Sun Java™ System

Portal Server 6
Desktop Customization Guide

2004Q2

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95054
U.S.A.

Part No: 817-5318-10

Contents

List of Procedures 9

About This Guide 11

Who Should Read This Book 11
What You Need to Know 11
How This Book Is Organized 12
Document Conventions Used in This Guide 14
- Monospaced Font 14
- Bold Monospaced Font 14
- Italicized Font 14
- Square or Straight Brackets 14
- Command-Line Prompts 14
- Variables 15

Where to Find Related Information 15
Where to Find This Guide Online 16

Chapter 1 Introduction to Customizing the Desktop 17
Types of Desktop Customizations 17
- End User Customizations 18
- Developer Customizations 18
- Administrator Customizations 19

What Are the Areas for Customizing the Desktop? 20
Using the Display Profile 21
- Display Profile Document Structure 24
- Modifying the Display Profile 27

Using JavaServer Pages and Template Files 29
- The Desktop and JavaServer Pages 30

The Desktop and Template Files 30
JSP and Template Files Lookup Scenario 31
Deploying JSP or Template Files 33
Creating Customized Organization JSP and Template Files 34
Using the Desktop Tag Library 35
What Is the Sample Portal? 35
Creating a New Desktop 37
Developing the Container 37
Editing the Display Profile 38
Developing and Deploying JSP or Template Files 39
Loading the Display Profile at the Appropriate LDAP Nodes 40
(Optional) Creating a Resource Bundle 41
Accessing the Desktop 41
Debugging the Desktop 41
Creating and Extending a Provider Class 42

Chapter 2 Display Profile Properties 43
Overview of Display Profile Properties 43
Display Profile Properties 44
Display Profile Property Types 45
Display Profile Global Properties 46
Display Profile Container Provider Properties 48
Available and Selected List 49
Common Properties for Table Container 50
Common Properties for Tab Container 53
Other Container Properties 54
Display Profile Leaf Provider Properties 54
Display Profile Properties of Building-Block Providers 54
Display Profile Properties of Service Providers 57
Display Profile Properties of Content Providers 61
Display Profile Common Properties for Leaf Providers 78
Other Leaf Provider Display Profile Properties 80
Display Profile Channel Properties 82

Chapter 3 Understanding the Sample Portal 83
Overview of the Sample Portal 83
The Sample Desktop 84
Guidelines for Using the Sample Portal 85
Sample Portal Installation Directories 87
JSP-Based Desktop 88
JSPTabContainer 88
JSPTableContainer 93

Template-based Desktop 97
Frame-based Desktop 97
FrameTabContainer 97

Chapter 4 Internally Used Containers 103

Chapter 5 Customizing Container Tabs 105
Adding a Tab to JSPTabContainer 105
Creating a Tab Within a Tab 110
Stretching a Tab Across an Entire Container 110
Changing the Tab Image for JSP-based Tab Containers 111
Changing the Color of Tabs 111
Making a Tab the Start Tab 112
Adding a Role-Based Tab 112
Adding a Channel to a User-defined Tab 114

Chapter 6 Customizing Channels 115
Customizing Channel Refresh Times and Container Caching 115
Customizing Window Preference 116
Removing a Button 117
Changing the Channel Layout for a Table Container 120
Removing the Title Bar from a Channel 120
Changing the Channel Border Width 121
Customizing the Channel Border 122

Chapter 7 Customizing Instant Messaging 123
Disabling the User from Editing Instant Messaging Server Information 123
Automatically Closing the Instant Messaging Invite Window 126
Customizing Display of Instant Messaging Contacts 126

Chapter 8 Customizing the Anonymous Desktop 129
Configuring Anonymous Authentication 129
Accessing the Anonymous Desktop 132
Disabling the Initial Identity Server Software Login Page and Always Use Anonymous Log In . . 134
Modifying the Anonymous Banner and Menu Bar 135
Adding the Login Channel to the Anonymous Desktop of a Newly Created Organization 136
Modifying the Default Desktop (Container) for authlessanonymous User 140

Chapter 9 Customizing Authentication 141
Using UNIX Authentication with LoginProvider 141
Configuring LDAP Authentication for UserInfoProvider 142

Chapter 10 Modifying the Desktop Layout 145
Deriving More Desktop Layouts 145
Changing Content Layout to Support Categorizing the Available and Selected Lists 147
Customizing Existing JSPs 147
Writing a New Content Channel 147
Adding New Layouts 148
Changing the Desktop Column Layout 148

Chapter 11 Branding the Desktop 151
Changing the HTML Title (Title That Appears in the Browser) 151
Changing the Logo (Image) in the Banner Header 151
Changing the Header and Footer of the Theme, Content, and Layout Pages 152

Chapter 12 Changing Desktop Colors 155
Changing Desktop Colors 155
Changing the Default Color Scheme for an Organization 157

Chapter 13 Customizing the Global Themes 159
What is a Theme? 159
Customization Overview 159
GlobalThemes Display Profile Definition 160
Theme Properties 167
Glossary of Terms 170
Adding and Customizing Global Themes 170

