For the Key database type, select **PKCS12**. Enter File Name and Location or accept default values.
3. Click **OK**.
4. Enter a password, confirm it, and click **OK**.
5. Add the DCAS server's certificate to the keystore file. Be sure that the **Key database content** is displaying the **Signer Certificates**. If it is not, select the pull-down menu and change it. Then select **Add**.
   a. Select **Binary DER data** for the **Data type**. If the server certificate is in ASCII format, select **Base64-encoded ASCII data**.
   b. Enter the file name in the **Certificate file name** field.
   c. Enter the path name in the **Location** field.
   d. Click **OK**.
   e. Enter a label for the certificate and click **OK**.
6. Add the DCAS client's certificate to the keystore file.

a. Change the **Key database content** to **Personal Certificates** and click **Export/Import**.
b. On the Export/Import Key panel, select **Import Key**.
c. Select **PKCS12** for the key file type.
d. Enter the client certificate's .p12 file name in the **File Name** field and the path name in the **Location** field.

   **Note:** You may have to browse to the keystore file (.p12/pkcs12) containing the certificate to import and enter the user id and password to open the file. It is best to make sure the keystore file contains only certificates that you want to import. You can also import certificates from a .kdb file. In this case, it will allow individual certificates to be selected.

e. Click **OK** and enter the password to open the source key database.
f. Click **OK**.

7. Exit the Certificate Management GUI.

**Note:** For more information about the Certificate Management tool, see "Using IBM Certificate Management for HATS applications" on page 420.

## Using Kerberos service tickets

You can use Kerberos services tickets to automate sign-on for rich client applications that use 5250 Telnet sessions and run on Windows domain clients. To do so, edit the connection file (the .hco file) for the 5250 Telnet connection in your rich client application. Click the **Security** tab, and select the option to **Use Kerberos services ticket to automate sign-on (Windows domain clients only)**.

This support uses underlying IBM Host On-Demand connection-based automation in conjunction with IBM i Kerberos-based network authentication. For more information, see the section on configuring connection-based automation in the Host On-Demand Knowledge Center at: http://publib.boulder.ibm.com/infocenter/hodhelp/v11r0/index.jsp?topic=/com.ibm.hod.doc/doc/logon/logon14.html

## Java 2 security

Java 2 security provides a policy-based access control mechanism that checks for permission before allowing an application access to certain protected system resources, such as file I/O, sockets, and environment properties.

Java 2 security is supported by the application server. It is disabled by default but is enabled automatically when you configure WebSphere security to enable global security. Although Java 2 security is enabled when you enable WebSphere global security, you can disable it. You can configure Java 2 security and global security independently of each other. Disabling global security does not disable Java 2 security automatically. You need to explicitly disable it.

HATS supports Java 2 security when running on WebSphere Application Server on any supported server software platform. For more on HATS supported server software, see "System Requirements for Host Access Transformation Services" at http://www.ibm.com/support/docview.wss?uid=swg27011794.

HATS does not support Java 2 security in the following environments:

- The Run on Server (ROS) function in HATS Toolkit.
- WebSphere Portal on any platform except Windows.

### Policy file

The HATS Toolkit includes a Java 2 security policy file that gives deployed HATS applications the required permissions to operate when Java 2 security is enabled. Although basic HATS operations have been tested with Java 2 security enabled, you might need to modify the policy file for some HATS applications to operate with Java 2 security enabled. For example, the business logic in your application accesses system resources, or a keyring file or another file exists in a path not already covered by the policy file.

You can find the policy file for a HATS application in the **Navigator** view of the HATS perspective. Expand the EAR project for the HATS application, and open the META-INF folder to find the was.policy file. HATS copies the default policy file into every new project you create. If you want to modify the default policy file, it is located under the installed HATS plug-in in the following directory:

```
<shared_install_directory>\plugins\com.ibm.hats_nnn\predefined\
projects\earProject\META-INF\
```

where *shared_install_directory* is the shared resources directory where you installed the HATS offering using IBM Installation Manager, and *nnn* is the version and build level of HATS.

For more information about Java 2 security and policy files, see the WebSphere Knowledge Center at http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp?topic=/com.ibm.websphere.express.doc/info/exp/ae/tsec_enablejava2sec.html

## Using IBM Certificate Management for HATS applications

Certificate Management, provided by the IBM Key Management tool, allows you to create and manage certificates required for Secure Sockets Layer (SSL) connections.

For HATS, SSL is used between the following:

- A HATS application and the Telnet server for secure connections.
- The Digital Certificate Access Server (DCAS) and the client for Web Express Logon (WEL) in a HATS application to retrieve host credentials.

### Creating a key database file

Before you begin configuring SSL for HATS applications, you must create a key database file, also referred to as a keystore file or truststore file.

To create a new key database file:

1. Click **Start > All Programs >** *IBM Rational SDP package group* **> IBM Rational HATS 9.7> Certificate Management** (where *IBM Rational SDP package group* is the name of the Rational SDP package group you have installed).
2. This launches the IBM Key Management tool.
3. Click **Key Database File > New**.
4. Select **PKCS12**, **JKS**, or **JCEKS** key database file for the key database type.

   **Note:** PKCS12 is required for secure connections between HATS applications and the Telnet server.

5. Enter a file name. HATS does not require a particular file name.
6. Enter a directory name for the Location. HATS does not require a particular directory for creation of the key database file, but the key database file will need to be copied into an Enterprise Archive file (.ear file) in order to deploy an application that uses it.
7. Click **OK**.
8. Enter a password, confirm it, and click **OK**.

To open an existing key database file:

1. Click **Key Database File > Open**.
2. Select the key database type.
3. Click **Browse** to browse for the key database file.
4. Select the key database file and click **Open**.
5. Click **OK**.
6. Enter the current password and click **OK**.

After you have created or opened the key database file, you can:

- Request a certificate from a predefined well-known certificate authority (CA). This procedure requires less configuration because the key database files are pre-configured with the CA signer certificates required to identify the CAs from whom the server certificate is issued. See "Requesting and storing certificates from Certificate Authorities."
- Request a certificate from an unknown certificate authority. This procedure requires more configuration because you must also obtain and store the CA's signer certificate in the key database file. In addition, you must make the signer certificate available to the SSL partners from which you want to obtain host credentials. See "Requesting and storing certificates from Certificate Authorities."
- Create a self-signed certificate. This procedure does not require a certificate authority and can be used immediately after installing the server. However, the self-signed certificate must be made available to the SSL partners. This procedure can also be used for testing until a certificate is obtained from a certificate authority. See "Using a self-signed certificate" on page 423.
- Exchange certificates with SSL partners. In some configurations, certificates in the server's key database file must be made available to any SSL partners, such as the Telnet or DCAS server, to enable SSL communications. See "Exchanging certificates" on page 424.

**Note:** Whenever you change the key database file used by a running HATS application, you must stop and restart the HATS application.

## Requesting and storing certificates from Certificate Authorities

When you create a key database file, it is pre-configured with the CA signer (trusted root) certificates of well-known CAs required to identify the CA from whom the server certificate is issued. CAs whose signer certificates are not predefined in a key database file are referred to as unknown CAs. The following well-known CA signer certificates are automatically stored in a newly created key database file and marked as trusted root certificates.

- Thawte Personal Premium CA
- Thawte Personal Freemail CA
- Thawte Personal Basic CA

- Thawte Premium Server CA
- Thawte Server CA
- RSA secure server CA (also obtained from VeriSign)
- VeriSign class 4 public primary CA
- VeriSign class 3 public primary CA
- VeriSign class 2 public primary CA

The following sections provide an overview of the steps used to request and store certificates from well-known and unknown CAs:

- Creating the certificate request
- Sending the certificate request
- Storing the certificate

## Creating the certificate request

To create the public/private keys and certificate request:

1. On the IBM Key Management window, under **Key database content**, select **Personal Certificate Requests** from the drop-down list.
2. Click **New**.
3. Enter the key label to identify the key and certificate within the database.
4. Select the number corresponding to the key size you want to use. Choosing a larger key size results in stronger security, but requires more processing on the client and the server to establish a connection.
5. Enter the common name (TCP/IP host name) of the server that owns the certificate, for example, `myserver.mycompany.com`.
6. Enter an organization name.
7. Optionally enter an organization unit, and other location information.
8. Choose a country code.
9. Enter a certificate request file name, or use the default file name.

When you click **OK**, your information is processed and several files are produced or updated in the directory where you created the certificate request. If you backup your key database, backup these files also. Do not attempt to edit these files, as the key database file or the certificate request can be corrupted.

## Sending the certificate request

Follow the CA's procedures to submit the certificate request.

The following list provides the URLs of two well-known CAs:

- Thawte: http://www.thawte.com/
- Verisign: http://www.verisign.com/

Depending on the CA you choose, you can either e-mail the certificate request generated by Certificate Management or incorporate the certificate request into the form or file provided by the CA.

While you are waiting for the CA to process your certificate request, you can enable SSL security for testing by creating and storing a self-signed certificate using the procedure described in “Using a self-signed certificate” on page 423.

### Storing the certificate

If you receive the applied-for certificate from an unknown CA, contact the CA to obtain the CA's signer (root) certificate. You must store the unknown CA's signer certificate in the key database file before you store the applied-for certificate. The CA signer certificate is used to validate the applied-for certificate.

To store the unknown CA's signer certificate, make a backup copy of the unknown CA's signer certificate, then perform the following steps:

1. On the IBM Key Management window, under **Key database content**, select **Signer Certificates** from the drop-down list.
2. Click **Add**.
3. Select data type of **BASE64 encoded ASCII data** (armored 64 format).
4. Enter the certificate file name.
5. Enter the location, or path, of the certificate.
6. Click **OK**. The file is marked as "trusted" and is stored.

To store the applied-for certificate received from either a well-known or unknown CA, make a backup copy of the certificate, then perform the following steps:

1. Choose **Personal Certificates** from the drop-down list then click **Receive** to receive the certificate request. The Receive Certificate from a File window appears.
2. The data type must be **BASE64-encoded ASCII data** (armored 64 format).
3. Enter the certificate file name.
4. Enter the location (path name) of the certificate.
5. Click **OK**. The certificate you just stored is displayed as the first item.
6. If you want to view the key information, highlight the certificate and click **View/Edit**.
7. The certificate name should appear under the Personal Certificate drop-down list and the certificate request should disappear from under the Personal Certificate Requests drop-down list.
8. Copy the key database file to the Enterprise Archive (.ear file) for deployment. If the Enterprise Archive is running, stop and restart it.

## Using a self-signed certificate

Use the following procedures to create and use a self-signed certificate:

1. On the IBM Key Management window, under **Key database content**, select **Personal Certificates** from the drop-down list.
2. Click **New Self-signed**.
3. Enter the key label to identify the key and certificate within the database.
4. Select **X509 V3** as the certificate version.
5. Select the number corresponding to the key size you want to use. Choosing a larger key size results in stronger security, but requires more processing on the client and the server to establish a connection.
6. Enter the common name (TCP/IP host name) of the server that owns the certificate, for example, `myserver.mycompany.com`.
7. Enter an organization name.
8. Optionally enter an organization unit, and other location information.
9. Select a country code.
10. Enter the number of days the self-signed certificate is to be valid.

11. Click **OK**.
12. Copy the key database file to the Enterprise Archive (.ear file) for deployment. If the Enterprise Archive is running, stop and restart it.

## Exchanging certificates

In some configurations, certificates must also be made available to SSL partners, for example the DCAS server. If your server uses a certificate from an unknown CA, the unknown CA's signer (root) certificate must be made available to SSL partners. If your server uses a self-signed certificate, a copy of the self-signed certificate must be made available to SSL partners.

To create a certificate file to exchange:

1. Open the key database file. See "Creating a key database file" on page 420.
2. Extract the certificate.
   - If your server uses a certificate issued by an unknown CA:
     a. Under **Key database content**, select **Signer Certificates** from the drop-down list.
     b. Highlight the signer (root) certificate of the CA that issued the certificate for your server.
     c. Click **Extract**.
   - If your server uses a self-signed certificate:
     a. Under **Key database content**, select **Personal Certificates** from the drop-down list.
     b. Highlight the certificate used by your server.
     c. Click **Extract Certificate**.
3. On the Extract Certificate to a File window, choose either **Base64-encoded ASCII data** or **Binary DER data**. **Base64-encoded ASCII data** is usually used if the certificate will be securely transferred through e-mail. The certificate file name and location can be any you choose.
4. Click **OK** to extract the certificate file.
5. Securely transfer the certificate file to the SSL partner, for example the DCAS server, and add the certificate to the its key database file.

## Certificate Management tool considerations

### DOS window

When the Certificate Management tool is invoked, a DOS window appears and is displayed the entire time the tool is running. The window is finally closed once you exit from the tool. The reason why the DOS window is displayed is because the default run property for the Certification Management tool is **Normal Window**.

To minimize the DOS window when the tool is invoked, take the following steps:

1. Go to **Start > All Programs >** *IBM Rational SDP package group* **> IBM Rational HATS 9.7** (where *IBM Rational SDP package group* is the name of the Rational SDP package group you have installed.
2. Right click **Certificate Management** and select **Properties**.
3. Configure the Run property to be **Minimized**.
4. Click **OK**.

## Run as administrator

On Windows Vista, the Certificate Management tool does not start if you attempt to use the **Run as administrator** option to start the program. This occurs in two ways:

1. You right-click on the Certificate Management icon, and select **Run as administrator**.
2. You specify **Run as administrator** on the advanced settings for the Certificate Management icon properties, and double-click the icon to run the program.

You can run Certificate Management without using the **Run as administrator** option.

## Using IBM CMS keystore files

To use an IBM CMS keystore file, you must register the appropriate provider class by adding it to the provider list. To do this, perform the following steps:

1. Edit the provider list located at:

   `<RationalSDP_install_directory>\jdk\jre\lib\security\java.security`

   where *RationalSDP_install_directory* is the directory where you installed the Rational SDP offering using IBM Installation Manager.
2. Add the following statement to the list of providers:

   `security.provider.<n>=com.ibm.security.cmskeystore.CMSProvider`

   where <*n*> is the preference order. For example,

   `security.provider.10=com.ibm.security.cmskeystore.CMSProvider`

# Chapter 22. Language support

The HATS user interface and context-sensitive help are provided in these languages:

- Arabic
- Brazilian Portuguese
- Chinese (Simplified)
- Chinese (Traditional)
- Czech
- English
- French
- German
- Hungarian
- Italian
- Japanese
- Korean
- Polish
- Russian
- Spanish
- Turkish

All the languages are installed in a single product image. National language support is operating-system dependent, therefore the appropriate font and keyboard support for the language you want to use must be installed in the operating system. For example, if you want to use French as the host-session language but do not have the French font and keyboard support installed, you may not be able to display the correct characters.

**Notes:**

1. The **HATS Toolkit** start menu item is located in the IBM Rational HATS HATS V9.7 group under the Rational SDP package group in which you have installed HATS.
2. When you use the **HATS Toolkit** start menu item, the HATS Toolkit starts in the language installed for Rational SDP, assuming the HATS Toolkit is translated in the same language.
3. For more information about starting HATS using Arabic, see "HATS Toolkit screen orientation" on page 452.

## Language codes

Language codes are used in different parts of the product to determine which language to present to the user. Refer to Table 11 for the language and language code.

*Table 11. HATS language codes*

| Language | Language Code |
|---|---|
| Arabic | ar |

*Table 11. HATS language codes (continued)*

| Language | Language Code |
|---|---|
| Czech | cs |
| German | de |
| English | en |
| Spanish | es |
| French | fr |
| Hungarian | hu |
| Italian | it |
| Japanese | ja |
| Korean | ko |
| Polish | pl |
| Brazilian Portuguese | pt_BR |
| Russian | ru |
| Turkish | tr |
| Chinese (Simplified) | zh |
| Chinese (Traditional) | zh_TW |

## Code pages

HATS supports the following code pages. You can select the code page for each HATS project when you create the project, and you can modify it later in the project editor.

*Table 12. Code pages*

| Code page | Location or usage |
|---|---|
| 037 | Belgium<br>Brazil<br>Canada<br>Netherlands<br>Portugal<br>United States |
| 273 | Austria<br>Germany |
| 274 | Belgium (Old) |
| 275 | Brazil (Old) |
| 277 | Denmark<br>Norway |
| 278 | Finland<br>Sweden |
| 280 | Italy |
| 284 | Spain<br>Latin-America (Spanish) |
| 285 | United Kingdom |
| 297 | France |
| 420 | Arabic Speaking |

*Table 12. Code pages (continued)*

| Code page | Location or usage |
|---|---|
| 424 | Hebrew (New Code) |
| 500 | Multilingual |
| 803 | Hebrew (Old Code) |
| 838 | Thai |
| 870 | Bosnia/Herzegovina<br>Croatia<br>Czech Republic<br>Hungary<br>Poland<br>Romania<br>Slovakia<br>Slovenia |
| 871 | Iceland |
| 875 | Greece |
| 924 | Multilingual ISO Euro |
| 930 | Japanese (Katakana)<br>Japan (Katakana Extended) |
| 933 | Korea |
| 937 | Taiwan (Traditional Chinese)<br>Note: To support accented characters see<br>"Using accented characters for code page 937" on page 431. |
| 939 | Japanese (Latin Extended) |
| 1025 | Belarus<br>Bulgaria<br>FYR Macedonia<br>Russia<br>Serbia/Montenegro (Cyrillic) |
| 1026 | Turkey |
| 1047 | Open Edition |
| 1112 | Latvia<br>Lithuania |
| 1122 | Estonia |
| 1123 | Ukraine |
| 1137 | Hindi |
| 1140 | Belgium Euro<br>Brazil Euro<br>Canada Euro<br>Netherlands Euro<br>Portugal Euro<br>United States Euro |
| 1141 | Austria Euro<br>Germany Euro |
| 1142 | Denmark Euro<br>Norway Euro |
| 1143 | Finland Euro<br>Sweden Euro |
| 1144 | Italy Euro |

*Table 12. Code pages (continued)*

| Code page | Location or usage |
|---|---|
| 1145 | Latin-America Euro (Spanish)<br>Spain Euro |
| 1146 | United Kingdom Euro |
| 1147 | France Euro |
| 1148 | Multilingual Euro |
| 1149 | Iceland Euro |
| 1153 | Bosnia/Herzegovina Euro<br>Croatia Euro<br>Czech Republic Euro<br>Hungary Euro<br>Poland Euro<br>Romania Euro<br>Slovakia Euro<br>Slovenia Euro |
| 1154 | Belarus Euro<br>Bulgaria Euro<br>FYR Macedonia Euro<br>Russia Euro<br>Serbia/Montenegro (Cyrillic) Euro |
| 1155 | Turkey Euro |
| 1156 | Latvia Euro<br>Lithuania Euro |
| 1157 | Estonia Euro |
| 1158 | Ukraine Euro |
| 1160 | Thai Euro |
| 1166 | Kazakhstan Euro |
| 1364 | Korea Euro |
| 1371 | Taiwan (Traditional Chinese) Euro |
| 1388 | PRC (Simplified Chinese Extended; GB18030) |
| 1390 | Japanese (Katakana Unicode Extended) |
| 1399 | Japanese (Latin Unicode Extended) |

## Encoding settings

The character set used in HATS JSPs, such as HATS transformations and templates, must match the encoding configured in the user's browser, as well as the encoding specified in the application server. HATS applications create JSPs in all locales using the UTF-8 character set.

WebSphere Application Server has different default encoding settings for different locales. Some double-byte characters and bidirectional language data input might be decoded incorrectly if the encoding setting for that locale is not UTF-8. You must configure the application server on which your HATS application runs to use the same encoding used by the JSP in your HATS application. To configure UTF-8, follow these steps:

1. Open the WebSphere Application Server administrative console. If using the local test environment in Rational SDP, in the Servers view, right-click on the server and select **Administration > Run administrative console**.
2. Log into the administrative console, open the **Servers** tree in the left navigation panel, and select **Application servers**.
3. Select the server name to modify.
4. In the right panel, open the **Java and Process Management** tree, select **Process Definition**, and on the panel that opens next, select **Java Virtual Machine** .
5. In the **Generic JVM arguments** section, enter `-Dclient.encoding.override=UTF-8`.
6. Click the **OK** button.
7. Select the **Save** links in the Messages blocks.
8. Log off, and restart WebSphere Application Server.

## 5250 Unicode support

The Enable Unicode Data Stream parameter, allows you to configure a connection parameter for 5250 hosts. The **unicodeDataStreamEnabled** parameter is a parameter on the **Advanced** tab of the Connection Editor. To add the parameter, go to the **Configure optional, advanced connections settings** and click the **Add** button. Select the **unicodeDataStreamEnabled** parameter from the pull down **Name** menu and enter the **Value**. Click **OK**.

## Using accented characters for code page 937

If you use the Traditional Chinese code page 937 for a 5250 connection and want to support the use of accented characters, you must add the advanced connection settings parameter, **useAccentedCharacters**, and set its value to **true** on the **Advanced** tab of the Connection editor. See "Advanced" on page 133 for how to add Host On-Demand session parameters. In addition, because the CCSID 937 does not support accented characters, to use this support you must ensure that the CCSID is 65535 on the host. This feature is only supported for 5250 connections.

## Using code page 1388 (GB18030)

Following is a list of considerations when using the PRC (Simplified Chinese Extended; GB18030) code page 1388:

- If you add GB18030 characters to a template, you can view them properly in the **Source** tab, but they might not appear correctly in the Design view of the template editor.
- When working with widgets in screen transformations, GB18030 data might not display in the Rich Page Editor. If not, perform the following steps:
  1. In the Insert Host Component wizard on the Rendering Options panel, click the **Widget Settings** button.
  2. On the Settings panel, clear the **Use project defaults** box.
  3. Next to the Style field, click the **CSS Style properties** button.
  4. Add the GB18030 four-byte supported fonts such as **SimSun-18030** to the font family list.
  5. Click **OK** and **Finish**, as appropriate, to close the wizard.

## Host code mapping for code pages 1390 and 1399

There have been host-to-PC code mapping inconsistencies between IBM Personal Communications and Host On-Demand products on a number of DBCS code points when Japanese host code page 1390 or 1399 is used. In Host On-Demand, the **UseHodCDRA1399** parameter controls the Unicode remap to correct the inconsistencies. The following table summarizes the mapping for both true and false settings of the **UseHodCDRA1399** parameter:

*Table 13. Summary of Unicode mapping*

| EBCDIC code point | Unicode code point (UseHodCDRA1399=true) | Unicode code point (UseHodCDRA1399=false) |
|---|---|---|
| 0x4260 | U+FF0D (Fullwidth hyphen-minus) | U+2212 (Minus sign) |
| 0xE9F3 | U+2212 (Minus sign) | U+FF0D (Fullwidth hyphen-minus) |
| 0x43A1 | U+FF5E (Fullwidth Tilde) | U+301C (Wave Dash) |
| 0xE9F4 | U+301C (Wave Dash) | U+FF5E (Fullwidth Tilde) |
| 0x447C | U+2225 (Parallel To) | U+2016 (Double Vertical Line) |
| 0xDFE5 | U+2016 (Double Vertical Line) | U+2225 (Parallel To) |
| 0x444A | U+2015 (Horizontal Bar) | U+2014 (EM Dash) |
| 0xDDB7 | U+2014 (EM Dash) | U+2015 (Horizontal Bar) |
| 0x426A | U+FFE4 (Fullwidth Broken Bar) | U+00A6 (Broken Bar) |
| 0xE9F5 | U+00A6 (Broken Bar) | U+FFE4 (Fullwidth Broken Bar) |

HATS uses the Host On-Demand **UseHodCDRA1399** parameter for the same purpose. The HATS default for the **UseHodCDRA1399** parameter is **false**. To set the parameter to **true**, edit the connection in your HATS project and perform the following steps:

1. Click the **Advanced** tab.
2. Click **Add**.
3. From the drop-down list of parameter names, select **UseHodCDRA1399**.
4. Set the parameter value to **true**.
5. Click **OK**.

## JIS2004 support

JIS2004, which is a new Japanese national standard coded character set, adds three types of new characters:

1. Normal new characters that were not available before JIS2004 support.
2. Surrogate characters, which are two linked Unicode code points that represent a single character. The two Unicode code points are meaningless if they are separated.
3. Combining characters, which are two Unicode code points combined as one character. The two individual Unicode code points that combine to form a combining character can be displayed separately also.

### JIS2004 support considerations and limitations

When using JIS2004 in your applications, consider the following items:

1. JIS2004 is the default Japanese font support with Windows Vista SP1. If you want to continue using JIS90 on the Vista operating system, you can install

Microsoft service pack KB927490, which can be downloaded and applied through Windows Update as additional software.

2. To display JIS2004 characters in Windows XP, you must install Microsoft patch KB927489. Even when the patch is installed, you will not be able to insert some JIS2004 characters, such as the surrogate characters, since IME is not supported.
3. The default font setting in HATS does not display JIS2004 characters correctly without some changes to the font settings.

   For HATS Web applications, enable the style setting in each HATS widget, such as the Field Widget, to add the following fonts:

   - MS MinCho
   - MS Gothic
   - Meiryo

   For HATS rich client applications, set the font setting in each HATS widget to the MS Mincho font.
4. When using Internet Explorer V7 and higher, new JIS2004 characters can be supported correctly by setting the fonts as described in the previous item.
5. When using Internet Explorer V6, or when using the preview function in the HATS Toolkit running on Windows XP, some new JIS2004 characters are not displayed correctly at all.
6. Surrogate character support:
   a. In the HATS terminal, when entering JIS2004 characters using IME, the candidate window displays surrogate characters as black squares. However, when the characters are sent to HATS, the HATS terminal displays the surrogate characters correctly.
   b. JIS2004 surrogate characters are not supported for Host On-Demand custom tables.
7. Combining character support:
   a. Windows applications, such as Notepad, and Firefox Version 3 correctly handle JIS2004 combining characters as one character. However, when running HATS rich client applications at runtime, or running HATS Web applications in Internet Explorer, the combining character is handled as two separate characters instead of one character in input fields. The cursor can be placed between the two combining characters.
   b. When running HATS Web applications at runtime in Internet Explorer, if you enter combining characters in the input field, and the **Show unprotected Shift Out/Shift In characters as spaces** setting in the DBCS section of the Rendering tab is enabled, combining characters become white squares. You can correct the white squares by disabling the setting, or using the Firefox browser.
   c. The HATS PDF print function cannot display combining characters correctly.

## JIS2004 support for PDT printing and Print-to-File for 3270E sessions

To print JIS2004 characters:

- For PDT printing, specify a pdfpdt file, such as ibm5577.pdfpdt, and ensure that the parameter KANJI_CODE=SHIFT_JIS is specified.
- For Print-to-File, specify the file encoding using the parameter KANJI_CODE:
  – To save the output as a PC file, dumping the output as a PC file with a font image, set the file encoding to Shift JIS by specifying KANJI_CODE=SHIFT_JIS in the pdfpdt file. The print output is saved as a native PC file that can be used to output the file content to a physical printer.

– To save the output as an Unicode file, dumping the output as a Unicode file with code points of surrogate pairs, set the file encoding to Unicode by specifying KANJI_CODE=UNICODE in the pdfpdt file. The print output is a UCS-2 file that can be used for further data processing or archiving purposes.

The following table shows the results of specifying different values for the KANJI_CODE session parameter with different pdfpdt files when printing data with JIS2004 characters to an output file (Print-to-File):

*Table 14. KANJI_CODE session parameter summary*

| **Session parameter in pdfpdt file** | **ASCII text mode (basic_dbcs.pdf)** | **Other supported printers (esc_*.pdf, ibm*.pdf, lips*.pdf, etc.)** |
|---|---|---|
| KANJI_CODE=SHIFT_JIS | Not applicable since surrogate pairs can not be stored in a native PC file. | The output is saved to a native PC file and font images (binary data) for JIS2004 characters and UDA characters are stored in this file. |
| KANJI_CODE=UNICODE | The output is saved to a Unicode file; surrogate pairs and UDA characters are stored in UCS-2. | Not applicable since a PC printer does not accept Unicode data. |

In addition, you should also configure the session parameters printerMimeType and printSaveAsExtension properly when printing to a file:

*Table 15. Print-to-File session parameter summary*

| **Setting** | **Session parameter in pdfpdt file** | **Session parameter** *printerMimeType* **in HATS** | **Session parameter** *printSaveAsExtension* **in HATS** | **Results** |
|---|---|---|---|---|
| Setting for further printing | KANJI_CODE=SHIFT_JIS | application/octet-stream | .bin | A native PC file with binary data. |
| Setting for ASCII text mode | KANJI_CODE=UNICODE | text/plain | .txt | A UCS-2 file. |

For more information see “Defining print support for your project” on page 353 and “Printing” on page 135.

## Disabling JIS2004 support for code pages 1390 and 1399

To disable JIS2004 support for a connection using code pages 1390 and 1399, perform the following steps:

1. Expand your project in the HATS Projects view.
2. Expand the Connections folder
3. Double-click on the connection to open the connection editor.
4. Click the **Advanced** tab.
5. Under **Configure optional, advanced connection settings**, click **Add**.
6. On the Add Parameter panel, enter the following settings.
   a. In the Name field enter `enableJIS2004`.
   b. In the Value field enter `false`.
   c. Click **OK**.

**Note:** You must disable JIS2004 support to enable viewing certain PDFs files created by HATS print support. For more information, see "For 3270E connections" on page 353.

## Remapping keyboard and display characters

HATS support for Host On-Demand custom tables allows you to remap the user's keyboard and display characters by customizing the host to PC code page conversion tables. A custom table file can contain any number of tables, in any combination of keyboard and display tables. Keyboard tables remap the user's keyboard characters as input customized for a host application. Display tables remap host display characters into screen output customized for the user.

**Note:** This remapping has no relation to HATS keyboard mapping. This remapping deals with character remapping, for example, `a`, `b`, and `c`. Whereas HATS keyboard mapping deals with control key mapping, for example, `Clear`, `Enter`, and `F1`. For more information about HATS keyboard mapping see Chapter 16, "Enabling keyboard support," on page 359.

A sample custom table file, CustomTableExample.txt, is located in the *<RationalSDP_install_directory>*\hats\customtables directory. The file format is similar to that of Java properties files. Lines that start with # are comments, and variables shown as n and N are hexadecimal values. The following example shows the format of the sample custom table file.

```
##### Start of file
# Map SBCS Local nn to SBCS EBCDIC NN in keyboard table
sbcs.keyboard.0xnn=0xNN
# Map DBCS Unicode nnnn to DBCS EBCDIC NNNN in keyboard table
dbcs.keyboard.0xnnnn=0xNNNN
...
# Map SBCS EBCDIC NN to SBCS local nn in display table
sbcs.display.0xNN=0xnn
# Map DBCS EBCDIC NNNN to DBCS Unicode nnnn in display table
dbcs.display.0xNNNN=0xnnnn
...
##### End of file
```

To enable the use of custom tables in your project:

1. Create a custom table file using the sample as an example and copy it into a folder named, **customtables**, in the root of the .ear project of your HATS Web application or in the root of the HATS rich client Runtime Extension project for your HATS rich client project. If the **customtables** folder does not exist, create one.
2. Open your HATS project.
3. Edit the connection by double-clicking the name in the Connections directory.
4. Click the **Advanced** tab.
5. Click **Add**.
6. In the Add Parameter dialog, select **CustomTable** from the drop-down list for **Name** and enter the name of your custom table file in the **Value** field.
7. Click **OK** to exit the Add Parameter dialog.
8. Save your changes to the connection settings.

## Priority of character replacement

There are several ways to replace characters in HATS. Character replacement by these functions occurs in the following order:

1. Custom table support
2. UseHodCDRA1399 parameter for DBCS only
3. User Defined Characters (UDCs) for DBCS only
4. Text replacement

For example, if you use custom table support to change a display character, and text replacement is defined to change the same character, that character does not exist when text replacement occurs.

# Chapter 23. Bidirectional application support

This chapter explains how to use the functions provided by HATS for developing applications in bidirectional languages. Generally its contents apply to both Hebrew and Arabic application developers. The functions that are specific to Arabic users are described separately.

HATS provides these functions to support bidirectional languages:

- Control over the orientation of the HATS Toolkit development environment.
- Support of bidirectional code pages.
- Correct bidirectional text processing and symmetric swapping.
- Support of left-to-right and right-to-left screen customization.
- Support of Model 1 and JSF Web pages, bottom-up traditional (WSDL-based) Web services, and RESTful Web services generated from Integration Objects.
- Support for standard portlets including support for portlet communication between standard portlets.
- A **Screen reverse** button to toggle between left-to-right (LTR) and right-to-left (RTL) screen orientation.
- Different levels of user control over screen orientation.
- Control over the orientation of host components, widgets, and text, which may be opposite to general screen orientation.
- Optional recognition of both right-to-left and left-to-right components in default rendering, including optional recognition of bidirectional components in advanced rendering.
- Control over the orientation of macro prompts and extracts.
- Mixed bidirectional text may be used in widget settings.
- Visual Input Field support on various browsers and platforms in Web projects and Visual Text support in rich client projects.
- Bidirectional text in global variables and text replacement in screens with left-to-right and right-to-left orientation.
- Right-to-left printing support in 3270 sessions.
- Support for VT logical and visual bidirectional sessions including correct insertion and extraction.
- Global rules using bidirectional text.
- Bidirectional support of SWT Terminal.
- Bidirectional support of spreadsheets.
- Full bidirectional support of rich client projects.

This chapter explains all these features.

**Note:** There is no bidirectional support for Dojo transformations and Light pen (attention) and Light pen (selection) host components.

## Software environment

Following are requirements for bidirectional application support:

- Supported browsers for bidirectional applications are:
  - Firefox V3.4, or later
  - Internet Explorer V6.0, or later
  - Mozilla V1.8, or later
- To support correct bidirectional data input, UTF-8 must be specified. For instructions about how to do this, see "Encoding settings" on page 430.
- Default alignment of HATS rich client applications is left-to-right. For example, selection lists, tool bar icons, and function keys are left-to-right aligned. To provide right-to-left alignment for HATS rich client platform applications, you must:
  1. Create a .product file using the New Product Configuration wizard by selecting **File > New > Other > Plug-in Development > Product Configuration**. The .product file appears in the Navigator view in the project_name\src folder.
  2. Select the **Launcher** tab.
  3. Enter `dir rtl` in the Program Arguments section.

## Working with the host terminal

The HATS host terminal allows Host On-Demand bidirectional-specific keystrokes, therefore you can perform the following bidirectional functions. The following host function keys are available for both 3270, 5250 and VT sessions:

**Ctrl+L: Latin Layer**
This key combination changes the language layer to Latin and the operator information area (OIA) is updated to show English Language.

**Ctrl+N: Bi-di Layer**
This key combination changes the language layer to bidirectional and the OIA is updated to show the bidirectional language.

**Ctrl+S: Screen reverse**
If the screen orientation is left-to-right, this key combination changes the screen image to right-to-left and the language layer changes to bidi. If the screen orientation is right-to-left, this key combination reverses the screen image to left-to-right and the language changes to Latin. This is not available in VT visual sessions.

**Ctrl+F: Field reverse**
If the field orientation is left-to-right, this key combination changes the field orientation to right-to-left, the cursor moves to the other side of the field, and the language layer becomes bidirectional. If the field orientation is right-to-left, this key combination changes the field orientation to left-to-right, the cursor moves to the other side of the field, and the language layer becomes Latin. This is for 5250 sessions only.

The following host function keys are available only for 3270 sessions:

**Ctrl+P: Push**
You can enter and edit text in the opposite direction from the field direction.

**Ctrl+O: End Push**
Push mode is ended and the cursor moves to the end of the push segment.

**Ctrl+A: Auto Push**
You can type mixed left-to-right and right-to-left text by changing the language layer.

The following host function key is available only for 5250 sessions:

**Ctrl+C: Close**
The data entered in one keystroke direction (either left-to-right or right-to-left) is concatenated with the data that was previously entered in the opposite direction. The cursor direction is set to be the same as the field direction, and the language layer is set to the default for the field direction. If the screen orientation is currently left-to-right, the cursor is positioned at the first null to the right of the concatenated text. If the screen orientation is currently right-to-left, the cursor is positioned at the first null to the left of the concatenated text.

The following host function keys are available only for Arabic VT sessions:

**Ctrl+K: Column heading**
Column heading mode causes blanks between columns of text to break insertions allowing the English titles to columns of data to maintain their correct position.

The following host function keys are available only for Hebrew VT sessions:

**Ctrl+D : Cursor direction**
If the current cursor direction is left-to-right, pressing Ctrl+D changes it to right-to-left, and back again. This function is allowed for visual text type only. When the cursor direction is set to right-to-left, this does not affect cursor addressing and moving, insert and delete characters, erase in line or erase in display. The following functions are affected by right-to-left cursor direction settings:

- Backspace (the cursor moves one position to the right)
- Carriage return (the cursor moves to the right most position on the current line)
- Line feed (the cursor moves to the right most position of the next line)
- Typing in the auto wrap mode (current line is continued from right most position of the next line)

**Ctrl+D : Character set modes**
This function switches between 7-bit and 8-bit character sets. If the current character set is DEC Hebrew (8-bit) or ISO Hebrew Supplemental (8-bit), pressing Ctrl+B changes the current character set to Hebrew NRCS (7-bit). If the current character set is Hebrew NRCS (7-bit), pressing Ctrl+B loads one of two 8-bit character sets based on the following rule:

- If the session is configured with one of the 8-bit character sets, that session's 8-bit character set will be loaded. If the session is configured with 7-bit character set, ISO Hebrew Supplemental will be loaded. The current language layer and cursor direction are not changed.

## Host terminal limitations

There is no support for Unicode fields in the Host Terminal; therefore, switching languages in Unicode fields is not possible.

Complex text, containing both bidirectional text and numbers, should not be used as screen recognition criteria.

### Capturing screens

In bidirectional sessions, screens can be captured either as left-to-right screens or as right-to-left screens. Captured screens are displayed exactly as they were captured. To capture a screen as a right-to-left screen, press **Ctrl+S** (Screen reverse) in a left-to-right screen and click **Create Screen Capture**.

## Recognizing bidirectional host components

Typically the orientation of a host component is the same as the screen orientation. However, in some cases, the orientation of a host component is the opposite of the general orientation of the screen. When a HATS project uses a bidirectional code page, a check box labeled **Recognize component oriented opposite to screen** is added to the Insert Host Component wizard and the Create a Screen Combination wizard. When you add a host component to a transformation, select this box if the host component's orientation is the opposite of the screen orientation. Selecting the box enables HATS to recognize command prompts, function keys, selection lists, subfile, and input fields with hints whose orientation is the opposite of the screen orientation.

Mixed bidirectional text may be used in function key and selection list settings. This means that strings like: BIDI1 = BIDI2, where BIDI1 and BIDI2 are segments of bidirectional text, may be recognized as function keys and selection lists .

By default, only left-to-right components may be recognized in default rendering of screens (as in an English session). However, selection of the **Recognize component oriented opposite to screen** property in the Project Settings editor, on the **Rendering** tab in the **Default Rendering** section will enable recognition of both left-to-right and right-to-left components in default rendering.

Also by default, components on a screen transformation are recognized only according to the screen orientation. At the project level you can specify that if a component is not recognized, HATS should try to recognize it using the opposite screen orientation. To make this specification, in the Project editor on the **Rendering** tab in the **Advanced** section under **Default Rendering**, select the **Recognize component oriented opposite to screen** check box. If this selection is made and the component is still not recognized, then rendering will follow the specification made with the setting, **Use default rendering on component rendering failure**.

You can also specify this setting at the component level. To do so, when you add a component to a transformation using the Insert Host Component wizard, on the Rendering Options panel, click the **Advanced settings for rendering** button, and select the **Recognize component oriented opposite to screen** box. The value of this setting at the component level is not inherited from the project setting. The following table shows the possible combinations of this setting at the project and component levels and which setting is used. For more general information see “Advanced rendering” on page 93.

*Table 16. Project and component setting combinations*

| Project setting | Component setting | Setting used |
|---|---|---|
| cleared | cleared | component |
| cleared | selected | component |
| selected | cleared | project |
| selected | selected | component |

## Controlling the orientation of widgets

There are two special check boxes that provide you control over widget presentation when your application runs in the WebSphere Application Server. This control applies to:

- The whole customized screen, in case of **Prepopulated using Default Rendering** or **Prepopulated using Fields** in the new transformation wizard
- Selected specific widgets in the case of the:
  - Insert Host Component wizard
  - **Properties** view of a HATS specific component on a transformation
  - **Rendering options** or **Rendering** tab of a HATS screen combination

These two check boxes are:

- **Reverse text in widget**
- **Reverse widget's orientation**

On a left-to-right screen, when the **Reverse widget's orientation** check box is selected, the GUI image is changed to be right-to-left. Similarly, when the **Reverse text in widget** check box is selected, the text within the GUI image is changed to right-to-left.