Chapter 14 Customizing the Service Providers 175
Overview of Customizing the Service Providers 175
Overview of Customizing the Search Provider 175
Overview of Customizing the Discussion Provider 177
Tips for Customizing the Service Providers 179
Debugging the Service Providers 179
Location of JavaServer Pages 180
Modifying JavaServer Pages 180
Accessing Channels Directly 180
Customizing the Search Provider 181
Customizing the Discussion Channels 187
Customizing DiscussionLite Channel 187
Customizing Discussions Channel 189

Chapter 15 Customizing the Desktop End-User Online Help 191
Overview of the Desktop End-User Online Help 191

Location of the Desktop End-User Online Help HTML Files 193
Modifying the Desktop End-User Online Help HTML files 195
Editing An Existing Help File 195
Creating a New Help File 195

Appendix A Desktop Template Files 199
Overview of Desktop Templates 199
Desktop Templates in the default Directory 200
AddressBookProvider 200
AppProvider 201
BookmarkProvider 201
CalendarProvider 202
error 204
LoginProvider 205
LotusNotesAddressBookProvider 205
LotusNotesCalendarProvider 206
LotusNotesMailProvider 208
MailCheckProvider 209
MailProvider 209
MSExchangeAddressBookProvider 210
MSExchangeCalendarProvider 211
MSExchangeMailProvider 213
TemplateTabContainerProvider 214
TemplateTableContainerProvider 215
UserInfoProvider 216
default 217
Desktop Templates in the sampleportal Directory 218
MyFrontPageTemplatePanelContainer 218
PredefinedFrontPageTemplatePanelContainerProvider 219
PredefinedSamplesTemplatePanelContainerProvider 220
SamplesTemplatePanelContainer 222
TemplateTabContainerProvider 222
TemplateTabCustomTableContainerProvider 222
TemplateTableContainer 223
ToolsTemplatePanelContainer 223
sampleportal 224
Miscellaneous Template Information 224
Dynamic Template Reloading 225

Appendix B Desktop Tag Reference 227
Overview of the Tags 227
How the Desktop Template Tags Work 227
Kinds of Tags Used in the Desktop 228

Provider-Specific Tags 228
AddressBookProvider 229
AppProvider 229
BookmarkProvider 229
CalendarProvider 231
LoginProvider 240
MailCheckProvider 240
MailProvider 241
TemplateTabContainerProvider 242
TemplateTableContainerProvider 243
UserInfoProvider 244
Common Tags 245

Appendix C JavaServer Pages Reference 251
Anonymous Desktop JSPs 251
FrameTabContainer JSPs 252
JSPDynamicSingleContainer JSPs 252
JSPTabContainer JSPs 252
JSPTableContainer JSPs 253
PredefinedFrontPageFramePanelContainerProvider JSPs 254
PredefinedFrontPageTabPanelContainerProvider JSPs 255
PredefinedSamplesFramePanelContainerProvider JSPs 256
PredefinedSamplesTabPanelContainerProvider JSPs 257
Desktop JSPs in the default Directory 257
DiscussionLite JSPs 258
Discussions JSP 259
DiscussionsProvider JSPs 261
DummyChannel JSPs 262
IMProvider JSPs 263
JSPContentContainer JSPs 263
JSPDynamicSingleContainer 264
JSPEditContainer JSPs 264
JSPFrameCustomTableContainerProvider JSPs 264
JSPLayoutContainer JSPs 265
JSPProvider JSPs 266
JSPSingleContainerProvider JSPs 266
JSPTabContainerProvider JSPs 267
JSPTabCustomTableContainerProvider JSPs 268
JSPTableContainerProvider JSPs 269
Miscellaneous JSPs 270
SampleSimpleWebService JSPs 271
SampleSimpleWebServiceConfigurable JSPs 272
Search JSPs 272

SearchProvider JSPs 275
SimpleWebServiceConfigurableProvider JSPs 277
SimpleWebServiceProvider JSPs 278
Subscriptions JSPs 278
SubscriptionsProvider JSPs 279
TabJSPEditContainer JSPs 280
JSPs in the sampleportal Directory 280
FrameTabContainer JSPs 281
JSPContentContainer JSPs 282
JSPCreateChannelContainer JSPs 282
JSPCustomThemeContainer JSPs 283
JSPDynamicSingleContainer JSPs 283
JSPEditContainer JSPs 284
JSPFrameCustomTableContainerProvider JSPs 284
JSPLayoutContainer JSPs 285
JSPPopupContainer JSPs 285
JSPPresetThemeContainer JSPs 285
JSPSingleContainer JSPs 286
JSPTabContainer JSPs 286
JSPTabCustomTableContainerProvider JSPs 287
JSPTableContainerProvider JSPs 287
Miscellaneous JSPs 288
PredefinedFrontPageFramePanelContainerProvider 290
PredefinedFrontPageTabPanelContainerProvider 291
PredefinedSamplesFramePanelContainerProvider 293
PredefinedSamplesTabPanelContainerProvider 294
SampleJSP JSPs 295
SampleSimpleWebService JSPs 296
Miscellaneous JSP Information 296
Performing Redirects 296
JSP vs. Theme Color 297
Recompiling JSPs 297
JavaServer Page Caching Information 298
Debugging JSPs 298