For example, suppose you have the left-to-right screen that contains the following text (upper case letters are bidirectional data and lower case letters remain as English data):

```
BIDI TEXT

pf01=help
```

When you customize this screen as RTL. it is displayed as:

```
TXET IDIB

pleh=10fp
```

From the user's point of view, everything is correct except the function key. The function key is recognized, but it is displayed backwards.

For the customized screen to appear correctly, you must select both the **Reverse widget's orientation** and the **Reverse text in widget** check boxes when inserting the function key host component into a customized screen.

When you do this, the screen is displayed as:

```
TXET IDIB

pf01=help
```

When you replace a text input field with a drop-down list in a bidirectional project, the captions of the drop-down list items are not affected by the screen orientation . The captions will always appear in visual LTR format, unless you select the **Reverse text in widget** check box for the widget. In that case the captions will be appear in visual RTL format.

Also, when creating bidirectional HATS applications, and copying and pasting multiple lines of text from a protected field into a visual input field, the text may be reversed. If you copy and paste one line at a time, the orientation will be preserved.

## Controlling the alignment of widgets

To assign the proper alignment of widgets containing HATS component elements, a **Component Alignment** setting is available on the following HATS configuration panels:

- Rendering tab of the Project Settings editor, for Web application default rendering
- Rendering options panel in the Create a Transformation wizard
- Actions panel of the Create a Screen Customization wizard

In addition, to assign the proper alignment of individual widgets, **Field Alignment** and **Table Alignment** settings are available in the widget settings dialog of the Rendering Options wizard for the Field and Table widgets.

The values of these settings are `left` and `right`. The **Component Alignment** setting in the project settings affects the alignment of all widgets in the project. The value set in the project settings becomes the default for the Create a Transformation and Create a Screen Customization wizards. The alignment specified for an individual widget overrides the value specified for default rendering.

## Customized components and widgets bidirectional support

Bidirectional counterparts should be used when customizing new components and widgets, for example in Web projects:

```
import com.ibm.hats.transform.widgets.BIDI.LabelWidgetBIDI
```

public class newBIDILabel extends LabelWidgetBIDI implements HTMLRenderer

## Global variables

When global variables are extracted, the extracted data is exactly the data that appears on the screen, including the orientation of the current screen. Any global variables inserted onto a screen as an action of a screen event are inserted according to the screen orientation.

In both Web and rich client applications, bidirectional data from static text and input fields is stored in global variables in visual format, in both the Rational SDP and runtime environments.

When you use the Insert Global Variable wizard to insert a global variable value as static text into a transformation for a bidirectional screen, you have the option to **Insert global variable in visual format**. Select this option to insert the global variable value as static text in visual format. Clear this option to insert the value as static text in logical format.

When global variables are extracted from logical VT sessions, the extracted data is converted to visual format. Any global variables inserted onto a VT logical sessions, are converted to logical format. Text direction is set according to screen orientation.

## Text replacement

When you use text replacement in bidirectional sessions on a project level or HATS component tag level, there are three additional check boxes you can use:

**Match on LTR screen**
This option allows text replacement to be performed correctly for text on a left-to-right display screen.

**Match on RTL screen**
This option allows text replacement to be performed correctly for text on a right-to-left display screen.

**Match with screen reverse**
If you select this box and the **Match on LTR screen** check box, the reversed string would match in a right-to-left screen. When the **Screen reverse** button is clicked on a left-to-right screen, the data is consistent.

If you select this box and the **Match on RTL screens** check box, the reversed string would match in a left-to-right screen. When the **Screen reverse** button is clicked on a right-to-left screen, data is consistent.

You must select either **Match on LTR screen** or **Match on RTL screens**.

For example, suppose a left-to-right host screen contains the text: `NO on`. If you define text replacement to replace `no` with `yes` and ignore the case, the results depend on the boxes you selected, as follows:

*Table 17. Bidirectional text replacement options and results*

| Options selected | LTR screen | RTL screen |
|---|---|---|
| Match LTR screen only | yes on | no ON |
| Match RTL screen only | NO on | yes ON |
| Match LTR screen and RTL screen | yes on | yes ON |
| Match LTR screen and match screen reverse | yes on | no sey |
| Match RTL screen and match screen reverse | NO sey | yes ON |
| Match LTR and RTL screen and match screen reverse | yes sey | yes sey |

## Global rules

When you use either project-level or screen-level global rules in bidirectional sessions, there are three additional check boxes you can use (similar to text replacement):

**Match on LTR screens**
This option allows the configured global rule to be performed correctly for text on a left-to-right display screen.

**Match on RTL screens**
This option allows the configured global rule to be performed correctly for text on a right-to-left display screen.

**Match with screen reverse**
If you select this box and the **Match on LTR screen** check box, the reversed string would match in a right-to-left screen. When the **Screen reverse** button is clicked on a left-to-right screen, the data is consistent.

If you select this box and the **Match on RTL screens** check box, the reversed string would match in a left-to-right screen. When the **Screen reverse** button is clicked on a right-to-left screen, the data is consistent.

For more information, see Table 17 on page 443.

## Enabling the user to reverse the screen direction

In the application keypad settings you can provide a **Screen reverse** button on bidirectional transformations for the convenience of your users. The button appears on the application keypad (for Web applications) and on the Transformation view toolbar (for rich client applications). Clicking the button toggles the screen orientation—the operation performed on host systems by pressing the Alt+Enter keys.

At the project level you can include the **Screen reverse** button in the Application keypad section of the **Rendering** tab in the Project editor. At the transformation level you can include the **Screen reverse** button using the **Insert Application Keypad** functions in the transformation editor.

When you create a new HATS project and select a bidirectional code page, two additional check boxes appear on the Connection Setting panel: **Enable screen reverse for non-customized screens** and **Enable screen reverse for customized screens**. These check boxes determine whether the **Screen reverse** button you provide appears on these screens. By default, both boxes are selected. If you select the first box, the second box is enabled. There is no way to select only the second box.

A non-customized screen is one that is not matched by any screen customization. A non-customized screen, when viewed by the user, has the same screen orientation as the previous screen. If the first screen is non-customized, its screen orientation is left-to-right. If the screen orientation is changed on a previous screen, it is inherited by the next non-customized screen and reset by the next customized screen.

The initial screen orientation of customized screens is the same as it was when the screens were customized. For both customized and non-customized screens, clicking the **Screen reverse** button changes the screen orientation. During screen recognition, a reversed screen is considered different from the same screen before the screen has been reversed. Therefore, the **Screen reverse** button can cause a screen not to be recognized. If the developer is confident that all host components that appear on a customized screen are oriented properly, there is no need to enable the **Screen reverse** button for that screen. It is advisable to disable the **Screen reverse** button for customized screens.

Enabling the **Screen reverse** button to the user of an Arabic application will enable another check box to control shaping orientation. For more information see:
“Functions for Arabic code pages” on page 451

## VT bidirectional display options

For VT bidirectional sessions, some specific bidirectional properties need to be set to the terminal to work correctly for bidirectional users. A VT **BIDI display options** button which appears in the **Connection Settings** panel is added to access these options:

**Text type (Hebrew)**
Determines the format of the text characters stored. Select **Visual** or **Logical**. The default is **Visual**.

**Cursor direction (Hebrew Visual)**
Sets the cursor direction left-to-right (LTR) or right-to-left (RTL). When cursor direction is set RTL, all characters are displayed in the RTL direction because the cursor moves left by default after each displayed character. In general, only applications that are designed to receive input in a RTL direction will work properly when the Cursor Direction is set to RTL. The default is **LTR**.

**Smart Ordering (Arabic/Hebrew Logical)**
Determines whether segments of characters with different text attributes are ordered separately. The default is **Off**.

**Show Text Attributes (Arabic/Hebrew Logical)**
Enabled only when Smart Ordering is On. The default is **Yes**.

## Macro prompt and extract bidirectional options

When editing a macro prompt (or extract) in the Visual Macro Editor or the host terminal, you can define the host screen orientation to use for the prompt (or extract) during macro playback at runtime. Options are **Inherit from host screen capture**, **LTR**, and **RTL**. Selecting **Inherit from host screen capture** uses the same screen orientation value defined for the host screen associated with the macro prompt (or extract). If no host screen is associated, this option is not available, and you must specify either **LTR** or **RTL**.

If a pre-HATS V8.0 macro is copied into a HATS V9.7 project, or if a pre-HATS V8.0 project is migrated to HATS V9.7, then if a host screen is associated with the macro prompt (or extract), the **Inherit from host screen capture** option is used. Otherwise, the default value of **LTR** is used.

For Hebrew VT bidirectional logical sessions, there are two cursor direction radio buttons (**Left to right** and **Right to left**) which provide control over input control orientation for prompt action while recording a macro. In the editor for the VT connection, click **BIDI display options** to display the radio buttons.

The default input control orientation is the LTR direction which implies that the prompt value is in LTR implicit format.

## Macro SQL query bidirectional options

In the SQL wizard for bidirectional languages (Arabic and Hebrew), specific bidirectional properties need to be set. You can access these settings by clicking the **Advanced** button in the **Advanced Macro** editor panel. This panel is visible only if your workstation is configured for a bidirectional language. The following settings are available:

**Host-File Type**
This option specifies whether the Host file should be saved in logical or visual format. The default is **Visual**. (This option is not enabled for the JDBC driver from the AS/400 Toolbox for Java).

**Local-File Type**
This option specifies whether local files are in logical or visual format. The default is **Logical**.

**Host-File Orientation**
Specifies whether the Host file should be saved in left-to-right or right-to-left format. The default is **Left-to-Right**. (This option is not enabled for the JDBC driver from the AS/400 Toolbox for Java.)

**Local-File Orientation**
Specifies whether local files are in left-to-right or right-to-left format. The default is **Left-to-Right**.

**Lam-Alef Expansion**
Specifies the behavior of the Lam-Alef characters. When receiving Arabic data from the host through the SQL Wizard statement, the character Lam-Alef is expanded into two characters, Lam followed by Alef if there is space after Lam-Alef character.

**Lam-Alef Compression**
Specifies the behavior of the Lam-Alef characters. When sending Arabic data to the host through the SQL wizard statement, the characters Lam followed by Alef are compressed into one character and space is added after Lam-Alef character. This option is enabled for Arabic systems only. The default is **on**. (This option is not enabled for the JDBC driver from the AS/400 Toolbox for Java.).

**Symmetric Swapping**
Specifies the behavior of the symmetric characters such as brackets; the inversion of the screen causes directional characters to be replaced by their counterparts. The default is **on**. (This option is not enabled for the JDBC driver from the AS/400 Toolbox for Java).

**Round Trip**
Controls how bidirectional text mixed with numerals is processed.

**Numerals Shape**
Specifies the shape of the numeral on Host file at SQL wizard statement; the numeral shape can be (NOMINAL, NATIONAL and CONTEXTUAL). This option is enabled for Arabic systems only. The default is **NOMINAL**. (This option is not enabled for the JDBC driver from the AS/400 Toolbox for Java).

## Right-To-Left printing support

Bidirectional files can be either RTL or LTR files and in order to print correct RTL data, you must select the **Enable RTL Printing** check box which appears on the **Printing** tab of the Connection editor for the bidirectional 3270E default connection. Both PDF and non-PDF files are supported.

This check box is enabled if printing is enabled.

In the case of Arabic applications, enabling RTL printing enables the following two check boxes.

- **Enable Symmetric Swapping**: If symmetric swapping is enabled, swapping characters are swapped with RTL file printing.
- **Enable Numeric Swapping**: If numeric swapping is enabled, English numerals are replaced by Arabic numerals and Arabic numerals are replaced by English numerals with RTL file printing.

## BMS map support

Bidirectional support for BMS maps includes:

1. Bidirectional code pages: When importing BMS map sets to bidirectional projects, The following code pages should be used:
   - For Arabic projects: **Host Code Page** 420, **BMS File Code Page** 1256 or 864
   - For Hebrew projects: **Host Code Page** 424 or 803 , **BMS File Code Page** 1255
2. Bidirectional attributes: When importing a BMS map set into a bidirectional project, a **BIDI Options** button, is provided. Click this button to set the following BIDI attributes:
   - The text type of the BMS source file being imported.
     - Text Type: **Visual** or **Logical**
   - Options for screen captures created from the BMS map set:
     - Screen Orientation: **Left to Right** or **Right to Left**
     - Symmetric Swap: **On** or **Off**
     - Numeric Swap: **On** or **Off**

## Integration Object support

Integration Object support includes extending the Integration Object bean with bidirectional-specific properties that allow such things as reordering extracted text. For more information see the chapter, Using the HATS bidirectional API, in the *HATS Web Application Programmer's Guide*.

Support for Integration Object Web pages (Model 1 and JSF) includes the correct reordering of extracted/inserted data in runtime and full control by the application developer over screen and extraction/insertion controls orientation using a high level GUI.

Support includes the following capabilities:

- If you create Model 1 or JSF Web pages, you are provided the ability to set the direction of the generated web page (**LTR** or **RTL**) using the **Page direction** radio button options in the **Create Model 1 Web pages** or **Create JSF Web pages** dialog.

  You are also provided the ability to select either a **Visual** or **Logical** text interface using the **Input text** and **Output text** radio button options.
  - Selecting the **Visual** interface for input text allows data to be entered into input field prompts in visual input field format. Selecting the **Visual** interface for output text allows extracted results to be displayed in visual field format.
  - If **Logical** output is chosen, then you can select the **Prevent round-trip** option to use bidirectional marks to achieve correct reordering of mixed bidirectional text.
  - To create mashups using the iContext method or other methods which allow HTML contents to be modified with external data, which is probably in logical format, you must specify **Logical** for the **Input text** setting.
- If you defined input (or output) properties for the Integration Object, they are displayed on the **Define inputs** (or **Define outputs**) dialog. The property name, type of input (or output) control, and control direction are displayed. Click **Edit** to change these settings. On the **Define input property** (or **Define output property** ) dialog, you can change the leading text and type of input (or output) control. To change the control direction (**LTR** or **RTL**) use the **Input control direction** (or **Output control direction** ) radio buttons.

**Notes:**

1. The direction of the individual input (or output) controls is independent of the **Page direction** setting.
2. The direction of each input (or output) control defaults to the same direction as its corresponding prompt (or extract).
3. Use of the **Input control direction** and **Output control direction** radio buttons are supported for Dojo widgets as well as for HTML controls in Model 1 Web pages. However, bidirectional text and widget orientation are not correctly displayed in the Design view for Dojo Combo box and Text box widgets.

**Note:** The alignment of bidirectional text in tables may not render correctly when working with JSF Web pages. This is a limitation of the JSF design.

## Support of bottom-up Web services

In addition to supporting Model 1 and JSF Web pages generated from Integration Objects, HATS provides bidirectional support for bottom-up traditional (WSDL-based) Web services generated from Integration Objects. In this case, the Web service inherits all of the bidirectional functionality of the Integration Object. The Web service is extended by properties that allow such things as reordering extracted text. These properties can be tested using tools such as sample JSP pages and the Web services explorer. For more information see the chapters, Creating traditional (WSDL-based) Web services and Using the HATS bidirectional API, in the *HATS Web Application Programmer's Guide*.

Because sample JSP pages generated by the HATS Toolkit are logical LTR applications, they display the results of the Web service properly only if you set to invoke prompt and extract reordering, and set RTL reordering in both cases to **False**. Otherwise, the result will display incorrectly. However, you may want to see the "incorrect" result to verify how changes of the properties' values affect the Web service.

If you need to see correct results for extract reordering other than logical LTR with sample JSP pages, you may need to manually edit the Result.jsp file generated by the HATS Toolkit. Following the line with the `<body>` tag, insert the line, `<div dir=rtl>` for logical RTL, `<bdo dir=ltr>` for visual LTR, or `<bdo dir=rtl>` for visual RTL.

Use of the Web services explorer for cases of reordering other than logical LTR, is not recommended.

When creating or updating Web service support files, on the Choose Properties page, where you select the input and output properties that you want exposed in the Web service, the following bidirectional input and output properties are selected by default. When exposed to the Web service, these properties can be changed by the consumer of the Web service.

- hExtractRTLTextOrientation
- hExtractReordering
- hPreventBidiRoundTrip
- hPromptRTLTextOrientation
- hPromptReordering

## Support of RESTful Web services

RESTful Web services are supported for bidirectional sessions. When creating or updating RESTful service resources, on the Configure JAX-RS Resource Method page, where defaults are set that reflect the HTTP method and Integration Object you selected, the following bidirectional input and output method parameters are also set to their defaults.

- hPromptReordering (default is **Logical**)
- hExtractReordering (default is **Logical**)
- hPromptRTLTextOrientation (default is **false**)
- hExtractRTLTextOrientation (default is **false**)
- hPreventBidiRoundTrip (default is **false**)

You can change the settings for these method parameters to meet your requirements. For more information see the chapters, Creating RESTful Web services and Using the HATS bidirectional API, in the *HATS Web Application Programmer's Guide*

## Portlet support

Communication between portlets is implemented in HATS using global variables. HATS global variables are stored in visual LTR format, which is typical for 3270 and 5250 systems. However, data stored in other operating system environments is typically stored in logical LTR format. This means that by default HATS portlets use bidirectional text strings stored in the visual LTR format, while non-HATS portlets may use logical LTR format for bidirectional text strings.

To enable the exchange of bidirectional text strings between HATS portlets (using visual LTR format) and non-HATS portlets (using logical LTR format), you must specify to HATS that format conversion is required. To do this in a sending HATS portlet, when adding or editing the Send Global Variable action, select **Convert to logical bidi format**. For more information, see "Send global variable action" on page 159. To do this in a receiving HATS portlet, when adding or editing the received property (JSR168) or event (JSR286), select **Convert from logical bidi format**. For more information, see "Portlet settings" on page 124.

**Note:** Conversion should not be performed when communicating between two portlets using the same format, for example between two HATS portlets.

## Information for users

HATS applications using bidirectional code pages offer a special input field called a visual input field. Unlike regular fields, which implement logical data input and presentation, the visual field implements visual data input and presentation. When entering data in a visual field, you can use these functions:

**Alt+Shift: Language selection**
This key combination toggles the language layer back and forth between Latin and the bidirectional language.

**Alt+Enter: Screen reverse**
This key combination reverses the direction of the screen.

**Shift+NumLock: Push**
You can enter and edit text whose direction is opposite from the field direction.

**Shift+NumPad "/" character: End push**

Push mode is ended and the cursor moves to the end of the push segment.

**Alt+NumPad "/" character: Auto push**

You can type mixed left-to-right and right-to-left text by changing the language layer. Autopush is especially useful for typing digits in right-to-left fields. The push and end push functions are automatically activated according to the language of the text being typed. In right-to-left fields, typing a digit or a Latin letter causes the automatic initiation of push, without a language change. Additional Latin letters or digits will continue the push mode; any other character automatically terminates push mode. This feature allows you to type bidirectional text with embedded numbers or Latin words without using push and end push.

In left-to-right fields, typing a bidirectional character causes the automatic initiation of push. Typing any digit or Latin character causes the automatic termination of the mode. This enables the user to type Latin text with embedded bidirectional words by using language layer selection rather than push and end push.

You can initially enable or disable auto push mode using the **Enable Auto Push mode initially** setting. This setting is available in two places within the project editor. On the **Overview** tab, click the link to the current theme and use the **Enable Auto Push mode initially** setting. On the **Other** tab select the Client Settings section and use the **Enable Auto Push mode initially** setting.

**Note:** When using auto push on a JSF Web page that interfaces to an Integration Object, you may need to change the field size manually if necessary to enter more characters than the default browser input field size allows.

## Auto field reverse

**Alt+NumPad 5**

- When auto field reverse is enabled, upon initial entry to an alphanumeric field, the field orientation will be set to right-to-left (for both left-to-right and right-to-left screen orientations).
- When auto field reverse is enabled, upon initial entry to a numeric field, the field orientation will be set to left-to-right (for both left-to-right and right-to-left screen orientations).
- When auto field reverse is disabled, upon initial entry to a field (whether numeric or alphanumeric), the field orientation is always set equal to the screen orientation.

When Auto Field Reverse is activated all fields, with the exception of password fields will have the same (described above) behavior. In 3270 sessions right-to-left screens, auto field reverse is the default mode of operation.

## Field reverse

**Alt+Numlock**

When field reverse is activated, typing orientation in current (focused) field will be changed. If the field is empty, cursor will be adjusted to the opposite side of the field. When field is not empty, the cursor position will not change. Field reverse is active only in the current field. When focus is lost, the typing orientation returns to its previous value.