Appendix D JavaServer Pages Tag Library Reference 301
Tag Library Overview 301
Overview 302
Desktop Tag Library Hierarchy 303
Using Tags in JSPs 304
Using the Desktop Tag Library in Your Application 307
Example Tab Page with Selected Channels 309
Tag Overview 310

Attributes and Return Values 310
Tag Library Exceptions 311
Tag Reference 312
Context Setup Tags 312
Validator Tags 313
Normal Tags 313
Search Tags 326
Instant Messaging Tags 334

Glossary 337

Index 339

List of Procedures

To Create Customized Organization JSP and Template Files 34
To Access the Desktop 41
To change the Desktop type 86
To Add a Tab to JSPTabContainer 105
To Stretch a Tab Across an Entire Container 110
To Change the Tab Image for JSP-based Tab Containers 111
To Change the Color of Tabs 111
To make the tab the start tab 112
To Add a Role-based Tab 112
To Customize the Channel Window Preference 116
To Remove a Button From All Channels in a Container 117
To Remove a Button From a Single Channel 118
To Change the Channel Layout for a Table Container 120
To Remove the Title Bar from a Channel 120
To Change the Border Width for all Channels in a Container 121
To Customize Channel Borders to Have Bevelled Edges, Shadows, Curved Corners, and So On . 122
To Enable Anonymous Log In 130
To Disable Anonymous Log In 130
To Enable Authentication-less (authlessanonymous) Log In 130
To Disable Authentication-less (authlessanonymous) Log In 131
To Access the Anonymous Desktop through the Identity Server Host Name (obj.conf File) 132
To Access the Anonymous Desktop through the Portal Server Host Name (index.html File) 133
To always use Anonymous Log In 134
To Change the Banner for the Anonymous Desktop 135
To Add the Login Channel to the Anonymous Desktop of a Newly Created Organization 136
To Change the Default Channel Name for Authlessanonymous User 140
To Use UNIX Authentication with LoginProvider 141

To Enable End User Password Maintenance for LDAP Authentication 142
To Change the Desktop Column Layout from the Command Line 148
To Change the Desktop Column Layout from the Administration Console 149
To Modify Column Widths Directly (Using Scriptlets) 149
To Change the Logo (image) in the Banner 152
To Change the Header and Footer of the Theme, Content, and Layout Pages 153
To Change the Desktop Colors 155
To Add a Theme to the Sample Portal 170
To Customize the Current Themes 171
To Change the Text 171
To Change the Sample Anonymous Desktop Theme 172
To Modify the Default Search Server 181
To Add last-modified to the Search Result Display 182
To Remove content-length from Search Results 183
To Display the Total Number of Documents in the Search Result Status Message 183
To Remove author from the Advanced Search Interface 184
To Add a New Field to Advanced Search 185
To Customize the DiscussionLite Channel Link Display Window 187
To Display DiscussionLite on the Front tab 188
To Display Additional Fields in the List View of Discussions 189
To Modify the Sort Order in List All Discussions Page 189
To Modify viewHits in View Discussion Page 190
To Inherit Classification and readACL 190
To Control Access to Discussions 190
To Create a New Online Help File and to Define the `helpURL` Value Manually 195
To Change the Template Reload Interval 225

About This Guide

This guide explains how to customize the Sun Java™ System Portal Server 6 software Desktop. The Portal Server software provides a platform to create portals for your organization's integrated data, knowledge management, and applications. The Portal Server software platform offers a complete infrastructure solution for building and deploying all types of portals, including business-to-business, business-to-employee, and business-to-consumer.

This preface includes the following sections:

- Who Should Read This Book
- What You Need to Know
- How This Book Is Organized
- Document Conventions Used in This Guide
- Where to Find Related Information
- Where to Find This Guide Online

Who Should Read This Book

You should read this guide if you are responsible for customizing the Portal Server software Desktop at your site.

What You Need to Know

Before you customize the Portal Server software, you must be familiar with the following concepts:

- Basic Solaris™ administrative procedures
- HTML
- JavaServer Pages™
- LDAP
- XML

You must also be familiar with the following software configuration, administration, and customization tasks:

- Sun Java™ System Identity Server software
- Sun Java™ System Directory Server software
- Sun Java™ System Web Server software
- Sun Java™ System Application Server software

How This Book Is Organized

This book contains the following chapters and appendices:

About This Guide (this chapter)

Chapter 1, "Introduction to Customizing the Desktop" This chapter describes the Portal Server software Desktop, the different kinds of customizations, who would make those customizations, and the actual Desktop files that you customize.

Chapter 2, "Display Profile Properties" This chapter supplies information on the display profile definitions and properties that ship with the Portal Server software.

Chapter 3, "Understanding the Sample Portal" This chapter describes the sample portal that you can choose to install on your system during the Portal Server software installation.

Chapter 4, "Internally Used Containers" This chapter describes the contained containers used by the Portal Server software sample Desktop.

Chapter 5, "Customizing Container Tabs" This chapter provides a variety of tasks to customize the container tabs.

Chapter 6, "Customizing Channels" This chapter describes how to customize channels.