## Clipboard functions

Clipboard operations (such as copy, paste, and cut) with bidirectional data from platform logical editors into HATS visual field, and from HATS visual fields (including static text) into platform logical editors are enabled on Windows platforms. Users working in Windows can use the clipboard to exchange data between HATS and other applications that use logical data.

Internet Explorer clipboard operations are supported both from the Edit option on the menu bar or with the help of shortcuts (Ctrl+C, Ctrl+X, Ctrl+V).

Clipboard bidirectional data support on Mozilla browsers on Windows platforms will be enabled only if these functions are invoked using the shortcuts.

## Operator information area (OIA)

In addition to session related information being displayed in the Operation Information Area (OIA), bidirectional sessions include information about screen and typing direction, active layer, auto field reverse mode status and shaping status (in Arabic sessions).

Bidirectional session (keyboard) related information is presented in a form similar to that used on the host system. To display this information in the OIA of a HATS transformation, in the Project Settings editor on the **Rendering** tab, select **Operator Information Area** and then select the **Show OIA** check box. When the **Show OIA** check box in the Project Settings editor is cleared, the HATS bidirectional status area is displayed in the status bar for Web applications, but not displayed anywhere for rich client applications.

## Automatic keyboard layer switching

For HATS Web projects, with this function you can simulate switching the keyboard layer between bidirectional and Latin languages automatically when the screen is reversed. The simulation is performed through the use of keyboard remapping. This forces insertion of Arabic/Hebrew or Latin characters in accordance with the field typing direction. For example, in a reversed screen, when text typing takes place from right to left, Arabic/Hebrew characters are inserted irrespective of the actual keyboard layer state. To enable this function, in the Project Settings editor, on the **Overview** tab, click the current theme and select **Enable automatic keyboard layer switch**. This setting can also be selected in the Project Settings editor on the **Other** tab in the **Client Settings** section.

HATS rich client projects always behave as if this function is enabled.

**Note:** Password fields, and other special fields that are not based on visual input field, function according to the system keyboard setting, that is, keyboard remapping is not performed for these types of fields.

# IBM i 5250 Unicode support

Bidirectional text that appears in Unicode fields that use 5250 systems is displayed and edited using contextual ordering scheme (according to the first strong character) instead of using visual input field.

# Functions for Arabic code pages

These functions are specific to projects using Arabic code pages.

## HATS Toolkit screen orientation

To start the HATS Toolkit development environment using the desired screen orientation, you can use the HATS Toolkit start menu items. Use of the Arabic translation of the HATS Toolkit is dependent on Arabic support provided by the underlying Rational SDP platform. When installed on an Arabic translation of Rational SDP, two HATS Toolkit start menu items are provided, **HATS Toolkit (Arabic)**, which starts the Arabic translation of HATS Toolkit, and **HATS Toolkit (English)**, which starts the English translation of HATS Toolkit. Search the Knowledge Center for your Rational SDP platform to determine if Arabic is supported.

If not using the HATS Toolkit start menu items, to specify the orientation to use for the HATS Toolkit , use the -nl or -dir Eclipse startup arguments as shown in the table below.

*Table 18. HATS Toolkit orientation settings*

| Eclipse startup argument | HATS Toolkit orientation |
|---|---|
| -nl <BIDI locale> and the BIDI language pack is installed | right-to-left |
| -nl <BIDI locale> and the BIDI language pack is not installed | left-to-right |
| -nl <BIDI locale> -dir rtl | right-to-left |
| -dir rtl | right-to-left |
| -nl <BIDI locale> -dir ltr | left-to-right |
| -nl <non-BIDI locale> | left-to-right |
| -dir ltr | left-to-right |

For example, start the HATS Toolkit using `eclipse.exe -OpenHatsPerspective -nl ar -dir rtl` for Arabic. If no Eclipse startup arguments are specified, the HATS Toolkit orientation is left-to-right regardless of the development workstation locale.

## Customizing the direction for JSP pages

In some cases, you might need to customize a template, transformation, or other JSP page, for example, the stop.jsp page, to create the desired direction. To do this, add the attributes `dir=rtl` and `align=rtl` to the HTML tags as appropriate. For example, for the stop.jsp page you could add `dir=rtl` to the `<table>` tag that surrounds the text of the stop.jsp page.

## Shaping direction

If you enable the **Screen reverse** button in the application keypad settings, the **RTL Shaping** button is also displayed for Arabic code page projects. Clicking the **RTL Shaping** button switches the state of RTL shaping on or off. Turning RTL shaping off informs HATS that the user is currently editing a LTR file independent of screen orientation. Turning RTL shaping on informs HATS that the user is currently editing a RTL file independent of screen orientation, thus the same Arabic shaped data can be submitted correctly in both screen orientations. You should ensure before editing and submitting an LTR file that RTL shaping is turned off and ensure that RTL shaping is turned on before editing and submitting an RTL file.

## Arabic selective shaping

In some cases you may need to send your data to the host in isolated format. To meet such requirements a field disable shaping property has been added for Arabic sessions through two check boxes:

- **Disable field shaping for password fields**:

  When you create a new HATS project and select an Arabic code page, this check box appears. It determines whether the shaping for password fields is enabled or disabled. Hence it determines if the Arabic data of password fields will be submitted to the host in isolated or shaped form.

- **Disable field shaping**:

  This check box appears in the **Insert Host Component** wizard . It determines if the Arabic data, inside input fields in this selected region, will be submitted to the host in isolated or shaped form

## Symmetric and numeric swapping

These options are effective only in Arabic 3270 sessions. If symmetric swapping is enabled, swapping characters are swapped in right-to-left screens. If numeric swapping is enabled, English numerals are replaced by Arabic numerals in right-to-left screens and Arabic numerals are replaced by English numerals in right-to-left screens. These parameters are set on the **Advanced** tab of the connection editor in the **Configure optional, advanced connection settings** section. The parameters are identified as symmetricSwapEnabled and numericSwapEnabled, and the values of the parameters are either true or false.

The HATS host terminal is also affected by symmetric and numeric swapping parameters.

## Disable entry of Arabic-Western digits

When you create a new HATS project, or when you create or edit a HATS connection, and select the Arabic Speaking code page 420, the setting, **Disable entry of Arabic-Western digits in RTL screens**, is displayed. Select this option to disable entry of Arabic-Western numbers, that is, allow entry of only Arabic-Indic numbers in RTL screens. Do this so that, when submitted, all numbers are submitted as Arabic-Western numbers.

## Automatic keyboard layer switching

The Arabic keyboard has two layouts, 101 keyboard layout and 102 keyboard layout, which are slightly different in some keys. When using the automatic keyboard layer switching function, the following mappings are performed:

- The Tatweel character is mapped to shift+j for both keyboards.
- The Greater than and Less than key will be mapped to the key left of the 1 key for both keyboards.
- The Zal character is mapped to the key left of the Enter key for both keyboards.

## Host terminal limitations

The following limitations appear with both system keyboard layout English and the Arabic 101 keyboard:

- The dot (.) and slash (/) characters in the keypad area of the keyboard are not displayed correctly when using an English keyboard layout and an Arabic language layer.

- The Arabic character Tatweel using an English keyboard and an Arabic language layer is typed as the ZAL character. You can use the **shift+J** combination to type the ZAL character.

These limitations disappear with the Arabic 102 keyboard.

When using an Arabic 102 keyboard, the following keyboard limitations appear:

- Shift+s to type the letter **S** will result in nothing typed
- Shift+j to type the letter **J** will result in Arabic character Tatweel
- Shift+l to type the letter **L** will result in a backslash (\)

In order to correctly display any of these characters, you will need to switch the keyboard layer to English.

## Screen captures

For an Arabic session with right-to-left captured screens, brackets and numerals are affected by the symmetric and numeric swapping options of the application.

**Note:** In Arabic sessions, the Euro sign is displayed as a block in a captured screen.

## Setting Arabic printing sessions on System i Access for Web

Following is a summary of the settings to present Arabic data properly on System i Access for Web.

For Global settings:

- System value QCCSID (37). This is for the English language to allow the HTTP, TomCat, and background server to start properly if they need to use POSIX characters not represented in code page 420.

SCS reports:

- The report's code page is picked up from spool file's Character set identifier (CHRID) reflected in the printer file parameter (CHRID) that is picked up from the device description then system value. To display Arabic characters properly if the system value QCCSID is not set to 420 perform the following steps:
  1. CHGPRTF or OVRPRTF with the Character identifier parameter CHRID(235 420). Graphic character set 235 and Code page 420.
  2. Regenerate the spool file.

AFP reports:

- Add the fonts and overlay report to the resource library list in one of the two methods:
  - Add the library QFNTAR3 to system library list system value QSYSLIBL
  - Add the resource libraries containing the overlays and fonts to the printer file parameter: User resource library list USRRSCLIBL. This parameter overrides the system value setting above and does not check the rest of the libraries defined in system library list

  Regenerate the spool file.

## Other considerations

- To view Arabic numbers correctly in widget previews and in the deployed HATS application, digit substitution should be set to **None** in the regional settings.
- Screen recognition should always be done with whole Arabic words and not with a part of an Arabic word.
- Bidirectional functions are not supported in the transformation design, this is due to a limitation in the WebSphere Studio V5 JSP design with bidirectional data.
- Bidirectional IBM i input restrictions are not supported in password fields in Arabic sessions.

# Additions to HATS files

When a project uses an Arabic code page, screen event (.evnt) files have an additional <orientation>true</orientation> tag within the description tag. A value of true indicates that the screen is customized as a right-to-left screen; otherwise it is customized as left-to-right.

When a project uses any bidirectional code page, the connection (.hco) file has an additional enableScrRev attribute of the <hodconnection> tag, that can have the following values:

**(blank)**
The **Screen reverse** button is not placed on any screens.

**NotCustomized**
The **Screen reverse** button is placed only on screens that do not match a screen customization.

**Customized**
The **Screen reverse** button is placed on all screens.

When a project uses any Arabic code page, the connection (.hco) file has an additional disableFldShp attribute of the <hodconnection> tag that can have either true or false values

If printing is selected in 3270 session an additional printRTLSupport setting will appear in <classSetttings> tag. For Arabic 3270 sessions, another two settings will appear for symmetric and numeric swapping which are printSymSwapSuppot and printNumSwapSupport. All these printing settings have values of true or false equivalent to enable or disable of the feature.

When a project uses any bidirectional code page, the application (.hap) file has additional attributes for the <replace> tag within the <textReplacement> tag:

**matchLTR**
Text is to be replaced when the screen orientation is left-to-right.

**matchRTL**
Text is to be replaced when the screen orientation is right-to-left.

**matchReverse**
Text is to be replaced when the screen orientation is reversed.

# Chapter 24. Double-byte character set support

This chapter explains the functions and considerations that are unique to developing HATS applications that provide double-byte character set (DBCS) support.

In addition to general functions, HATS provides the following specific functions to support DBCS:

- Data type checking
- Field length checking
- Copy-and-paste prevention
- Input Method Editor (IME)
- AutoIME switching
- Shift Out/Shift In considerations

This chapter explains these specific functions and special considerations when using HATS general functions to support DBCS.

## DBCS and SBCS field support

HATS supports the following types of DBCS and SBCS fields.

*Table 19. DBCS and SBCS field support*

| Field type | Allowable field contents | Platform |
|---|---|---|
| DBCS mix | SBCS and DBCS characters with Shift Out/Shift In (SO/SI) characters | 3270 |
| DBCS only (G) | DBCS characters without SO/SI | 3270 |
| Alphanumeric only | All SBCS characters but no DBCS characters | 3270 |
| Numeric only | Only digits 0 through 9, comma (,), period (.), plus sign (+), minus sign (-), and space ( ) | 3270 |
| | | |
| DBCS open (O) | SBCS and DBCS characters with SO/SI | 5250 |
| DBCS graphic (G) | DBCS graphic characters without SO/SI | 5250 |
| DBCS only (J) | DBCS characters with SO/SI | 5250 |
| DBCS either (E) | Either DBCS with SO/SI or SBCS characters | 5250 |
| Alphabetic only (X) | Only characters A through Z, comma (,), period (.), hyphen (-), and space ( ) | 5250 |
| Alphanumeric shift (A) | All SBCS characters but no DBCS characters | 5250 |
| Numeric shift (N) | All SBCS characters but no DBCS characters | 5250 |
| Signed numeric (S) | Only digits 0 through 9 | 5250 |
| Numeric only (Y) | Only digits 0 through 9, comma (,), period (.), plus sign (+), minus sign (-), and space ( ) | 5250 |
| Katakana field (W) | All SBCS characters but no DBCS characters | 5250 |
| Inhibit keyboard entry (I) | No input from the keyboard except from special devices, for example a light pen | 5250 |

*Table 19. DBCS and SBCS field support (continued)*

| Field type | Allowable field contents | Platform |
|---|---|---|
| Digits only (D) | Only digits 0 through 9 | 5250 |
| Numeric only character (M) | Only digits 0 through 9, comma (,), period (.), plus sign (+), minus sign (-), and space ( ) | 5250 |

## Data type checking

Each type of DBCS field must contain only the allowable type of data. For example, the DBCS graphic (G) field must contain only DBCS graphic characters without SO/SI characters. HATS performs data type checking to ensure that only allowable data types can be entered into DBCS fields.

## Field length checking

In a field on a host screen, DBCS characters have a length of two. In a field on a GUI page, DBCS characters have a length of one. This HATS function checks the length of DBCS characters being entered into a GUI field and does not allow more characters than will fit into the corresponding host field.

## Copy-and-paste prevention

This HATS function prevents a user from copying and then pasting data with an incorrect data type into a field. If mixed data is pasted, all characters after the first illegal character is trimmed. For example, if the user tries to paste `abc&cde` into a field, and the `&` is an illegal character for the field type, only `abc` is pasted.

**Note:** This function is supported only for HATS Web applications using Internet Explorer on Windows platforms and HATS rich client applications.

## Input Method Editor (IME)

The Input Method Editor (IME) is a front-end-processor (FEP) which handles keyboard input and generates DBCS strings. Typically, the IME is a component of the operating system, but it may be a single product or a part of another application like a word processor.

DBCS users can generate thousands of different characters using a regular-sized keyboard. Typically a user types a sequence of characters and then uses the IME to convert the sequence to double-byte characters. The conversion may have to be retried because there may be several possible translations, or candidates, for the same sequence of characters. While this conversion process is going on, the text logically belongs to the IME. After the conversion process, the user then uses the IME to commit the converted double-byte characters to the application. HATS supports use of the IME for committing double-byte characters into HATS fields.

If the user tries to enter characters into the IME that exceed the maximum length of the rendered HATS field, then the behavior differs depending on the **Eliminate maximum length in DBCS fields** and **Eliminate maximum length in SBCS fields** settings. For more information about the **Eliminate maximum length in DBCS fields** setting, see “Eliminate maximum length in DBCS fields” on page 460. For more information about the **Eliminate maximum length in SBCS fields** setting, see “SBCS eliminate maximum length” on page 462.

## AutoIME switching

Host screens may often contain both DBCS and SBCS fields. HATS will automatically turn on IME when the cursor is located on a DBCS field and turn off IME when the cursor is located on an SBCS field.

**Notes:**

1. Support is provide only for HATS Web applications using Internet Explorer and HATS rich client applications on Windows platforms.
2. AutoIME is supported for DBCS sessions for all fields except DBCS mix (3270), DBCS open (O), DBCS either (E), and Inhibit keyboard entry (I) .
3. Support is provided at the field level and not at the character level.
4. Support is not provide in the Combo widget.
5. In HATS Web applications on Japanese Windows platforms, the IME will switch to DBCS (or Zenkaku) hiragana mode when the cursor is in a katakana(W) field. Users can input DBCS (or Zenkaku) katakana characters, then use function key F8 to get halfwidth katakana characters, or use function key F10 to get alphanumeric characters.

## Shift Out/Shift In considerations

For DBCS mix, DBCS open (O), DBCS only (J), and DBCS either (E) fields, if the **Show unprotected Shift Out/Shift In characters as spaces** setting in the **DBCS** section of the **Rendering** tab is selected, the SO/SI characters are displayed as spaces in the GUI. If not selected, the SO/SI characters are not displayed in GUI.

To configure this property for connections used for HATS Integration Objects, use the **omitSOSI** Host On-Demand session parameter. This parameter can have a value of **true** or **false**. If **true**, the HATS Integration Object omits the SO/SI characters for the connection. If **false** (the default), the HATS Integration Object replaces the SO/SI characters with a space, as is the case normally. For information about how to add the **omitSOSI** property to your Integration Object connection, see “Configure optional, advanced connection settings” on page 135.

## Other considerations

This section explains other considerations to understand when using the general HATS functions to develop applications that support DBCS.

### Project theme settings

#### DBCS eliminate maximum length

**Eliminate maximum length in DBCS fields**

Use this setting at the theme level to select whether to eliminate the maximum length attribute in DBCS fields. You can also select this setting in the project settings editor. For more information about this setting, see the “Eliminate maximum length in DBCS fields” on page 460 setting on the Rendering tab of the project settings editor.

### Project settings editor

The following settings are available in the project settings editor for DBCS projects. See Chapter 5, “Modifying a HATS project,” on page 87 for other project settings.

## Rendering tab

**DBCS:**

On the **Rendering** tab you can configure the following DBCS options for your project if your default connection specifies a DBCS code page.

**Show unprotected Shift Out/Shift In characters as spaces**
Select this box to display unprotected Shift Out and Shift In characters as spaces in the GUI for DBCS mix, DBCS open (O), DBCS only (J), and DBCS either (E) fields. If not selected, SO/SI characters are not displayed in the GUI.

**Automatically convert SBCS characters to DBCS for G-type and J-type fields**
Select this box if you want to automatically convert single byte characters to double byte characters for 3270 and 3270E G-type or 5250 G-type and J-type fields.

**Eliminate maximum length in DBCS fields** `Web-only`

If not selected, then characters that exceed the maximum length of the DBCS field cannot be entered into the IME.

If selected, then

- If the **Enable automatic field advance** setting in the Project settings editor on the Rendering tab in the Client Settings section is selected, then characters that exceed the maximum length of the DBCS field can be entered into the IME. Also, when the user selects the IME candidate for the field, the excess characters are cut and pasted into the next field.
- If the **Enable automatic field advance** setting is not selected, then characters that exceed the length of the DBCS field can be entered into the IME. However, when the user selects the IME candidate for the field, the excess characters are removed and not entered into any other field.

This setting is supported for the following field types:

- 3270
  - DBCS mix
  - DBCS only (G)
- 5250
  - DBCS only (J)
  - DBCS graphic (G)
  - DBCS either (E)
  - DBCS open (O)

**Notes:**

1. This setting is available only for DBCS code pages in Web projects, including portlet projects.
2. This setting is supported only for Internet Explorer.

**Set default ATOK Input Mode to Roman** `RCP-only`
Select this box to set the ATOK default input mode to Roman. If not selected, the ATOK default input mode is set to Hanji. Valid only for rich client platform applications.

## Other tab

In the **Client Settings** section of the **Other** tab, consider the following settings for DBCS projects.

**Enable automatic field advance:**

For HATS Web applications, the auto advance feature depends on the current character count and field length to decide the time of advance. When the **Show unprotected Shift Out/Shift In characters as spaces** setting in the **DBCS** section of the **Rendering** tab is not selected, the character count never reaches the field limit if a DBCS character is in the field. So for HATS Web applications, if you need the auto advance function in the DBCS environment, you must select the **Show unprotected Shift Out/Shift In characters as spaces** setting.

When using the IME to input data into a field, two tables in the KBS.js file are used to define keys that trigger auto advance. A set of default triggering keys is provided. If necessary, you can edit the tables in the KBS.js file to change the defaults. To edit the KBS.js file, from the **Navigator** view double-click the file, which is located in the *project_name*\Web Content\common folder. The following example shows the default keys defined in the two tables. For example, the ALT + SBCS/DBCS (keycode 25) key combination is a default triggering key. To add a key, enter its keycode (either its variable name defined in the KBS.js file or its numeric value) and a key modifier (ALT, CTRL, or Shift) as appropriate. For how to determine the keycode value for a given key on the keyboard, see "Determining keycode values" on page 365.

```
var IMEkeysForKeyUp = [
//-Keycode----ALT-CTRL-Shift----//
[  CODE_ENTER,  0,  0,  0  ],
[  CODE_M,      0,  1,  0  ],
[  CODE_ENTER,  0,  1,  0  ]
];

var IMEkeysForKeyDown = [
//-Keycode----ALT-CTRL-Shift----//
[  25,          1,  0,  0  ] //ALT + SBCS/DBCS key
];
```

**Overwrite mode (initial):**

The overwrite mode (initial) feature is not supported for DBCS input.

**Notes:**

1. When you type into a browser, the browser controls whether you can overwrite. Some browser implementations do not allow any overwriting, while others allow overwriting only for SBCS.
2. One DBCS character is inserted before any SBCS or DBCS character regardless of whether overwrite mode is enabled.
3. An SBCS character can overwrite one DBCS character, including SO/SI characters.

**Select all text on focus:**

If this setting is selected, when a user tabs to a 5250 J-field, all the text in the field is selected when the field receives focus. If the user then inputs DBCS characters, and uses the Field Exit function to exit the field, the last character of the string may be deleted.

To avoid this behavior, clear the **Select all text on focus** box and select the **Show unprotected Shift Out/Shift In characters as spaces** box. In this case, the cursor is positioned properly in the first input position of the field. The user can begin typing at this position and then use the Field Exit function to exit the field.

**Note:** These considerations only apply to 5250 J-fields in Web applications.

# Screen event editor

## Screen recognition criteria / Begin screen

You should not select screen recognition criteria which contains half of a double-byte (DBCS) character. Any DBCS character selected must be completely included in the selection region. See "Screen Recognition Criteria or Begin Screen" on page 148 for more.

# Host components

## Selection list

The following setting is available for the selection list component for DBCS projects. See "Selection list" on page 203 for other selection list component settings.

**Convert action number (DBCS only)**

If your host displays full-width characters for the list items' leading token, but it expects the item to be entered into the target input field as a single-byte character, then this parameter must be enabled. This will cause HATS to convert the DBCS item into a SBCS item before it is sent to the host. This setting does not apply for items which are displayed as single-byte characters on the host.

## Subfile

Following is a consideration when using the **Column Breaks** setting of the Subfile component. See "Subfile" on page 206 for other Subfile component settings.

**Column Breaks**

Column breaks cannot split a DBCS character.

## Table

Following is a consideration when using the **Column Breaks** setting of the Table component. See "Table" on page 212 for other Table component settings.

**Column Breaks**

Column breaks cannot split a DBCS double-byte character.

# Widgets

## SBCS eliminate maximum length

**Eliminate maximum length in SBCS fields (DBCS codepage only)**

If not selected, then characters that exceed the maximum length of the SBCS field cannot be entered into the IME.

If selected, then

- If the **Enable automatic field advance** setting in the Project settings editor on the Rendering tab in the Client Settings section is selected, then characters that exceed the maximum length of the SBCS field can be entered into the IME. Also, when the user selects the IME candidate for the field, the excess characters are cut and pasted into the next field.
- If the **Enable automatic field advance** setting is not selected, then characters that exceed the length of the SBCS field can be entered into the IME. However, when the user selects the IME candidate for the field, the excess characters are removed and not entered into any other field.

This setting is supported for the following widgets:

- Calendar
- Field
- Popup
- Subfile (check box)
- Subfile (drop-down)
- Subfile (popup)
- Table
- Text input

This setting is supported for the following field types:

- 3270
  - Alpha numeric
- 5250
  - Katakana (W)
  - Alpha numeric (A)
  - Float (F)
  - Numeric shift (N)

**Notes:**

1. This setting is supported only for DBCS code pages in Web projects, including portlet projects.
2. This setting is supported only for Internet Explorer V6 and V7.
3. When this setting and the **Enable automatic field advance** setting are both selected, if a user inputs characters using the keyboard, instead of using the IME, the automatic field advance feature does not work for the supported field types listed above.

## Macro support

Following are considerations when using macro support in DBCS projects.

### Recording a macro

When recording a macro you can use the **Add Extract Action** to extract information from the host screen. See more information about the Add Extract Action "Add Extract Action" on page 330. Consider the following case when extracting DBCS characters.

When you extract a region of the host screen into a table, you can modify the format of the table. One of the modifications you can make is to divide a table column into two columns. When you click **Divide**, HATS divides the selected column. If the column contains double-byte (DBCS) characters, the division can split a DBCS character into two columns. In this case, the DBCS character will not appear in either column. If you see this happen while editing an extract action, use the **Right** and **Left** buttons to adjust the columns. The divided DBCS character will reappear when it is contained in a single column.

### Exporting a macro

If you export a macro containing DBCS characters to a HOD macro, and then import it into Host On-Demand , the DBCS characters may become unreadable. To avoid this problem, you can open the exported macro with an editor that supports UTF-8 encoding, such as Notepad, save the macro using native encoding, and then import the macro into Host On-Demand.

## Creating an Integration Object

Following is a consideration when creating an Integration Object in a DBCS project. For more information, see “Creating an Integration Object” on page 341.

When naming a macro used to create an Integration Object, double-byte (fullwidth) characters are not allowed.

## Working with mobile device applications

On Internet Explorer Mobile, changes to input fields for DBCS supported features are made when the input field loses focus, that is, when the blur event occurs. This is different from support on desktop browsers where changes are made in real time using the key event.

For data type checking, all characters beginning with the first illegal character are stripped from the input field when the input field loses focus, that is, when the blur event occurs.

For more information, see “Developing HATS applications for mobile devices” on page 44.

## Remapping keyboard and display characters

You can use HATS support for Host On-Demand custom tables to remap the user's keyboard and display characters by customizing the code page conversion tables. For more information, see “Remapping keyboard and display characters” on page 435.

## Working with user-defined characters

DBCS languages consist of ideographic characters instead of phonetic alphabets. Thousands of ideographic characters are used in these languages, and currently defined code pages or fonts do not always meet the requirements for these characters. User-defined characters (UDCs) are used to support the necessary characters that are not defined in code pages or fonts. Using the UDC mapping editor, you can assign a local code point from the User-Defined Area (UDA) to a specified host code point that is reserved for this character.

The HATS UDC functions support 3270, 3270E, and 5250 connection types. You use HATS UDC mapping support to display UDCs for 3270, 3270E, and 5250 connections, and you use HATS font-image support to print UDCs in either PDF or PDT mode for 3270E connections.

### Display support

To enable UDC display support for 3270, 3270E, and 5250 connections, follow these steps:

1. Launch the UDC wizard.
   a. In the HATS Projects view, edit the connection in your DBCS project.
   b. In the connection editor, click the **Advanced** tab.
   c. On the Advanced tab, click **Set Up User-Defined Characters (UDC) Files**.
2. On the Select default mapping table page:
   a. Select **Enable UDC mapping support**.
   b. To launch the mapping editor for the code page defined for the connection, select **Launch UDC mapping editor** and click **Launch**. Use the mapping editor to make your changes. When you save your UDC changes, a

mapping table file with the .gtt file extension is stored in the <*RationalSDP_install_directory*>\hats\UDC\udctransl\ directory. The following code pages are supported:

*Table 20. UDC mapping editor supported code pages*

| Code page | Location or usage |
|---|---|
| 930 | Japanese (Katakana) |
| 930 | Japan (Katakana Extended) |
| 933 | Korea (Extended) |
| 937 | Taiwan (Traditional Chinese Extended) |
| 939 | Japanese (Latin Extended) |
| 1364 | Korea Euro |
| 1371 | Taiwan (Traditional Chinese) Euro |
| 1388 | PRC (Simplified Chinese Extended; GB18030) |
| 1390 | Japanese (Katakana Unicode Extended) |
| 1399 | Japanese (Latin Unicode Extended) |

c. To import an existing mapping table directory, for example, one that was created using the Host On-Demand product, select **Import available UDC mapping table directory** and click **Browse**. A valid mapping table directory must contain one mapping index file (udctables.inx) and some mapping table files (*.gtt).

d. From the Select default mapping table section, select the table to use as the default. If necessary, click **Refresh List** to refresh the list of tables displayed.

e. Click **Next**.

3. On the Summary page, notice all of the setup information. Click **Back** to make any changes. Otherwise, click **Finish**.

When you click **Finish** in the UDC wizard, HATS copies the related UDC files to the appropriate HATS project and updates the **UDC_SETTING** and **UDC_TABLE_SELECTION** advanced connection settings in your project.

## Print support

To enable UDC print support for 3270E connections, follow these steps:

1. Launch the UDC wizard.
   a. In the HATS Projects view, edit the connection in your DBCS project.
   b. In the connection editor, click the **Advanced** tab.
   c. On the Advanced tab, click **Set Up User-Defined Characters (UDC) Files**.
2. On the Select default mapping table page, click **Next**.
3. On the Select font-image file page:
   a. To convert Windows user-defined fonts into a usable font-image file, select the appropriate conversion tool from the UDC font-image conversion tool drop-down, and click **Launch**.

      **Notes:**

      1) For the Japanese Windows 2000 operating system, select the **Japanese Windows 2000** conversion tool, otherwise, select the **Standard** conversion tool.

      2) If you use the Japanese Windows 2000 tool to convert the UDCs on Windows 2000, all the UDCs defined by Unicode are converted. HATS

can print not only the UDCs defined in local workstation codes (a total of 1880 characters), but also those defined in Unicode including the host UDCs (0x6941-0x89BD for code pages 930 and 939, and 0x6941-0x7FFE for code pages 1390 and 1399).

b. Using the conversion tool, click **Convert** to create a font-image file and save it. The name of the file depends on the operating system platform as shown in the following table.

*Table 21. Font-image filenames*

| Platform | Font-image filename |
|---|---|
| Japanese Windows | jpn24.fnt |
| Korean Windows | kor24.fnt |
| Simplified Chinese Windows | chs24.fnt |
| Traditional Chinese Windows | cht24.fnt |

c. To select the font-image file to use in your HATS application, click **Browse** and browse for the file you created using the conversion tool. If you want to use a font-image file previously created using the Host On-Demand product, browse for it instead.

d. Click **Next**.

4. On the Summary page, notice all of the setup information. Click **Back** to make any changes. Otherwise, click **Finish**.

When you click **Finish**, HATS copies the related UDC files to the appropriate HATS project.

## Printing UDCs in Adobe PDF mode

To print UDCs in Adobe PDF mode, configure these settings on the Printing tab of the connection settings editor:

1. Select **Default printing (Adobe PDF Format)** from the drop-down list in the Initialize Print Settings section.
2. Click **Initialize**.
3. Use the following default values in the Name/Value table:

*Table 22. Name/Value table settings*

| Name | Value |
|---|---|
| printDestination | false |
| printMimeType | application/pdf |
| printSaveAsExtension | .pdf |
| separateFiles | true |
| useAdobePDF | true |
| usePDT | false |
| useWindowsPrinter | false |

## Printing UDCs in PDT mode

To print UDCs in PDT mode, configure these settings on the Printing tab of the connection settings editor:

1. Select **Basic text files (Plain text format)** from the drop-down list in the Initialize Print Settings section.
2. Click **Initialize**.

3. Set the following values in the Name/Value table:

*Table 23. Name/Value table settings*

| Name | Value |
|---|---|
| PDTFile | The path to the PDTfile, for example, *MyPDT/PDTFileName*.hodpdt. See below the table of PDT file names to use depending on your code page and supported PDT printers. |
| printDestination | false |
| printMimeType | application/octet-stream |
| printSaveAsExtension | .out |
| separateFiles | true |
| useAdobePDF | false |
| usePDT | true |
| useWindowsPrinter | false |

**Note:** The following table shows the PDT file names to use for the PDTFile setting in the Name/Value table, depending on your code page and supported PDT printers. For information about using the Host On-Demand PDT compiler, see "Using the Host On-Demand PDT compiler" on page 355.

*Table 24. PDT file names*

| Code page | Supported PDT printer | PDT file name |
|---|---|---|
| 930 Japanese (Katakana)<br>930 Japan (Katakana Extended)<br>939 Japan (Latin Extended)<br>1390 Japanese (Katakana Unicode Extended)<br>1399 Japanese (Latin Unicode Extended) | ASCII text mode | /pdfpdt/basic_dbcs.hodpdt |
| | Printers based on ESC/P 24-J84 | /pdfpdt/esc_p.hodpdt |
| | IBM 5577-B02,F02,G02,H02 | /pdfpdt/ibm5577.hodpdt |
| | IBM 5585-H01 Printer | /pdfpdt/ibm5585.hodpdt |
| | IBM 5587-G01,H01 (without advanced function) | /pdfpdt/ibm5587.hodpdt |
| | Lips3a4 Printer | /pdfpdt/lips3a4.hodpdt |
| | Lips3b4 Printer | /pdfpdt/lips3b4.hodpdt |
| 933 Korea (Extended)<br>1364 Korea Euro | Korea IBM 5577 Printer | /pdfpdt/ibm5577k.hodpdt |
| | Ks_jo Printer | /pdfpdt/ks_jo.hodpdt |
| | Ks_wan Printer | /pdfpdt/ks_wan.hodpdt |
| | Kssm_jo Printer | /pdfpdt/kssm_jo.hodpdt |
| | Kssm_wan Printer | /pdfpdt/kssm_wan.hodpdt |

*Table 24. PDT file names (continued)*

| Code page | Supported PDT printer | PDT file name |
|---|---|---|
| 937 Taiwan (Traditional Chinese Extended)<br>1371 Taiwan (Traditional Chinese) Euro | Traditional Chinese ESC/P Printer (5550) | /pdfpdt/esc_5550.hodpdt |
| | Traditional Chinese ESC/P Printer (big-5) | /pdfpdt/esc_big5.hodpdt |
| | Traditional Chinese ESC/P Printer (cns) | /pdfpdt/esc_cns.hodpdt |
| | Traditional Chinese ESC/P Printer (tca) | /pdfpdt/esc_tca.hodpdt |
| | Traditional Chinese IBM 5577 Printer (without advanced function) | /pdfpdt/ibm5577t.hodpdt |
| | Traditional Chinese IBM 5585 Printer | /pdfpdt/ibm5585t.hodpdt |
| 1388 PRC (Simplified Chinese Extended; GB18030) | Simplified Chinese ESC/P Printer | /pdfpdt/esc_pp.hodpdt |

To print the file content in PDT mode with a local PDT printer, follow these steps:

1. Download the PDT file from the HATS print job list.
2. At a DOS prompt, enter `type xxxx.out > prn` to send the file to the local PDT printer.

## Limitations

- UDC conversion tool limitations:
  - In a Traditional Chinese Windows environment, there are 13 more UDCs than IBM's Big5 UDCs. Therefore, if the last 13 UDCs are defined in the range of 0xC8F2-0xC8FE, they are ignored by the utility and cannot be used.
  - In a Korean Windows environment, only local code page 949 is supported. You can define and print 188 UDCs in the ranges of 0xC9A1-0xC0FE and 0xFEA1-0xFEFE.
- Since User-Defined Area (UDA) ranges are defined based on Windows platforms, UDCs can only be displayed in Windows platforms.

# Appendix A. Runtime properties files

Runtime settings, such as log, trace, and license settings are stored in one of two different files:

- The runtime.properties file is used by HATS in the runtime environment and in the local test environment running in Run on Server mode (for Web applications) or Run mode (for rich client applications).
- The runtime-debug.properties file is used by HATS in the local test environment in Debug on Server mode (for Web applications) or Debug mode (for rich client applications).

For more information about how to administer runtime settings for Web applications, see "Administering problem determination components" on page 381. For more information about how to administer runtime settings for rich client applications, see "Administering HATS rich client applications" on page 76.

The runtime.properties and runtime-debug.properties files contain the following properties.

**adminPortNum**
The adminPortNum is the default value used by the HATS administrative console when the management scope is to be changed. The port value must be an active WebSphere BOOTSTRAP_ADDRESS port. The default is 2809, which is the default BOOTSTRAP_ADDRESS port of server1 for a WebSphere base installation, or of the nodeagent server when you federate an application server node into a deployment manager cell.

**ioPatternKey** `Web-only`
One or more patterns specifying the Integration Objects to be traced. See the description of trace.INTEGRATIONOBJECT below.

Each pattern can contain one or more wildcard (*) character. For example, *IntegrationObject.Callup** specifies that tracing is enabled for all Integration Objects that start with the letters `Callup`. To trace all Integration Objects, specify *IntegrationObject.**

If multiple patterns are specified, they should be delimited with commas.

**numLicenses**
Specifies the number of licenses you purchased. HATS tracks the number of HATS connections to host resources and logs a message when the value exceeds the number of licenses purchased.

The value is an integer. There is no default. Valid values for Authorized User licenses are 1 - 50000. For Value Unit licenses, the value must be -1.

**licenseHardLimit**
An encrypted field containing HATS licensing entitlement. Do not manually alter this field.

**licenseTracking**
Specifies whether HATS records license usage or not. The value is binary. The default is **0**.

**0** HATS does not record license usage.

**1** HATS records license usage for all application servers in a node.

HATS tracks the number of HATS connections to host or database resources and logs a message when the value exceeds the number of licenses purchased. The license usage information is written to a file named license*x*.txt in the `log` directory of the HATS installation directory on the server, where the *x* is either 1 or 2.

The maximum size of the license usage files is 512 KB. When the file size of the license1.txt file reaches 512 KB or if the HATS server is restarted, the file is renamed to license2.txt, and a new license1.txt file is created. The new license1.txt file contains the most recent license usage information. When the new license1.txt reaches 512 KB and is renamed, the old license2.txt is deleted.

The license usage files contain the following information, arranged in rows, with each row representing one hour of operation. The values are separated by a space ( ).

1. Date
2. Time
3. The highest license count since the server was started
4. The highest license count in the last hour (the maximum of the last 60 entries)
5. The license count for each minute (1–60)

**licenseFile**
The name used as a template to generate names for the license files for license usage information. The default value for this property is license.txt.

**logFile**
The name used as a template to generate names for each set of application server files to which log messages are written. The default base name for a log file is *messages.txt*.

**logMask**
Certain macro errors cause an image of a host screen (screen dump) to be placed into the HATS log files.

Screen dumps are classified as INFO messages. To control which types of messages are placed in the HATS log, use this setting. For example, if you do not wish to have screens appear in the HATS log due to security concerns, set logMask=6 to prevent INFO messages (including screen dumps) from being placed in the log.

You cannot shut off ERROR messages with the logMask setting.

**4** Log ERROR messages

**5** Log ERROR and INFO messages

**6** Log ERROR and WARNING messages

**7** Log ERROR and WARNING and INFO messages

**maxLogFiles**
The maximum number of message files. The default is **2**.

The base log file name in runtime.properties is used as a template to generate unique sets of message log files for each application server. The default base name for a log file can be changed in runtime.properties. The application server running HATS is the concatenation of: the underscore (_) character, followed by the name of the HATS instance, followed by another underscore (_) character.

The HATS instance ID (for example *_SSS_*) is then appended to the base file name to generate the template for the log files for an application server. For the log file, this becomes *messages _SSS_.txt*. Finally, an index (1, 2, 3, and so on) is added to this name to distinguish multiple files. So, for example, if the HATS instance ID is *cell_node_server*, the log file for the application server is named *messages_cell_node_server_.txt*. With multiple log files configured, the log file names for this application server are *messages_cell_node_server_1.txt*, *messages_cell_node_server_2.txt*, and so on.

When *messages_cell_node_server_1.txt* reaches maxLogFileSize, it is closed and renamed to *messages_cell_node_server_2.txt*. A new *messages_cell_node_server_1.txt* is opened.

When *messages_cell_node_server_1.txt* reaches maxLogFileSize again, previous log files are renamed—for example, *messages_cell_node_server_2.txt* is renamed to *messages_cell_node_server_3.txt*. Then *messages_cell_node_server_1.txt* is renamed to *messages_cell_node_server_2.txt*, and a new *messages_cell_node_server_1.txt* file is opened.

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

**Note:** When running on WebSphere Application Server for z/OS, the server portion of the log file name also includes the Address Space ID of the Servant which is writing to the log (example: *messages_asid_1.txt*). See "Log and trace file names" on page 382 for details.

**maxLogFileSize**
Specifies the maximum size, in kilobytes, that a message log file reaches before an additional log file is opened.

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

**traceFile**
The name used as a template to generate file names to which HATS trace messages are written. The default base name for a trace file is *trace.txt*.

**maxTraceFiles**
The maximum number of trace information files. The default is **5**.

The base trace file name in server.properties is used as a template to generate unique sets of trace files for each application server. The default base name for a trace file can be changed in runtime.properties. The application server running HATS is the concatenation of: the underscore (_) character, followed by the name of the HATS server instance, followed by another underscore (_) character.