Chapter 7, “Customizing Instant Messaging” This chapter describes a variety of tasks for customizing the Instant Messaging provider.

Chapter 8, “Customizing the Anonymous Desktop” This chapter describes customizations you can make for the anonymous Desktop.

Chapter 9, “Customizing Authentication” This chapter contains instructions for customizing authentication for LoginProvider and UserInfoProvider.

Chapter 10, “Modifying the Desktop Layout” This chapter describes how to modify the channel arrangement in the Desktop.

Chapter 11, “Branding the Desktop” This chapter describes how to brand the Desktop with your site’s logo and name.

Chapter 12, “Changing Desktop Colors” This chapter describes how to change the color for various Desktop components.

Chapter 13, “Customizing the Global Themes” This chapter describes the Global themes and some common customizations.

Chapter 14, “Customizing the Service Providers” This chapter describes common customization tasks for modifying the service providers in the Portal Server software.

Chapter 15, “Customizing the Desktop End-User Online Help” This chapter supplies information on customizing the Portal Server software Desktop online help.

Appendix A, “Desktop Template Files” This appendix describes the Desktop templates files.

Appendix B, “Desktop Tag Reference” This appendix serves as a reference for the tags used by the Desktop template files (discussed in Appendix A).

Appendix C, “JavaServer Pages Reference” This appendix describes the various JavaServer Pages used by the Portal Server software.

Appendix D, “JavaServer Pages Tag Library Reference” This appendix describes the Desktop custom JSP tag library and serves as a reference for the tags available within the library.

Document Conventions Used in This Guide

Monospaced Font

Monospaced font is used for any text that appears on the computer screen or text that you should type. It is also used for file names, distinguished names, functions, and examples.

Bold Monospaced Font

Bold monospaced font is used to represent text within a code example that you should type.

Italicized Font

An italicized font is used to represent text that you enter using information that is unique to your installation (for example, variables). It is used for server paths and names and account IDs.

Square or Straight Brackets

Square (or straight) brackets [] are used to enclose optional parameters. For example, in the Portal Server software documentation, you will see the usage for the `dpadmin` command described as follows:

```
dpadmin list|modify|add|remove [command-specific options]
```

The presence of [command-specific] indicates that there are optional parameters that may be added to the `dpadmin` command.

Command-Line Prompts

Command-line prompts (for example, % for a C-Shell, or $ for a Korn or Bourne shell) are not displayed in the examples. Depending on which operating system environment you are using, you will see a variety of different command-line prompts. However, you should enter the command as it appears in the document unless specifically noted otherwise.

NOTE The procedures in this book use the `dpadmin` command-line utility instead of the Identity Server software administration console to administer the display profile. See the *Portal Server Administration Guide* for information on using the administration console to administer the display profile.

Variables

The following is a two column table that describes the common variables used in this document. The first column lists the variables, and the second column provides a description of how the variables are used.

portal-server-install-root	The Portal Server software installation directory. For example, `/opt`.
web-server-install-root	For example: • Web Server `/opt/SUNWwbsvr` • Application Server `/opt/SUNWappserver7` • BEA WebLogic Server 6.1 `/opt/bea/wlserver6.1` • IBM WebSphere Application Server `/opt/WebSphere/AppServer`
directory-server-install-root	The Directory Server software installation directory. For example, `/usr/sunone/servers`.
identity-server-install-root	The Identity Server software installation directory. For example, `/opt/IS6.1`.
UserID	User identification. For example, `root` or `nobody`.

Where to Find Related Information

In addition to this guide, the Portal Server software comes with supplementary information for administrators as well as documentation for developers. Use the following URL to see all the Portal Server software documentation:

`http://docs.sun.com/prod/s1portalsrv`

Listed below are the additional documents that are available:

- *Portal Server Administration Guide*
- *Portal Server, Secure Remote Access Administration Guide*
- *Portal Server Developer's Guide*
- *Portal Server Deployment Planning Guide*
- *Portal Server Migration Guide*
- *Portal Server Release Notes*

Where to Find This Guide Online

You can find the *Portal Server Desktop Customization Guide* online in PDF and HTML formats. This book can be found at the following URL:

`http://docs.sun.com/coll/PortalServer_04Q2`

Chapter 1

Introduction to Customizing the Desktop

This chapter provides an introduction to customizing the Sun Java System Portal Server software Desktop. It describes the different kinds of customizations and who should make those customizations. This chapter also provides an overview of the display profile, sample Desktops included with the product, and how you create and deploy a new Desktop and provider.

This chapter contains the following sections:

- Types of Desktop Customizations
- What Are the Areas for Customizing the Desktop?
- Using the Display Profile
- Using JavaServer Pages and Template Files
- What Is the Sample Portal?
- Creating and Extending a Provider Class

Types of Desktop Customizations

The Desktop can be customized by end users, administrators, and developers. Though this guide covers only administrator customizations, it includes an overview of end user and developer customizations, and where to go for more information on those customizations. This guide refers to administrator customizations such as changes and modifications made to the Desktop that involve modifications to the display profile, JavaServer Pages™ and template files, search provider, and online help.