The HATS server instance ID (for example *_SSS_*) is then appended to the base file name to generate the template for the trace files for an application server, which becomes *trace _SSS_.txt*. Finally, an index (1, 2, 3, and so on) is added to this name to distinguish multiple trace files. So, for example, if the HATS server instance ID is *cell_node_server*, the trace file for the application server is named *trace_cell_node_server_.txt*. With multiple trace files configured, the trace file names for this application server are *trace_cell_node_server_1.txt*, *trace_cell_node_server_2.txt*, and so on.

When *trace_cell_node_server_1.txt* reaches maxTraceFileSize, it is closed and renamed to *trace_cell_node_server_2.txt*. A new *trace_cell_node_server_1.txt* is opened.

When *trace_cell_node_server_1.txt* reaches maxTraceFileSize again, previous trace files are renamed—for example, *trace_cell_node_server_2.txt* is renamed to *trace_cell_node_server_3.txt*. Then *trace_cell_node_server_1.txt* is renamed to *trace_cell_node_server_2.txt*, and a new *trace_cell_node_server_1.txt* file is opened.

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

**Note:** When running on WebSphere Application Server for z/OS, the server portion of the trace file name also includes the Address Space ID of the Servant which is writing to the log (example: *trace_asid_1.txt*). See "Log and trace file names" on page 382 for details.

**maxTraceFileSize**
Specifies the maximum size, in kilobytes, that a trace file reaches before an additional trace file is opened.

The value is an integer. The default is **10240** KB.

**traceLogDirectory**
Sets the directory to be used for writing log and trace output when the application EAR is deployed. If this keyword is not specified, or the value specified is not valid, the directory used is *logs*, and it placed under the EAR's installed directory. This value is only used when the EAR is deployed to an application server. It has no effect within the Toolkit environment.

**maxHODThreadManagerThreads**
Specifies the maximum number of threads that can be created in the Host On-Demand ThreadManager thread pool. You can specify a value between 10 and 10000. If you specify a value outside of this range, the setting is ignored and the default Host On-Demand value of **10000** is used.

## Tracing options

The tracelevel values used by the tracing options described in this section represent a hexadecimal digit string. Each bit of the digit string controls one type of tracing for the runtime. The defaults are:

```
tracelevel.1 : 0000000000020000
tracelevel.2 : 000000000000020f
tracelevel.3 : 000000000004023f  (minimum)
tracelevel.4 : 0000000000041a3f
tracelevel.5 : 00000000000c1bbf  (normal)
tracelevel.6 : 00000000000c1bbf
tracelevel.7 : 00000000001c1bbf  (maximum)
tracelevel.8 : 00000000001c1bbf
tracelevel.9 : 00000000001c1bbf
```

Following are the definitions of each bit in the trace mask:

```
x000001  - Informational messages
x000002  - Warning messages
x000004  - Error messages
x000008  - Critical error messages
x000010  - API traces
x000020  - Callback API traces
x000080  - Method entry
x000100  - Method exit
x000200  - Exceptions
x000400  - Miscellaneous traces
x000800  - Object creation
```

```
x001000  - Object disposal
x020000  - performance tracing - use this value alone
x040000  - Miscellaneous data - level 1
x080000  - Miscellaneous data - level 2
x100000  - Miscellaneous data - level 3
```

To customize the tracelevel values, add together the hexadecimal values in the trace mask. For example, if you specify `tracelevel.9=180` and then use `trace.RUNTIME=9` , only method entry and method exit runtime traces are performed.

**trace.RUNTIME**
Specifies the level of tracing for the main runtime and for all settings under RUNTIME.* that do not specify a trace level.

The value is an integer from 0–9. The default is **0**, which means that no runtime tracing is performed.

**trace.RUNTIME.ACTION**
Specifies the level of tracing for the event actions. This setting overrides the trace.RUNTIME setting.

The value is an integer from 0–9. The default is **0**, which means that no event action tracing is performed.

**trace.TRANSFORM**
Specifies the level of tracing for HATS transformations and for all settings under TRANSFORM.* that do not specify a trace level.

The value is an integer from 0-9. The default is **0**, which means that no transformation tracing is performed.

**trace.TRANSFORM.COMPONENT**
Specifies the level of tracing for HATS components. This setting overrides the trace.TRANSFORM setting.

The value is an integer from 0–9. The default is **0**, which means that no component tracing is performed.

**trace.TRANSFORM.WIDGET**
Specifies the level of tracing for HATS widgets. This setting overrides the trace.TRANSFORM setting.

The value is an integer from 0–9. The default is **0**, which means that no widget tracing is performed.

**trace.INTEGRATIONOBJECT** `Web-only`
Specifies the level of tracing for Integration Objects.

The value is an integer from 0–9. The default is **0**, which means that no Integration Object tracing is performed.

**trace.UTIL**
Specifies the level of tracing for HATS runtime utilities.

The value is an integer from 0–9. The default is **0**, which means that no runtime utility tracing is performed.

**trace.APPLET** `Web-only`
Specifies the level of tracing for the applet.

The value is an integer, where

**0** No tracing is performed.

**3** Minimum level of tracing is performed.

**5** Normal tracing is performed.

The default is **0**.

## Host On-Demand tracing

**trace.HOD.COMMEVENT**
Specifies the level of tracing for the Host On-Demand COMM events.

**0** Host On-Demand COMM event tracing is not enabled.

**1** Host On-Demand COMM event tracing is enabled.

The value is binary. The default is **0**.

**trace.HOD.DISPLAYTERMINAL**
Specifies whether Host On-Demand displays a terminal window for each connection.

**0** Host On-Demand does not display a terminal window for each connection.

**1** Host On-Demand displays a terminal window for each connection.

The value is binary. The default is **0**.

If trace.HOD.DISPLAYTERMINAL is 1, when HATS creates a host connection (for example, in response to a request from an application), it automatically creates a host terminal display. If this property is set to 0, this does not occur; however, regardless of whether the host terminal display is created automatically during host connection creation, a HATS administrator can use HATS administrative console to turn host terminal displays on or off for individual host connections.

**trace.HOD.DS**
Specifies the level of tracing for the Host On-Demand data stream tracing.

The value is an integer in the range 0 to 3, where

**0** No tracing is performed.

**1** Minimum level of tracing is performed.

**2** Normal tracing is performed.

**3** Maximum level of tracing is performed.

The default is **0**.

**trace.HOD.MACRO**
Specifies the level of tracing for Host On-Demand macros.

The value is an integer in the range 0 to 2, where

**0** Host On-Demand macro tracing is not enabled.

**1** Host On-Demand event tracing is enabled.

**2** Host On-Demand support tracing is enabled.

The default is **0**.

**trace.HOD.PS**
Specifies the level of tracing for the Host On-Demand presentation space.

The value is an integer in the range 0 to 3, where:

**0** No tracing is performed.

**1** Minimum level of tracing is performed.

**2** Normal tracing is performed.

**3** Maximum level of tracing is performed.

The default is **0**.

**trace.HOD.PSEVENT**
Specifies the level of tracing for the Host On-Demand PS events.

**0** Host On-Demand PS event tracing is not enabled.

**1** Host On-Demand PS event tracing is enabled.

The value is binary. The default is **0**.

**trace.HOD.OIAEVENT**
Specifies the level of tracing for the Host On-Demand OIA events.

**0** Host On-Demand OIA event tracing is not enabled.

**1** Host On-Demand OIA event tracing is enabled.

The value is binary. The default is **0**.

**trace.HOD.SESSION**
Specifies the level of tracing for Host On-Demand sessions.

The value is an integer in the range 0 to 3, where

**0** No tracing is performed.

**1** Minimum level of tracing is performed.

**2** Normal tracing is performed.

**3** Maximum level of tracing is performed.

The default is **0**.

**trace.HOD.TRANSPORT**
Specifies the level of tracing for the Host On-Demand transport.

The value is an integer in the range 0 to 3, where

**0** No tracing is performed.

**1** Minimum level of tracing is performed.

**2** Normal tracing is performed.

**3** Maximum level of tracing is performed.

The default is **0**.

**trace.HOD.USERMACRO**
Specifies the level of tracing for trace actions in Host On-Demand user macros.

The value is an integer in the range 0 to 3, where

**0** No tracing is performed.

**1** Minimum level of tracing is performed.

**2** Normal tracing is performed.

**3** Maximum level of tracing is performed.

The default is **0**.

## Host simulation tracing

**recordSimulationTrace**
Specifies whether host simulation trace recording is enabled at runtime.

**0** Host simulation trace recording is not enabled.

**1** Host simulation trace recording is enabled.

The value is binary. The default is **0**.

**startPort**
Specifies the starting port of the range of ports to use during runtime to perform host simulation trace recording for multiple host sessions concurrently. The default is port **7021**.

**endPort**
Specifies the ending port of the range of ports to use during runtime to perform host simulation trace recording for multiple host sessions concurrently. The default is port **7050**.

# Appendix B. HATS screen-settling reference

This information introduces you to screen-settling, settings for screen-settling, and other settings that affect screen-settling.

## Screen-settling overview

Host applications can send outbound data in any number of transmissions, the number of which is not provided to the client. This situation is handled easily when using a terminal or heavy client terminal emulator, as communications remain always active and results in the client's view being updated whenever necessary. However, when using HATS to transform screen-based host applications into a HTML-based format, the client is only connected to the host (indirectly, through the HATS runtime on the application server) for the duration of each browser update requested by the client. Therefore, HATS must analyze the outbound data received during this browser update cycle, and decide when to send the current host screen to the screen recognition engine for transformation or other event processing. This process of analyzing the outbound data and deciding when to use the current host screen is called screen-settling.

The HATS runtime performs screen-settling:

- Only when transforming screens (that is, not while running a macro or Integration Object, which contain their own requirements for the next screens to check for)
- When it is determined that a host interaction, such as an AID key like [Enter], has taken place and outbound data is expected.

After screen-settling, HATS examines the host screen (presentation space) and compares it to the set of enabled screen customizations (screen recognition) for further processing. It is important that this settled screen is the intended one.

## Screen-settling procedure

HATS performs screen-settling in two distinct operations

- Analyzing outbound data
- Waiting for operator information area (OIA) flags

When analyzing outbound data, HATS performs one of several strategies that analyze the outbound data, trying to determine when the host application has finished sending data. Each of these strategies is used in different cases and each has settings that customize its behavior.

When waiting for OIA flags, which is independent of the strategy used in analyzing outbound data, the session's OIA flags are examined to ensure that the host is ready for the user to send more inbound data. There are also settings that customize this behavior.

## Analyzing outbound data

Depending on runtime circumstances, HATS employs one of several available strategies to determine when the host has finished sending each screen update. HATS uses a strategy based on multiple factors, including the host connection type, connection state, and various user-configurable settings.

If you need to see which strategy your connection is using, you can trace the HATS runtime. For more information about tracing, see "Using the functions in HATS administrative console" on page 379. Each of these strategies can be customized by adding or changing connection settings. Table 25 describes the strategies and Table 26 on page 479 describes the settings.

*Table 25. Available strategies*

| Strategy | Java Class Name | Description and Usage |
|---|---|---|
| Timing | TimingNextScreenBean | A starting value is used as a wait time. Based on the arrival of the host's outbound data transmissions, the wait might include additional intervals, set to half of the starting value, to ensure that all presentation space data has been received. HATS waits an additional interval when presentation space data has been received in the second half of a wait interval.<br><br>This strategy is used when the host type is TN5250, TN3270, and when the host type is TN3270E and contention resolution is not used. |
| Fast 3270E | Fast3270ENSB | This strategy allows improved communication over the TN3270E protocol provided by contention resolution to better determine when the outbound data has completely arrived. This strategy does no additional timing of outbound data arrival, therefore the **default.delayInterval** and **default.appletDelayInterval** settings described in Table 26 on page 479 are ignored.<br><br>This strategy is used when the host type is TN3270E and the contention resolution feature is used on the connection. Contention resolution must be activated on the Telnet server, and negotiated by the Telnet protocol when the connection starts, to be used by HATS. See "Contention resolution (TN3270E only)" on page 482 for more information.<br><br>This strategy is also used for host types TN3270 and TN3270E, when the SNA side of the host session is currently connected to a system services control point (SSCP), because HATS can depend on the TN3270E protocol to quickly determine outbound data arrival. |
| Fast 5250 | Fast5250NSB | This strategy is based on the Timing strategy, but also tries to shortcut the nominal wait, if possible. The shortcut is taken, and the nominal wait ended, when both the OIA System Lock flag is cleared and a non-empty Presentation Space update (that is, containing characters other than blanks) has been received from the host.<br><br>This strategy is used by default for the host type TN5250. |

Table 26 on page 479 describes customization settings for screen-to-screen transitions.

*Table 26. Screen-to-screen transition settings*

| Setting (case-sensitive) | Description | Default |
|---|---|---|
| default.delayInterval | The nominal amount of time, in milliseconds, that HATS waits while analyzing outbound data using screen-settling strategies. The actual time might be more or less than this setting, depending on the screen-settling strategy, such as the Timing strategy adding additional intervals, or the Fast 5250 strategy taking its shortcut. A higher value increases the reliability of screen transformations, in case of delays at the host due to long processing time or network delays. However, increasing this value lengthens response time, especially when using the Timing strategy and this value is the minimum screen-to-screen delay.<br>**Note:** The Fast3270E strategy does not use this setting. | 1200 milliseconds |
| default.appletDelayInterval | The nominal amount of time, in milliseconds, that HATS waits while analyzing outbound data using screen-settling strategies, instead of using default.delayInterval if the asynchronous update applet is enabled and is running for that client. The actual wait might be more or less depending on the screen-settling strategy, such as the Timing strategy adding additional intervals, or the Fast 5250 strategy taking its shortcut. See "Automatic refresh" on page 485 for more information.<br>**Note:** The Fast3270E strategy does not use this setting. | 400 |
| fast3270E.minimumWait | The minimum amount of time, in milliseconds, that HATS waits while analyzing outbound data using the Fast 3270E strategy with negotiated contention resolution. This setting, which is 250 milliseconds by default, is useful in two cases where using a small minimum wait enables more dependable screen-settling.<br>1. Depending on the timing of received outbound data, the Fast 3270E strategy can complete screen-settling before all of the related events are processed at the Telnet client.<br>2. If the host application sends short-duration transient screens that aren't intended to be transformed, you might need to use this setting to ensure HATS waits for the transient screen to be updated with the intended screen.<br>Take care with regard to user response time when modifying this value, as a minimum wait specified with this setting is used for settling all screens. | 250 |
| fast5250.minimumWait | The minimum amount of time, in milliseconds, that HATS waits while analyzing outbound data using the Fast 5250 strategy. This optional setting can be useful in two general cases where using a small minimum wait allows for more dependable screen-settling.<br>1. Depending on the timing of received outbound data, the Fast 5250 strategy can complete screen-settling before all of the related events are processed at the Telnet client.<br>2. If the host application sends short-duration transient screens that aren't intended to be transformed, you might need to use this setting to ensure HATS waits for the transient screen to be updated with the intended screen.<br>Take care with regard to user response time when modifying this value, as a minimum wait specified with this setting is used for settling all screens. | 0 (no minimum) |

*Table 26. Screen-to-screen transition settings (continued)*

| Setting (case-sensitive) | Description | Default |
|---|---|---|
| default.nonHostKeyWait | Specifies the time, in milliseconds, that HATS waits after processing one of a set of pseudo-AID keys that does not require an actual screen-settling step, such as [fldext], [field+], or [eraseeof]. Modifying this setting is not recommended. | 100 |
| nextScreenClass | Allows overriding of the strategy used while analyzing outbound data, by specifying the fully qualified Java class name of the alternate strategy, instead of the strategy that HATS would use. For tuning 5250 applications, this setting is sometimes used to increase reliability if the host application sends 5250 outbound data in a way that results in HATS transforming and displaying partial or incorrect host screens. In this special case, the setting forces analysis of outbound data to use the Timing strategy instead of the Fast5250 strategy, by using a value of **com.ibm.hats.runtime.TimingNextScreenBean**. The Java package name for HATS strategies is **com.ibm.hats.runtime**. Using this setting is not recommended with host types of TN3270 or TN3270E. | Depends on the host type and other factors described in Table 25 on page 478. |

Table 27 describes customization settings for the initial screen, which apply to all strategies.

*Table 27. Initial screen settings*

| Setting (case-sensitive) | Description | Default |
|---|---|---|
| default.delayStart | The nominal amount of time, in milliseconds, that HATS waits while analyzing outbound data, but only for the initial host screen. Since the connection setup process already waits for a usable screen before proceeding, this value is not an additional delay. See connecttimeout in "Related HATS settings" on page 487. It is the minimum amount of time HATS takes, from the time communications are initiated with the host until trying to recognize an initial screen. For fast networks, this value can be decreased to improve initial screen response time. | 2000 |
| ignoreBlankStartupPS | Whether a totally blank presentation space can be used as a successful first screen, after connecting to the host. Most host connections result in a **MSG10** screen or other logon screen. If a completely blank screen can be a valid first screen for HATS to transform or otherwise process, set this value to **false** or HATS will not consider the connection setup to be successful. | true |

## Waiting for OIA flags

After analyzing outbound data, HATS waits for the OIA flags to indicate that the host is ready for the user to send more inbound data. In some cases, OIA flags already indicate this condition after analyzing outbound data, and waiting for OIA flags completes immediately. By default, HATS waits for up to 5 minutes for the STATE_SYS_LOCK (System Lock) and the STATE_TIME (Keyboard Inhibited) flags to be cleared. Both these conditions can be customized with the settings defined in Table 28 on page 481, which apply regardless of which strategy is used to analyze outbound data.

*Table 28. OIA customization settings*

| Setting (case-sensitive) | Description | Default |
|---|---|---|
| keyboardInhibitedWait | Specifies whether to wait for the keyboard inhibited OIA flag to be cleared. If set to `false`, HATS only waits for the system lock OIA flag to be cleared. This setting is used to send keyboard-inhibited screens to the client. | true |
| oiaLockMaxWait | Specifies the maximum amount of time, in milliseconds, that HATS waits for the OIA flags to indicate that the user can send more data. This value can be decreased if you want to return locked screens faster. Values too low can impair screen reliability. It is recommended that if you need to return certain locked screens to the user, try using the keyboardInhibitedWait setting before changing oiaLockMaxWait. | 300000 milliseconds (equals 5 minutes), unless the asynchronous update applet is enabled and running. If so, the default is the value of default.appletDelayInterval, described in Table 26 on page 479. |

## Changing customization settings

Screen-settling customization settings can be changed using the connection editor in the HATS Toolkit. To change the settings, typically in the main.hco file for a connection named main, open the **HATS Projects** view, expand **Connections**, and select **main (default)**.

Five of the settings can be changed on the **Screen Handling** tab of the connection editor:

- default.delayInterval
- default.delayStart
- default.appletDelayInterval
- default.blankScreen

  See Table 29 on page 486.
- default.blankScreenKeys

  See Table 29 on page 486.

All of the settings described above can be configured on the **Source** tab. In the <classSettings> tag, the screen-settling settings are in the **com.ibm.hats.common.NextScreenSettings** class.

You can add or modify the settings as needed to configure screen-settling performance. The default connection settings are:

```
<classSettings>
    ...
   <class name="com.ibm.hats.common.NextScreenSettings">
      <setting name="default.delayInterval" value="1200"/>
      <setting name="default.appletDelayInterval" value="400"/>
      <setting name="default.blankScreen" value="timeout"/>
      <setting name="default.delayStart" value="2000"/>
   </class>
</classSettings>
```

The following example updates one setting and adds another:

```
<classSettings>
    ...
   <class name="com.ibm.hats.common.NextScreenSettings">
      <setting name="default.delayInterval" value="1200"/>
      <setting name="default.appletDelayInterval" value="400"/>
      <setting name="default.blankScreen" value="timeout"/>
      <setting name="default.delayStart" value="1500"/>
      <setting name="nextScreenClass"
               value="com.ibm.hats.runtime.TimingNextScreenBean"/>
   </class>
</classSettings>
```

## Determining which strategy HATS is using

Follow these steps to determine the strategy being used by HATS:

1. Turn on traces for HOD Transport and for HATS Runtime. For more information about tracing, see "Using the functions in HATS administrative console" on page 379. In the runtime.properties file, the settings are:

   ```
   trace.HOD.TRANSPORT=2
   trace.RUNTIME=7
   ```

2. View the trace file (typically contained within the **logs** directory of the HATS.ear file and has a file name starting with **trace**.
3. Look for entries similar to these when HATS performs screen-settling:

   ```
   Text        RUNTIME      runtime.NextScreenBean.getInstance()
      oiaStatusFlags/isSSCP=x7 false

    Text        RUNTIME      runtime.Fast3270ENSB.<init>()
      Create: Fast3270E: initialConnect=false, ttwOriginal=1200, tStartedWaiting=0,
                         tLastPSEvent=0, keyToSend=[enter], minimumWait=250
   ```

   The key text, highlighted in **bold**, is the method name of the first example entry **runtime.NextScreenBean.getInstance()**, which begins a screen-settling process. The resulting object created in the entry begins with `Create:`. The strategy object created in this example has its name (Fast3270E) and its Java class name (Fast3270ENSB) on the second example trace entry, highlighted in *italics*.