End User Customizations

End users can customize the Desktop in the following ways:

- Setting the channel time out
- Selecting column layout from the Layout page
- Moving channels up and down, as well as side to side
- Arranging channels by width
- Resizing the channel window
- Adding and removing certain channels from the Content page
- Customizing channels by using the channel Edit page
- Selecting theme from a set of preset themes or customize the theme by changing color scheme and font type for the Desktop channels
- Creating, removing, and editing tabs

Users customize channels by using the Edit Channel icon for a particular channel (as long as the administrator has made it available). Users customize the look and feel of the Desktop through the Themes page. See the Portal Server software End User Desktop Online Help for more information.

Users can also configure:

- Time zone, language, mail server, and password by using the User Info channel Edit page.
- Personal and mail information by using the Sun Java System Identity Server software administration console. The URL to use is:

  `http://`*`hostname`*`:`*`port`*`/amconsole`

Developer Customizations

The Portal Server software developers can customize the Desktop by creating:

- Provider classes
- Desktop templates
- Tag libraries
- JavaServer Pages

Developers can also use the Provider Application Programming Interface (PAPI) and the Search service APIs to extend the Portal Server software. See the *Portal Server Developer's Guide* for more information.

In addition, developers can create new portal services to integrate applications and enable Single Sign-on across multiple applications, as well as implement authentication modules, using the Identity Server software APIs. See the *Identity Server Programmer's Guide* for more information.

Administrator Customizations

The Portal Server software administrators can customize the Desktop by:

- Using supplied providers to define additional content channels.
- Creating and customizing the display profile, which involves creating or modifying provider, channel, and container channel objects. When you modify the display profile, you use the appropriate XML tag definitions for providers, channels, and container channels.
- Create new preset themes in the display profile.
- Using supplied JavaServer Pages and template files to modify the user interface.
- Customizing the search provider.
- Customizing the Desktop end user online help.

By performing these customizations, you can arrive at:

- A site-specific look and feel of the Desktop: whether it uses tabs or frames, what channels are available to users and how they are situated out on the Desktop, what applications are available to end users, what kind of online help is available, and so on.
- Different Desktops for different LDAP roles or organizations.
- Desktop behavior based on user roles.

This guide describes these administrator customizations.

What Are the Areas for Customizing the Desktop?

In general, the Portal Server software documentation divides Desktop interface customization into three areas:

Authentication screens You can modify the look and feel of the HTML authentication templates, including images, HTML structure, and color.

NOTE Authentication screens are provided by the Identity Server software. This guide does not describe how to customize the authentication screens. See the *Identity Server Programmer's Guide* for that information.

Desktop You can modify the look and feel of the Desktop, customize the top container used by the organization, control Desktop themes, and so on. This chapter describes the Desktop customization tasks.

Files for dynamic content for providers, channels, and containers are located under the `/etc/opt/SUNWps/desktop` directory; that is, the Base Desktop templates are in `/etc/opt/SUNWps/desktop/default` and sample portal templates are in `/etc/opt/SUNWps/desktop/sampleportal` directories. For more information, see "What Is the Sample Portal?" on page 35.

Some other static resources are located in the *web-container*`/portal` directory. This includes images used on the Desktop, such as channel control buttons, and style sheets.

NOTE References to the locations and contents of files in the `web-apps` directory are for information only. They do not represent the definition of an interface that you can depend on for any future release.

Search channel You can modify the default Search server and add or remove fields from the Advanced search interface. See Chapter 14, "Customizing the Service Providers" for more information.

Using the Display Profile

Much of your work in customizing the Portal Server software involves creating or editing the display profile to provide the kind of Desktop you want for your site. The display profile is an XML document that defines the Desktop structure and content. The display profile Document Type Definition file (DTD) defines valid syntax for the display profile XML documents. See `/etc/opt/SUNWps/dtd/psdp.dtd` for more information.

The hierarchical structuring of the display profile document does not define the visual layering of channel on the portal Desktop. The display profile exists only to provide property values for channels on the Desktop.

The display profile contains definitions that enable you to construct the Desktop. These definitions include providers, channels, containers, and properties. Some of these definitions create the Desktop containers—the frames, tables, and tabs that arrange the content of the Desktop—and others create channels for the Desktop via the respective providers. A display profile provider definition is a template for building channels based on that provider.

NOTE A provider is a Java class responsible for converting the content in a file, or the output of an application or service into the proper format for a channel. A number of providers are shipped with the Portal Server software. As the desktop is imaged, each provider is queried in turn for the content of its associated channel. Some providers are capable of generating multiple channels based upon their configuration.