## Contention resolution (TN3270E only)

Contention resolution, as defined in the latest draft of the Internet Engineering Task Force (IETF) TN3270E protocol standard (RFC2355), and available on many TN3270E servers, helps overcome limitations in the conversion of the SNA protocol of the host to the Telnet protocol of the clients. Using contention resolution improves the performance of TN3270E clients, including HATS.

The use of contention resolution is negotiated during connection setup, between the Telnet client in HATS (the HACL Telnet client) and the TN3270E server. If HATS successfully negotiates contention resolution with the TN3270E server, communication is more efficient, resulting in fewer delays interacting with host systems. By default, HATS attempts to negotiate contention resolution with any 3270-based host system defined with the TN3270E host type.

### Contention resolution using z/OS Communications Server

Starting with z/OS 1.2, IBM implemented contention resolution in z/OS Communications Server , including VTAM. However, the original implementation in z/OS 1.2, 1.3, and 1.4 did not precisely match the emerging standard. Consequently, HATS might have difficulty communicating with these original

releases of z/OS Communications Server. To address the problem, IBM issued patches (PTFs) and published an informational APAR (technical note) listing the recommended patches.

Ask your z Systems administrator to refer to APAR II13135, entitled "Common Telnet Problems Under z/OS". APAR II13135 lists the recommended PTFs that enable HATS to take advantage of contention resolution. For example, for z/OS 1.4, the APARs PQ71574 and PQ72265 apply, and are corrected by PTF UQ76065. Without the proper fixes, HATS might experience random hangs. HATS developers observed this problem as a clocked screen in the HATS Toolkit terminal window (a screen frozen with a clock icon displayed at the bottom).

If your z Systems administrator cannot immediately install the II13135 patches, use the **negotiateCResolution** setting described in "Related HATS settings" on page 487 and specify a value of `false`. This setting with a value of `false` causes HATS to skip contention resolution negotiation, reverting to the earlier TN3270E specification. Disabling contention resolution can degrade HATS performance by lengthening 3270 response time, but otherwise has no negative consequences.

This contention resolution negotiation problem does not occur when HATS connects through a front end 3270 communications gateway instead of directly to the affected z/OS host system. These gateways might or might not support contention resolution, or might require additional service to be applied to support contention resolution.

## Contention resolution using other Communications Servers

Recent releases of IBM Communications Server for AIX, Linux, Linux on zSeries, and Windows also properly support contention resolution. If it is not possible to upgrade to z/OS Communications Server 1.2 (with patches), or later, these alternative platform versions of Communications Server can provide HATS with the faster response benefits of contention resolution. IBM Communications Server might also prove useful for host systems running versions of VM, VSE, or TPF that do not support contention resolution.

In these instances, HATS connects through TN3270E, optionally with SSL or TLS, to IBM Communications Server running on AIX, Linux, Linux on zSeries, or Windows. IBM Communications Server connects to the host system using an SNA connection, either a traditional SNA connection or Enterprise Extender (SNA traffic flowing over an Internet protocol network). System administrators can manage TN3270E LUs on IBM Communications Server rather than on the host system.

## Performance impact of using contention resolution

Using default settings in HATS, contention resolution can reduce 3270 screen-to-screen transmission delays by approximately 800 milliseconds (in IBM, testing on an unconstrained, fast network). While this delay might be tolerable for casual Web users, contention resolution can prove beneficial for certain high velocity users and for high performance Web services.

Contention resolution has no direct impact on the HATS server processor capacity requirements for any given workload.

## Determining contention resolution status

You can determine whether HATS successfully negotiates contention resolution for a particular TN3270E connection. To examine a HATS trace to see the results of the negotiation, follow these steps:

1. Turn on traces for HOD transport and for HATS runtime. For more information about tracing, see "Using the functions in HATS administrative console" on page 379. In the runtime.properties file, the settings are:

   ```
   trace.HOD.TRANSPORT=2
   trace.RUNTIME=7
   ```

2. View the trace file (typically contained within the **logs** directory of the HATS.ear file with a file name starting with **trace**).
3. Look for entries together in the trace file, similar to these, when the connection was started:

   ```
   Text        HOD          HOD.()
     Correlator: HodConn:localhost:localhost:server1:HATS.ear#1
     trcmsg: -->TN3270 : Negotiate     CMD = DO      OPT = TN3270-E

    Text        HOD          HOD.()
     Correlator: HodConn:localhost:localhost:server1:HATS.ear#1
     trcmsg: <--TN3270 : Response      CMD = WILL    OPT = TN3270-E
   ```

   The key text, highlighted in **bold**, is `CMD = DO` and `CMD = WILL` when `OPT = TN3270-E` also appears (shown here in bold). If both `CMD = DO` and `CMD = WILL` appear, HATS successfully negotiated TN3270E for that particular host connection, and can request contention resolution. If `CMD = WONT`, `CMD = DONT`, or both appear, it is likely that the host does not support TN3270E, and HATS cannot request contention resolution.
4. Look for an entry similar to this one when the connection was started:

   ```
   Text        HOD          HOD.()
     Correlator: HodConn:localhost:localhost:server1:HATS.ear#1
     trcmsg: <--TN3270 : FUNCTION_STATUS: CR REQUESTED
   ```

   The key text, highlighted in **bold**, is **FUNCTION_STATUS: CR REQUESTED**. If this text appears, HATS has requested contention resolution for that particular host connection.
5. Look for an entry similar to this one when the connection was started:

   ```
   Text        HOD          HOD.()
     Correlator: HodConn:localhost:localhost:server1:HATS.ear#1
     trcmsg: <--TN3270E: FUNCTION_STATUS: CR ENABLED
   ```

   The key text, highlighted in **bold**, is **FUNCTION_STATUS: CR ENABLED**. If this text appears, HATS has successfully negotiated contention resolution for that particular host connection.
6. Look for entries together similar to these two when HATS performs screen-settling:

   ```
   Text        RUNTIME      runtime.NextScreenBean.getInstance()
     isEnhancedTNServer=true

    Text        RUNTIME      runtime.Fast3270ENSB.<init>()
     Create: Fast3270E: initialConnect=false, ttwOriginal=1200,
                                          tStartedWaiting=0,tLastPSEvent=0,
                                          keyToSend=[enter], minimumWait=250
   ```

The key text, highlighted in **bold**, is **EnhancedTNServer=true** and **Create: Fast3270E**. If these appear, HATS has successfully negotiated contention resolution and is responding by using the Fast3270E strategy for analyzing outbound data.

## Automatic refresh

HATS provides both client pull and server push functions for updating the client browser with asynchronous outbound data. See "Automatic Disconnect and Refresh Web-only " on page 109 for details about these functions.

Use of these functions is another way to tune the HATS screen-settling process. Whether you use these functions can be based on:

- How your applications work with just tuning the screen-settling settings, if necessary.
- Whether enabling Contention Resolution for TN3270E hosts is an option.
- Whether portal, firewall, or other considerations would limit their use in your deployment.

Using either of these functions can enhance performance by enabling screen-settling times to be reduced, since missed updates can be refreshed on the client when they are received at the application server. Several default screen-settling settings are changed when the server push (applet) function is configured and running on a client. See Table 26 on page 479 and Table 28 on page 481 for more information about these settings.

## Transient screen handling

A transient screen is a host screen that is intended (by the host that sent it) to be displayed to the client for a very minimal amount of time, before showing an application screen for the user to respond to (the intended screen). Two examples of transient screens include:

- A `Please wait` or other in-progress message
- Blank screens, containing only blank characters, generated by screen-clearing commands that might be issued by the host while moving within an application, or from one application or logon screen to another.

**Note:** For information about handling transient screens in macros, see "Handling transient screens" on page 336.

A user running a standard emulator might not notice a transient screen that appears momentarily, and is replaced by the intended screen. When HATS receives a transient screen, this can sometimes result in causing HATS to prematurely decide that the host has finished sending data. The transient screen is compared to the list of enabled screen customizations and in most cases rendered by the default transformation and shown on the client browser. The host sends the intended screen to replace the transient screen, but HATS has already completed screen-settling for that browser update and is not waiting for, or processing, host screen updates. If the user, upon seeing the unintended transient screen transformed, clicks **Refresh**, HATS synchronizes with the current host screen and the client browser is properly updated. This is not the intended overall result, and can typically be avoided by tuning screen-settling delays or by enabling either of the automatic refresh functions in some cases.

An alternate approach to eliminating this undesirable scenario is to create one or more screen customizations to specifically match the transient screens, and no other screens, and take the appropriate action. This action can include nothing, which can be implemented using a Play Macro action of a simple macro that only contains a short <pause> element. Combined with the screen-settling action that HATS performs after processing a screen customization action list, this results in HATS waiting until receiving the intended screen to respond to the browser update request. You must create the recognition criteria for the screen customizations with care, to not match screens you intend for the user to see.

HATS provides additional screen-settling settings, which enable easier customization of blank screen handling, without creating screen customizations and macros. These customization settings, for simpler blank screen handling, enable you to specify the action to take when HATS settles on a blank screen. An action can be waiting, sending a certain host key, or proceeding with screen recognition processing.

For blank screen handling on the initial screen, two additional screen-settling connection settings are available. See “Screen Handling” on page 137 in Chapter 6, “Managing connections,” on page 131 for details on configuring initial blank screen handling.

## Initial blank screen handling settings

Table 29 describes the initial screen handling settings for blank screens.

*Table 29. Blank screen handling settings*

| Setting | Description | Values | Default |
|---|---|---|---|
| default.blankScreen | The system waits for a non-blank screen for the period of time specified in the connect timeout on the Advanced tab. The default.blankScreen setting specifies what to do when the connect timeout expires, if there is still only a blank screen. Valid values are:<br>**timeout**<br>Issue an error message. This is the default value.<br>**normal** Display the blank screen. You can also specify a lower connect timeout if you expect a blank screen as the initial screen.<br>**sendkeys**<br>Send the host key defined on the default.blankScreen.keys setting. You can also specify a lower connect timeout if you expect a blank screen as the initial screen. | timeout, normal, or sendkeys | timeout |
| default.blankScreen.keys | Specifies the host key to send when `default.blankScreen` is set to `sendkeys`. | A valid host key | none |

For blank screen handling after the initial screen, HATS uses a defined application event. See the section titled “Blank screens” under “Application events” on page 102 in Chapter 7, “Working with screen events,” on page 147 for details of this application event.

## Related HATS settings

The following settings do not directly configure HATS screen-settling, but they have an impact on screen-settling, therefore they are listed here for reference:

| Name | Setting Location | Description |
|---|---|---|
| negotiateCResolution | You can add this connection property to the table of advanced connection settings on the **Advanced** tab of the connection editor in HATS Toolkit.<br><br>In the source of the .hco file, this becomes an entry in the <otherParameters> tag. | Tells the Telnet client (HACL) used by HATS whether to negotiate for Contention Resolution on the TN3270E connection. The default is `true`.<br><br>Setting this parameter to `false` causes HATS not to request contention resolution from the Telnet server. It is not used, therefore it causes the HATS screen-settling for this host to use the Timing strategy.<br><br>See "Contention resolution (TN3270E only)" on page 482 for more information about this function and why you might want to disable it.<br><br>The same effect on HATS screen-settling occurs as a side effect of changing the host type to TN3270 instead of TN3270E, since contention resolution is only requested when the host type is TN3270E. |
| connecttimeout | This connection property on the **Basic** of the HATS connection editor, is labeled as **Abandon attempt to connect after specified number of seconds**.<br><br>In the source of the .hco file, this is the connecttimeout attribute of the <hodconnection> tag. **Note:** This is **NOT** the connectionTimeout parameter seen on the drop-down list when you add advanced connection properties in the connection editor. This is a new property, which is exposed to HATS by the Host On-Demand product, when HATS requests a list of available properties. It is not related to the HATS parameter or HATS screen-settling. | Specifies the time, in seconds, that the server waits until a connection is available, either from the connection pool, if pooling is enabled, or from a new connection. The default is 120 seconds. A new connection is available when communications are started with the host and a usable host screen is received. Completely blank screens are not considered usable unless the ignoreBlankStartupPS screen-settling flag is set to `false`. |

| Name | Setting Location | Description |
|---|---|---|
| • ignorepauseforenhancedtn<br>• ignorepauseoverrideforenhancedtn<br>• delayifnotenhancedtn | These macro properties are specified in the source view of a macro. See Actions, Part 2: Timing issues in the *Advanced Macro Guide* for more information. | Macros encounter the same situations as screen-settling regarding when the host has finished sending data, therefore these properties and their descriptions in the *Advanced Macro Guide* are referred to here.<br><br>These macro properties address problems that the macro developer encounters when trying to support a single version of a macro to run on both contention resolution and non-contention resolution environments. They control how <pause> elements of the macro, which are needed for non-contention resolution environments, are performed in a contention resolution environment.<br><br>The <HAScript> property **ignorepauseforenhancedtn** has a default of `false`, meaning that <pause> elements are always processed. When changed to `true`, the <pause> elements are ignored when contention resolution is used.<br><br>The <pause> property **ignorepauseoverrideforenhancedtn**, when set to `true` causes the <pause> element to be performed always, not skipped, even in a contention-resolution environment when **ignorepauseforenhancedtn** is set to `true` in the<HAScript> element.<br><br>The <HAScript> property **delayifnotenhancedtn** should be used whenever **ignorepauseforenhancedtn** is `true`, to enable contention resolution performance benefits, and the macro might also be used in a non-contention resolution environment. The property specifies the delay, in milliseconds, to incur when the OIA changes, and its default is 0.<br><br>See Actions, Part 2: Timing issues in the *Advanced Macro Guide* for more information about these properties. |

## Tuning HATS screen-settling

You might be able to tune HATS screen-settling in several different ways to improve reliability, performance, or both. Following are some decisions that you might need to make:

**Use Contention Resolution to improve performance and reliability?**
If HATS successfully negotiates contention resolution, communication is more efficient, resulting in fewer delays interacting with host systems. See "Contention resolution (TN3270E only)" on page 482 for more information.

**Tune strategy delays?**
You might want to change the value of default.delayInterval to improve response time or to allow for reliable performance in applications or

environments where response time is longer. Some host systems repeatedly have a long response time latency before sending certain screens. The delay might be at a host step that requires considerable processing, or it might involve a network resource that is in heavy demand from time-to-time. In these cases, increasing the delayInterval value might result in more reliable screen-settling. You should accommodate not only the average time for host response, but also the typical longest variation. Choose a value that accommodates your typical operating conditions. Similarly, the default.delayStart, default.appletDelayInterval, fast3270E.minimumWait, and fast5250.minimumWait settings control delays in various cases. See the descriptions for the settings in Table 26 on page 479.

**What happens when a delay like default.delayInterval is set too small?**
The HATS runtime can decide the screen is complete and the host is finished sending data. Some examples of the symptoms that can happen in that case are:

- The HATS screen is blank, and clicking `Refresh` corrects this.
- A screen appears to be partially (incompletely) rendered, and clicking `Refresh` corrects this.
- The wrong screen customization event is called, which can result in the wrong custom transformation being shown, because the screen recognition criteria cannot examine the incomplete screen properly.

**What happens when a delay is set too large?**
The user might experience long response times, and in the worst cases, stop waiting for the screen. Application tuning requires operational judgment regarding the application and the network environment in which it is being used, and there is a balance to be struck.

**Correct other problem scenarios?**
There are situations that appear similar to those solved by tuning strategy delays but have different solutions:

- Enabling applications to process inhibited screens. Use the waiting for OIA flags customization settings **keyboardInhibitedWait** and **oiaLockMaxWait**.
- HATS screen update for 5250 connections is unsuccessful in some cases, even after tuning strategy delays. You might need to turn off the Fast5250 strategy by using the **nextScreenClass** setting.

**Use automatic refresh?**
Based on your network environment and requirements, using automatic refresh might be useful for improving user response time. See "Automatic refresh" on page 485 for more information.

# References

The references in this section also might be helpful in tuning screen-settling.

## Contention resolution information

**RFC2355 extensions (March, 2003, Draft)**
http://www.ietf.org/proceedings/03mar/I-D/draft-ietf-tn3270e-extensions-04.txt

**Informational APAR II13135**
http://www.ibm.com/support/docview.wss?uid=isg1II13135

**z/OS Communications Server**
Refer to the z/OS Knowledge Center at http://www-01.ibm.com/support/knowledgecenter/SSLTBW_1.12.0/com.ibm.zos.r12/zosr12home.html and search on the tern `SNAEXT`.

**IBM Communications Server for Linux, Linux on zSeries, AIX, and Windows**
http://www.ibm.com/software/network/commserver

## Advanced Macro Guide

See Actions, Part 2: Timing issues in the *Advanced Macro Guide* for a discussion of timing issues during macro play that are affected by macro design parameters and options, and screen-settling issues from the macro point of view.

# Appendix C. Notices

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

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

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

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan, Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

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

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

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

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

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

Intellectual Property Dept. for Rational Software
IBM Corporation
5 Technology Park Drive
Westford, MA 01886
U.S.A.

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

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

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

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

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrates programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

## Programming interface information

This User's and Administrator's Guide contains information on intended programming interfaces that allow the customer to write programs to obtain the services of HATS.

## Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at “Copyright and trademark information” at www.ibm.com/legal/copytrade.shtml.

Adobe is a registered trademark of Adobe Systems Incorporated in the United States, and/or other countries.

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

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

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

# Glossary

**action.** A defined task that an application performs on a managed object as a result of an event, such as a host screen matching the screen recognition criteria specified for a screen event. A list of actions is part of the definition of each event.

**ADB.** See **application data buffer**.

**administrative console.** The HATS administrative console is a Web-based utility that provides views and functions to manage licenses and connections, set log and trace settings, view messages and traces, and perform problem determination for HATS Web applications.

**application.** See **HATS application**.

**application data buffer.** The format of data that is returned by the WebFacing Server for consumption by the WebFacing application.

**application event.** A HATS event that is triggered by state changes in the application's life cycle. Examples of application events include a user first accessing a HATS application (a Start event), or an application encountering an unrecognized screen (an Unmatched Screen event).

**application keypad.** A set of buttons or links representing HATS application-level functions. (Contrast with **host keypad**.)

**artifact.** See **resource**

**background connection.** Any connection defined in a HATS application other than the default connection. HATS does not transform screens from background connections. (Contrast with **default connection**.)

**bidirectional (bidi).** Pertaining to scripts such as Arabic and Hebrew that generally run from right to left, except for numbers, which run from left to right.

**BMS map.** A screen definition file used with Basic Mapping Support in CICS. A BMS map defines a set of fields which are to be displayed as a group by a CICS application

**business logic.** Java code that performs advanced functions, such as interacting with other applications, databases, or other systems accessible via Java APIs. Business logic is invoked as an action in an application or screen event.

**checkin screen.** The screen identifying the host screen that should be active for a connection to be considered ready to be returned to the connection pool. If the application is not on the screen specified by the checkin screen, the connection will be discarded or recycled in attempt to return the connection to the host screen specified by the checkin screen. The checkin screen is only meaningful if connection pooling is specified for a connection.

**component.** A visual element of a host screen, such as a command line, function key, or selection list. HATS applications transform host components into widgets.

**connection.** A set of parameters used by HATS, stored in an .hco file, to connect to a host application. (See also **default connection** and **background connection**.)

**connection pool.** A group of host connections that are maintained in an initialized state, ready to be used without having to create and initialize them.

**credential mapper.** The component of Web Express Logon that handles requests for host credentials, which have been previously authenticated by a network security layer. (See **network security layer**.)

**DDS map.** Data Description Specification map. These maps define the layout and behavior of the presentation space for IBM i terminal applications.