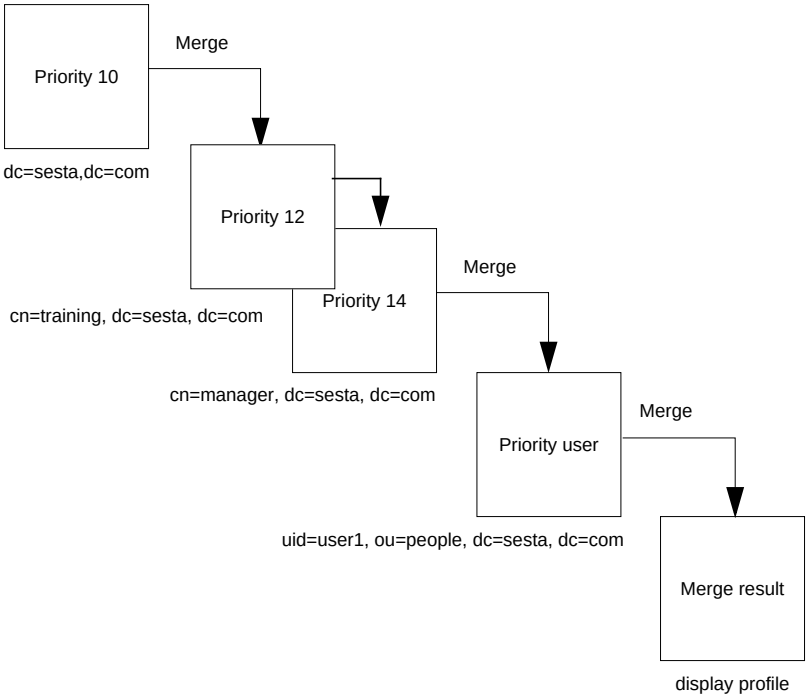
Using the display profile mechanism, you can create a new portal Desktop or modify the sample portal provided with the product. See “What Is the Sample Portal?” on page 35 for more information.

Display profile documents are stored in their entirety as a single attribute in the Identity Server software services layer. Potentially, the Portal Server software could store a display profile document for a user’s organization or sub-organization, each role the user belongs to, and the user. That is, for the different LDAP nodes (base DN, org DN, role DNs, and uid DN), you can store a display profile document. There is also a global display profile document.

A user’s display profile document set is made up from the non-empty documents stored at the user’s organization, various sub-organizations, any roles, and the user LDAP nodes. This display profile document set is “merged” at runtime to form a single configuration for the user’s Desktop.

For example, Figure 1-1 shows a sample DIT with each level having its own display profile document. At the top of the tree is the global display profile. Next is the base DN, `dc=sesta,dc=com`. It has a two role DNs, training and manager. The tree ends at the uid DN, `user1`. Each node in the DIT has its own display profile document. The resultant display profile document is produced through a merge of all the display profile documents, based on their priorities. This merge result is presented to the user at login. Merge semantics control how display profile documents are combined. These semantics include replace, remove, and fuse. Display profile objects can also be locked. During the merge, a higher priority document can lock a display profile object to prevent a lower priority document from altering it. See the *Portal Server Administration Guide* for more information.

Figure 1-1 Example of Merge Process



The display profile documents themselves consist of display profile objects. The display profile DTD defines the XML tags that represent the allowable display profile objects. These are:

Provider object The provider is a programmatic entity responsible for fetching and displaying content in a channel. The XML tag for defining a provider object is `<Provider name="providerName" class="providerClass">`.

The display profile document for a provider provides default property values for all properties that channels based on that provider will use. It provides the class name to use to create the provider object (see "Provider object" on page 23 for more information) at runtime.

Channel object A channel is a unit of content, usually (but not necessarily) arranged in rows and columns. You can also have channels of channels, that is, container channels. The XML tag for a channel is `<Channel name="channelName" provider="providerName">` and the XML tag for a container channel is `<Container name="containerName" provider="providerName">`.

A display profile channel definition defines the configuration for a Desktop channel. Specifically, it defines the provider Java object that is used to create a running channel instance (indirectly, by naming the display profile provider definition that names the Java class file), and defines the properties used by the channel instance. Only property values that differ from the defaults defined in the associated display profile provider definition should be defined in the display profile channel definition's properties. See also, "Channel object" on page 23.

Property object A property value that can be specified for a channel. Individual properties are grouped within the `<Properties> </Properties>` tags inside a channel definition. See also, "Property object" on page 23.

Like display profile objects are grouped within their appropriate XML tag pairs. That is, providers are grouped within `<Providers></Providers>` tags, channels within `<Channels></Channels>` tags, properties within `<Properties></Properties>` tags.

Because you can have multiple display profile documents defined at different LDAP nodes, at runtime, the system merges these multiple display profile documents to deliver a particular Desktop to the user. This process of merging display profile documents affects the final display profile object values.

For a complete discussion of the display profile, how the merging works, and a description of the display profile DTD syntax, see the *Portal Server Administration Guide*.

Display Profile Document Structure

The display profile format is intended to define the Desktop's display configuration by defining provider and channel objects and their properties. Thus, a display profile is made up of some number of display profile objects. The display profile objects map directly to the XML tag that defines them. For example, the `<Channel name=>` XML tag defines a channel object.

In general, the document structure of a display profile document resembles the following:

```
<DisplayProfile>
    <Properties>...global properties...</Properties>
    <Channels>...channel definitions...</Channels>
    <Providers>...provider definitions...</Providers>
</DisplayProfile>
```

The Channels, Providers, and Properties display profile objects are used to group other display profile objects. These grouping objects are loosely referred to as "bags." These bags add more structure to the DP XML documents.