**Debug.** For rich client projects, the same as Run, and in addition enables you to:

- Use the display terminal to see host screens as they are navigated while testing your project
- See debug messages in the Rational SDP console

- See changes you make to your project, for example changing the template or a transformation, without having to restart your application
- Modify and test runtime settings, defined in the runtime-debug.properties file, without modifying the settings, defined in the runtime.properties file, that are deployed to the runtime environment
- Step through Java code, such as HATS business logic

**Debug on Server.** For Web projects, the same as Run on Server, and in addition enables you to:
- Use the display terminal to see host screens as they are navigated while testing your project
- See debug messages in the Rational SDP console
- See changes you make to your project, for example changing the template or a transformation, without having to restart your application on the test server
- Modify and test runtime settings, defined in the runtime-debug.properties file, without modifying the settings, defined in the runtime.properties file, that are deployed to the runtime environment
- Step through Java code, such as HATS business logic

**default connection.** The connection on which HATS transforms and presents host application screens to the user. Also referred to as **transformation connection**. (Contrast with **background connection**.)

**default rendering.** The method used by HATS to render parts of the host screen for which no specific transformation is specified.

**deploy.** To make a HATS application ready for use in a runtime environment. For HATS Web applications, this includes exporting the HATS project as a Java EE application, that is, as an .ear file, and installing it on WebSphere Application Server. For HATS rich client applications, this includes exporting the HATS project as an Eclipse feature and installing it on individual client systems, either as a stand-alone Eclipse application or from an update site to an existing Eclipse runtime environment.

**descriptor.** See **screen recognition criteria**.

**developer.** The person who uses HATS Toolkit to develop applications; also application developer or Web developer. (Contrast with **user**.)

**Device Runtime Environment (DRE).** A package containing other runtime environments, including the J2SE runtime, which is required to run HATS rich client applications in Lotus Expeditor Client V6.2.0 and earlier. The DRE installs into the runtime environment for Lotus Expeditor Client.

**display terminal.** A terminal window that displays host screens you can use while testing and debugging to observe interactions between a HATS application and a host application at runtime. You can also interact with the host application using host screens in the terminal window.

**Eclipse.** An open-source initiative that provides ISVs and other tool developers with a standard platform for developing plug-compatible application development tools. Eclipse is available for download from http://www.eclipse.org.

**editor.** An application that enables a user to modify existing data. In HATS Toolkit, editors are used to customize resources that have been created by wizards.

**Enhanced Non-Programmable Terminal User Interface (ENPTUI).** Enables an enhanced interface on non-programmable terminals (NPT) and programmable work stations (PWS) over the 5250 full-screen menu-driven interface, taking advantage of 5250 display data stream extensions.

**enterprise archive (EAR).** A specialized Java archive (JAR) file, defined by the Java EE standard used to deploy Java EE applications to Java EE application servers. An EAR file contains enterprise beans, a deployment descriptor, and Web archive (WAR) files for individual Web applications. (Sun)

**Enterprise JavaBeans (EJB).** A component architecture defined by Oracle for the development and deployment of object-oriented, distributed, enterprise-level applications. (Oracle)

**event.** A HATS resource that performs a set of actions based on a certain state being reached. There are two types of HATS events, application events and screen events.

**export.** To collect the resources of a HATS project, along with the necessary executable code, into an application EAR file (for Web applications) or Eclipse feature (for rich client applications) in preparation for deploying the application.

**Extensible Markup Language (XML).** A standard metalanguage for defining markup languages that was derived from and is a subset of SGML.

**GB18030.** GB18030 is a new Chinese character encoding standard. GB18030 has 1.6 million valid byte sequences and encodes characters in sequences of one, two, or four bytes.

**global rule.** A rule defining how the rendering of specific input fields should be modified based on certain criteria. Global rules are used in customized screens and screens rendered using default rendering. Global rules can be defined at the project level or at the screen event level.

**global variable.** A variable used to contain information for the use of actions. The values of global variables can be extracted from a host screen or elsewhere, and can be used in templates, transformations, macros, Integration Objects, or business logic. A global variable can be a single value or an array, and it can be shared with other HATS applications sharing the same browser session.

**HATS.** See **Host Access Transformation Services**.

**HATS application.** An application that presents a version of a host application to users, either as a Web-enabled application deployed to WebSphere Application Server, a portlet deployed to a WebSphere Portal, or as an Eclipse client-side processing plug-in deployed to an Eclipse rich client platform such as Lotus Notes or Lotus Expeditor Client. A HATS application is created in HATS Toolkit from a HATS project and deployed to the applicable environment. The deployed application might interact with other host or e-business applications to present combined information to a user.

**HATS EJB project.** A project that contains the HATS EJB and Integration Objects that other applications can use to get host data. A HATS EJB project does not present transformed screens from a host application.

**HATS entry servlet.** The servlet that is processed when a user starts a HATS Web application in a browser.

**HATS project.** A collection of HATS resources (also called artifacts), created using HATS Toolkit wizards and customized using HATS Toolkit editors, which can be exported to a HATS application.

**HATS Toolkit.** The component of HATS that runs on Rational SDP and enables you to work with HATS projects to create HATS applications.

**Host Access Transformation Services (HATS).** An IBM software set of tools which provides Web-based access to host-based applications and data sources.

**host component.** See **component**.

**host keypad.** A set of buttons or links representing functions typically available from a host keyboard, such as function keys or the Enter key. (Contrast with **application keypad**.)

**host simulation.** Host simulation enables you to record host simulation trace files that can be saved and then used instead of a live host connection. The recorded trace files can be played back to create screen captures, screen events, and screen transformations using the host terminal function, create and test macros using the host terminal function, test HATS applications using the Rational SDP local test environment, and, along with other traces and logs, aid in troubleshooting a failing scenario in a runtime environment.

**host simulation trace.** Host simulation trace files record host screens and transactions that can be saved and played back later instead of using a live host connection. Trace files can be recorded using the host terminal function or while in the runtime environment.

**host terminal.** A HATS Toolkit tool. A session tied to a particular HATS connection, which the HATS developer can use to capture screens, create screen customizations, and record macros.

**HTML.** Hypertext Markup Language.

**HTML widget.** See **widget**

**Integration Object.** A Java bean that encapsulates an interaction with a host screen or a series of host screens. Integration Objects are constructed from macros and can be included in traditional (WSDL-based) Web services, RESTful Web services, or HATS EJB projects. Integration Objects cannot be used in rich client platform applications.

**interoperability.** The ability of a computer or program to work with other computers or programs.

**interoperability runtime.** Common runtime used by a combined HATS/WebFacing application to provide management of common connection to the backend host. This runtime decides whether data being returned by the WebFacing server should be handled by the HATS or WebFacing part of the application.

**Java Platform, Enterprise Edition (Java EE).** An environment for developing and deploying enterprise applications, defined by Oracle. The Java EE platform consists of a set of services, application programming interfaces (APIs), and protocols that provide the functionality for developing multitiered, Web-based applications. (Oracle)

**JavaServer Faces (JSF).** A framework for building Web-based user interfaces in Java. Web developers can build applications by placing reusable UI components on a page, connecting the components to an application data source, and wiring client events to server event handlers. (Oracle)

**JavaServer Pages (JSP).** A server-side scripting technology that enables Java code to be dynamically embedded within Web pages (HTML files) and run when the page is served, returning dynamic content to a client. (Oracle)

**JavaServer Pages Standard Tag Library (JSTL).** A standard tag library that provides support for common, structural tasks, such as: iteration and conditionals, processing XML documents, internationalization, and database access using the Structured Query Language (SQL). (Oracle)

**JSF.** See **JavaServer Faces**.

**JSP.** See **JavaServer Pages**.

**JSR 168.** The Java Portlet Specification addresses the requirements of aggregation, personalization, presentation, and security for portlets running in a portal environment. Version 1.0 of the Java Portlet Specification, Java Specification Request 168 (JSR 168), defines standards to enable portlet compatibility between portal servers offered by different vendors. See **JSR 286**.

**JSR 286.** The Java Portlet Specification addresses the requirements of aggregation, personalization, presentation, and security for portlets running in a portal environment. Version 2.0 of the Java Portlet Specification, Java Specification Request 286 (JSR 286), defines standards to extend the capabilities of Version 1.0 (JSR 168) to include coordination between portlets, resource serving, and other advanced features. See **JSR 186**.

**JSTL.** See **JavaServer Pages Standard Tag Library**.

**keyboard support.** The ability for a developer to enable a user to use a physical keyboard to interact with the host when the application is running in a Web browser or rich client environment. The developer also decides whether to include a host keypad, an application keypad, or both, in a project. If keypads are included, the developer decides which keys are included and how those keys and the keypad appear in the client interface.

**keypad support.** The ability for a developer to enable a user to interact with the host as if the physical keys on a keyboard were pressed, or to perform tasks related to the application, such as viewing their print jobs or refreshing the screen. See also **application keypad** and **host keypad**.

**linked HATS/WebFacing project.** A project created by linking a single HATS Web project with a single WebFacing project for the purpose of creating an enterprise application that includes a HATS Web application interoperating with a WebFacing application and sharing a connection to a 5250 backend host.

**Lotus Expeditor Client.** A standalone client of the Lotus Expeditor product. It is installed on a user or development machine.

**Lotus Notes Client.** A standalone client of the Lotus Notes product. It is installed on a user or development machine.

**macro.** A macro, stored in a .hma file, automates interactions with the host. It can send commands to the host, enter data into entry fields, extract data from the host, and be used to navigate screens on behalf of the user.

**Model 1 Web pages.** A single JSP that contains the information to be presented to the user, formatting tags that specify how the information is displayed, and logic that controls the order in which pages are displayed. (Contrast with **Struts Web pages**.)

**network security layer.** Software that is responsible for authenticating users and authorizing them to access network resources, such as IBM Tivoli Access Manager.

**Operator Information Area (OIA).** OIA is the area at the bottom of the host session screen where session indicators and messages appear. Session indicators show information about the workstation, host system, and connectivity.

**perspective.** In the Rational SDP workbench, a group of views that show various aspects of the resources in the workbench. The HATS perspective is a collection of views and editors that allow a developer to create, edit, view, and run resources which belong to HATS applications.

**pooling.** See **connection pool**.

**portal.** An integrated Web site that dynamically produces a customized list of Web resources, such as links, content, or services, available to a specific user, based on the access permissions for the particular user.

**print support.** The ability for a developer to specify a printer session to be associated with a host session, and enable the user to view host application print jobs, send them to a printer, or save them to disk. Print support is available only for the default connection

**Profile.** For rich client projects, the same as Run, and in addition enables you to locate the operations that require the most time, and identify actions that are repeated, to eliminate redundancy. You can use this function for performance analysis, helping you to get a better understanding of your application.

**Profile on Server.** For Web projects, the same as Run on Server, and in addition enables you to locate the operations that require the most time, and identify actions that are repeated, to eliminate redundancy. You can use this function for performance analysis, helping you to get a better understanding of your application.

**project.** A collection of HATS resources (also called artifacts) that are created using HATS Toolkit wizards and customized using HATS Toolkit editors. These resources are exported as a HATS application. Types of HATS projects include Web, portlet, EJB, rich client, and for purposes of administering HATS Web (including portlet and EJB) applications, HATS administrative console projects. See **HATS project** or **HATS EJB project**.

**Rational Software Delivery Platform (Rational SDP).** A family of IBM software products that are based on the Eclipse open-source platform and provide a consistent set of tools for developing e-business applications.

**rendering set.** A rendering set is configured by creating a prioritized list of rendering items. Each rendering item defines a specific region in which a specified host component is recognized and then rendered using a specified widget.

**resource.** Any of several data structures included in a HATS project. HATS resources include templates, screen events, transformations, screen captures, connections, and macros. Other Rational SDP plug-ins sometimes call these "artifacts."

**RESTful Web service.** See **Web service, RESTful**.

**rich client.** A plug-in designed to run on the Eclipse Rich Client Platform in a client environment, and designed to provide an enhanced user experience by the appearance and behavior native to the platform on which it is deployed.

**Run.** For rich client projects, a function in Rational SDP that enables you to test your HATS rich client projects in an Eclipse, Lotus Notes, or Lotus Expeditor Client instance. In this mode you can modify and test the runtime settings, defined in the runtime.properties file, that are deployed to the runtime environment. Be aware that any changes made to the runtime settings while testing in this mode are retained and become effective when you deploy the HATS application to a runtime environment.

**Run on Server.** For Web projects, a function in Rational SDP that enables you to test your HATS Web and portlet projects in a WebSphere Application Server as appropriate. In this mode you can modify and test the runtime settings, defined in the runtime.properties file, that are deployed to the runtime environment. Be aware that any changes made to the runtime settings while testing in this mode are retained and become effective when you deploy the HATS application to a runtime environment.

**runtime settings.** Log, trace, and problem determination settings defined in the runtime.properties file that are deployed to the runtime environment.

**screen capture.** An XML representation of a host screen, stored in a .hsc file, used to create or customize a screen customization, screen combination, transformation, global rule, or macro. Screen captures are useful because they enable you to develop a HATS project even when not connected to the host. They are also useful in creating macros which are the core of HATS Integration Object and Web services support.

Screen captures of video terminal (VT) host screens can be used to create or customize a macro using the Visual Macro Editor and as the check-in screen when configuring pooling. They cannot be used to create screen customizations, screen combinations, transformations, default rendering, or global rules.

**screen combination.** A type of HATS screen event designed to gather output data from consecutive, similar host screens, combine it, and display it in a single output page. The screen combination definition, stored in a .evnt file, includes a set of screen recognition criteria for both the beginning and ending screens to be combined, how to navigate from screen to screen, and the component and widget to use to recognize and render the data gathered from each screen.

**screen customization.** A type of screen event designed to perform a set of actions when a host screen is recognized. A screen customization definition, stored in a .evnt file, includes a set of criteria for matching host screens, and actions to be taken when a host screen matches these criteria.

**screen event.** A HATS event that is triggered when a host screen is recognized by matching specific screen recognition criteria. There are two types of screen events, screen customizations and screen combinations.

**screen recognition criteria.** A set of criteria that HATS uses to match one or more screens. When a host displays a screen, HATS searches to determine whether the current host screen matches any of the screen recognition criteria defined for any screen event in your project. If HATS finds a match, the defined actions for the screen event are performed.

Screen recognition criteria are also used in the process of recording a macro; in this context they are sometimes called **descriptors**.

**Secure Sockets Layer (SSL).** A security protocol that provides communication privacy. SSL enables client/server applications to communicate in a way that is designed to prevent eavesdropping, tampering, and message forgery. SSL was developed by Netscape Communications Corp. and RSA Data Security, Inc.

**source.** The files containing the markup language that define a HATS project or one of its resources. Also the name of a folder contained in each HATS project.

**SSL.** See **Secure Sockets Layer**.

**standard portlets.** Portlets that comply with the standard portlet APIs defined by Java Portlet Specifications JSR 168 or JSR 286. See **JSR 168** and **JSR 286**. .

**Standard Widget Toolkit (SWT).** An Eclipse toolkit for Java developers that defines a common, portable, user interface API that uses the native widgets of the underlying operating system.

**Struts Web pages.** Web pages built using the Apache Software Foundation's Struts open-source framework for creating Java web applications. This method of building Web pages creates class files that set values and contain getters and setters, input and output JSPs, and a Web diagram to display the flow and logic of the Web pages. (Contrast with **Model 1 Web pages**.)

**SWT.** See **Standard Widget Toolkit**.

**system screen.** An IBM i screen for which data description specification (DDS) display file source members are not available. System screen is specific to an application on an IBM i platform that has been WebFaced.

**template.** A template, stored in a .jsp file (for Web projects) or a .java file (for rich client projects), controls the basic layout and style, such as color and font, of the application. It also defines the appearance of areas that are common in your GUI, such as a banner and a navigation area.

**text replacement.** A HATS function used to transform text on a host system into images, HTML code, or other text on a HATS screen transformation,

**theme.** A theme groups a set of common appearance and behavior characteristics for a project. These attributes can be individually modified later.

**transfer.** To copy an application EAR file to the server, typically by FTP.

**transformation.** A transformation stored in a .jsp file (for Web projects) or a .java file (for rich client projects) defines how host components should be extracted and displayed using widgets in your GUI.

**transformation connection.** See **default connection**.

**transformation fragment.** A HATS resource that contains the content with which to replace all occurrences of a pattern in any given transformation.

**Unicode.** A universal character encoding standard that supports the interchange, processing, and display of text that is written in any of the languages of the modern world. It also supports many classical and historical texts in a number of languages. The Unicode standard has a 16-bit international character set defined by ISO 10646.

**user.** Any person, organization, process, device, program, protocol, or system that uses the services of a computing system.

**user list.** A list containing information about accounts (user IDs) that a HATS application can use to access a host or database. User lists contain user IDs, passwords, and descriptions for the accounts.

**UTF-8.** Unicode Transformation Format, 8-bit encoding form, which is designed for ease of use with existing ASCII-based systems.

**Web archive (WAR).** A compressed file format, defined by the Java EE standard, for storing all the resources required to install and run a Web application in a single file.

**Web Express Logon (WEL).** A HATS feature that enables users to log onto several hosts using a set of credentials that are authenticated by a network security layer. (See **network security layer**.)

**Web service.** A self-contained, self-describing modular application that can be published and invoked over a network using standard network protocols.

**Web service, RESTful.** A Web service that uses a stateless architecture and is viewed as a resource rather than a function call. Well-formatted URIs are used to identify the Web service resource, HTTP method protocols are used to do create, retrieve, update, and delete (CRUD) activities, and HTTP header information is used to define the message format.

**Web service, traditional, WSDL-based.** A Web service where typically, XML is used to tag data, SOAP is used to transfer data, WSDL is used for describing the services available, and UDDI is used for listing what services are available.

**WebFacing feature.** The IBM WebFacing Tool for IBM i feature of the HATS Toolkit. The WebFacing feature provides the ability to convert IBM i data description specification (DDS) display file source members into a Web-based user interface for existing 5250 programs.

**WebFaced application.** A Web application produced by the WebFacing feature of the HATS Toolkit.

**WebSphere.** An IBM brand name that encompasses tools for developing e-business applications and middleware for running Web applications. Sometimes used as a short name for WebSphere Application Server, which represents the runtime half of the WebSphere family of products.

**WebSphere Application Server.** Web application server software that runs on a Web server and that can be used to deploy, integrate, run, and manage e-business applications. HATS applications, when exported and transferred to a server, run as WebSphere Application Server applications.

**WEL.** See **Web Express Logon**.

**widget.** A reusable user interface component such as a button, scrollbar, control area, or text edit area, that can receive input from the keyboard or mouse and can communicate with an application or with another widget. HATS applications transform host components into widgets.

**wizard.** An active form of help that guides users through each step of a particular task.

**workbench.** The user interface and integrated development environment (IDE) in Eclipse-based products such as Rational SDP.

**XML.** See **Extensible Markup Language**.

Various Java definitions reprinted with permission from Oracle.

# Index

## Special characters



## Numerics



## A



## B



## C

# D

# E



# F



# G



# H

# I



# J



# K



# L



# M

# N



# O



# P

# R

# S

# T



# U

# V



# W



# X

# Readers' Comments — We'd Like to Hear from You

**IBM Host Access Transformation Services**
**User's and Administrator's Guide**
**Version 9.7**

**Publication No. SC27-5904-03**

We appreciate your comments about this publication. Please comment on specific errors or omissions, accuracy, organization, subject matter, or completeness of this book. The comments you send should pertain to only the information in this manual or product and the way in which the information is presented.

For technical questions and information about products and prices, please contact your IBM branch office, your IBM business partner, or your authorized remarketer.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you. IBM or any other organizations will only use the personal information that you supply to contact you about the issues that you state on this form.

Comments:

Thank you for your support.

Submit your comments using one of these channels:

- Send your comments to the address on the reverse side of this form.
- Send a fax to the following number: 1-800-227-5088 (US and Canada)
- Send your comments via email to: USIB2HPD@VNET.IBM.COM

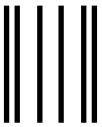
If you would like a response from IBM, please fill in the following information:

| | |
|---|---|
| Name | Address |
| Company or Organization | |
| Phone No. | Email address |

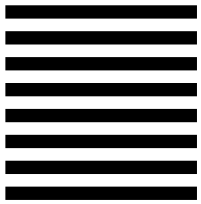
**Readers' Comments — We'd Like to Hear from You**
SC27-5904-03

Cut or Fold
Along Line

Fold and Tape **Please do not staple** Fold and Tape



NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES



# BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

IBM Corporation
Rational Enterprise Modernization UAD
Department 67RA/ Building 503
Research Triangle Park, NC 27709-9990

Fold and Tape **Please do not staple** Fold and Tape



Cut or Fold
Along Line

Printed in USA

SC27-5904-03