The following sections describe the display profile objects in more detail. For a complete discussion of the display profile, see the *Portal Server Administration Guide*.

Channel Object

A channel object represents a single display element. The objects contained by a channel object can be thought of as properties for the channel. The channel definition includes a symbolic reference to the provider. You only need to include channel-specific properties when the provider defaults are not appropriate.

Code Example 1-1 Example Channel Object XML Syntax

```
<Channel name="SampleXML" provider="XMLProvider">
    <Properties>
        <String name="refreshTime" value="600"/>
        <String name="title" value="XML Test Channel"/>
        <String name="description" value="This is a test of the XML Provider
system"/>
        <String name="url"
value="file:///etc/opt/SUNWps/desktop/default/SampleXML/getQuotes.xml"/>
```

Code Example 1-1 Example Channel Object XML Syntax *(Continued)*

```
        <String name="xslFileName"
value="/etc/opt/SUNWps/desktop/default/SampleXML/html_stockquote.xsl"/>
    </Properties>
</Channel>
```

Container Object

A container object is identical to a channel object, except that it primarily generates its content by aggregating the content of other (its child) channels. A container object allows for available and selected channel lists and can contain leaf channel definitions. A leaf channel is typically aggregated on a page with other channels and generates its own content. A container channel primarily generates content by aggregating the content of one or more leaf channels. Both leaf and container providers are building blocks in that they can be extended (through their public interfaces) to create new or custom providers.

The leaf building-block providers include:

- JSPProvider
- XMLProvider
- URLScraperProvider

The container building-block providers include:

- Tab
- Table
- Single

See the *Portal Server Developer's Guide* for more information on the leaf and container building-block providers.

Code Example 1-2 Example Container Object XML Syntax

```
<Container name="JSPTableContainer" provider="JSPTableContainerProvider">
    <Properties>
        <String name="title" value="JSP Based Table Container Channel"/>
        <String name="contentPage" value="toptable.jsp"/>
        <String name="description" value="This is a test for front table
containers"/>
        <String name="Desktop-fontFace1" value="Sans-serif"/>
        <Collection name="categories">
```

Code Example 1-2 Example Container Object XML Syntax *(Continued)*

```
            <String value="Personal Channels"/>
            <String value="Sample Channels"/>
            <String value="News Channels"/>
        </Collection>
        ...
    </Properties>

    <Available>
        <Reference value="UserInfo"/>
        <Reference value="MailCheck"/>
        ...
    </Available>

    <Selected>
        <Reference value="UserInfo"/>
        <Reference value="MailCheck"/>
        ...
    </Selected>

    <Channels> ...leaf definitions go here...</Channels>

</Container>
```

Provider Object

A provider object is a pointer to the display profile provider definition. The provider is a contract between ProviderContext and channel instance (provider object).

The display profile provider definition contains the information necessary for a client of the display profile to construct the provider object, namely, the Java™ class name.

The provider definition sets default property values for all channels that point to this provider. Channel-specific properties are only necessary when the provider defaults are not appropriate. The provider display profile object should contain default values for all properties that are used in the provider Java object. For example, if the provider Java code contains:

```
getStringProperty("color")
```

the provider display profile object should have a default value for color.

Code Example 1-3 Example Provider Object XML Syntax

```
<Provider name="XMLProvider"
class="com.sun.portal.providers.xml.XMLProvider">
    <Properties>
        <String name="title" value="*** XML Provider ***"/>
        <String name="description" value="*** DESCRIPTION ***"/>
        <String name="width" value="thick"/>
        <String name=”color” value=”blue”/>
        <String name="refreshTime" value="0" advanced="true"/>
        <Boolean name="isEditable" value="false" advanced="true"/>
        <String name="helpURL" value="desktop/xmlchann.htm"
advanced="true"/>
        <String name="fontFace1" value="Sans-serif"/>
        <String name="productName" value="Sun Java System Portal Server"/>
        <String name="url"
value="file:///etc/opt/SUNWps/desktop/default/xml/getQuotes.xml"/>
        <String name="xslFileName" value="html_stockquote.xsl"/>
        <Integer name="timeout" value="100"/>
        <String name="inputEncoding" value="iso-8859-1"/>
        <String name="urlScraperRulesetID" value="default_ruleset"/>
        <Boolean name="cookiesToForwardAll" value="true"/>
        <Collection name="cookiesToForwardList">
        </Collection>
    </Properties>
</Provider>
```

Modifying the Display Profile

You can modify display profile objects by performing one of the following:

- Using the Identity Server software administration console Channel and Container Management link.
- Using the Identity Server software administration console Edit XML link.
- Manually editing an existing display profile document and then loading it at the appropriate LDAP node by using the `dpadmin modify` command.
- Running the `dpadmin` command with the specified XML text changes, on standard input. When adding a new object, use the `add` subcommand. When modifying an existing object, use the `modify` subcommand.
- Creating a new display profile document from scratch and then loading it at the appropriate LDAP node by using the `dpadmin modify` command.

For information on using the dpadmin command, see the *Portal Server Administration Guide*. Use the following guidelines when running the dpadmin command to update the display profile:

- Make sure no other administrator is currently using the Identity Server software administration console or dpadmin command to make display profile modifications. Such a situation could cause changes to be lost, if you are running dpadmin modify, as there is no locking mechanism to prevent dpadmin and the administration console from accessing the display profile at the same time.
- The preferred sequence when using dpadmin is to put your modifications into a file as an XML "fragment" then run the dpadmin command with the appropriate subcommand. For example,

  portal-server-install-root/SUNWps/bin/dpadmin add -u *uid* -w *password* -d *distinguishedname* *fragmentfile*.xml

  In this example, newtheme.xml is a file containing the XML "fragment" to be added to the display profile.
- If you edit a display profile document directly, first use the dpadmin command with the list subcommand to obtain the latest contents of the display profile, make your edits, then run the dpadmin command with the modify subcommand. For example,

  portal-server-install-root/SUNWps/bin/dpadmin list -u *uid* -w *password* -d *distinguishedname* > dp-org.xml

  Edit the dp-org.xml file.

  portal-server-install-root/SUNWps/bin/dpadmin modify -u *uid* -w *password* -d *distinguishedname* dp-org.xml

CAUTION Between the time you run the dpadmin list and dpadmin modify commands, do not change the display profile document in the LDAP server in any way (by using the administration console, dpadmin, or ldapmodify commands). Otherwise, those changes will be overwritten by the latest dpadmin modify command.

Using JavaServer Pages and Template Files

To generate the rendered Desktop user interface (what the industry refers to as the "presentation"), The Portal Server software makes use of either JavaServer Pages (JSP™) or template files. JSPs are preferred because they enable a much easier customization process without having to change the provider Java classes. JSPs also provide a way to enable a strict separation of business and presentation logic. Specifically, this means having the business logic in the provider classes and presentation logic in JSPs.

NOTE In general, a three-tier architecture consists of presentation logic, business logic, and the data. Tag libraries or Enterprise JavaBeans™ provide the business logic, a database contains the data, and JavaServer Pages (JSPs) or templates provide the presentation logic. However, this view is based on a "small" system where the entire system is contained in one server, or perhaps only the data is on another server.

The Portal Server software takes the "larger" system view, where all of the product is presentation. The business logic resides in some back end resource server that a content provider accesses. The data is on yet another back end server. Because all of the Portal Server software is presentation, there really is no business logic in the product.

The default set of JSPs and template files are installed in `/etc/opt/SUNWps/desktop/default` directory. The sample portal JSPs and template files are installed in `/etc/opt/SUNWps/desktop/sampleportal` and `/etc/opt/SUNWps/desktop/anonymous` directories. The Desktop Type attribute in the Desktop attributes page of the Identity Server software administration console specifies from what subdirectory to retrieve either the JSP or template files for the Desktop. For more information, see "JSP and Template Files Lookup Scenario" on page 31.

NOTE How the JSPs and Template files are referenced by the providers is provider-specific. Some providers specify the file in the display profile, other providers specify fixed names.

For example, for JSPProvider, there are display profile properties such as contentPage, editPage, and so on, that reference Desktop files under the `/etc/opt/SUNWps/desktop` directory. For other providers, such as BookmarkProvider, the name of the template file is fixed, for example, `display.template` and that name is mentioned in the display profile document for that provider.

The Desktop and JavaServer Pages

The JSPProvider class reads in the JSP, compiles it, and runs it to produce the channel content.

The JSPProvider class reads at most three JSPs, one for content, one for the Edit page, and one to process the form submission from the Edit page. All other JSPs used in a JSP-based channel are referenced from one of those JSPs, either by an include or a forward statement.

A simple JSP-based channel can have just one JSP. Multiple JSPs are useful when a single part of the Desktop has to be replicated in several places. For example, consider a channel that has several display modes based on links clicked in that channel. Further, assume that the channel has a banner that must be displayed in all modes but one. You could construct a JSP to reference a `banner.jsp` file that captures common formatting that is used in multiple branches of the logic. If you didn't use this method, you would need to duplicate the content from the `banner.jsp` file, which is more difficult to maintain if that content needs to be changed. See Appendix C, "JavaServer Pages Reference" for more information.

The Desktop and Template Files

Providers that use template files hardcode their names and the template file names are not configurable in the Display Profile. For Providers that are based on JSP provider, the names of the JSP used by provider can be administratively changed as it is a property of the channel.

Both JSP and Desktop templates serve the purpose of separating business and presentation logic in the Portal Server. JSP is an accepted standard and is widely employed in many web-based applications. Desktop templates pre-date the emergence of JSP. JSP has many advantages over Desktop templates.

It is highly recommended that new Portal Server providers be based on JSP and the Portal Server JSPProvider. However, there may be cases where the simplicity of Desktop templates provides an advantage. Desktop templates are fully supported in Portal Server. Many pieces of the product, such as the communications channels, continue to use them.

JSP and Template Files Lookup Scenario

The Portal Server software uses a specific lookup scenario to find the JSP and template files it needs for containers.

desktoptype_locale/channelname/clientPath

desktoptype_locale/provider/clientPath

desktoptype_locale/channelname

desktoptype_locale/provider

desktoptype_locale/clientPath

desktoptype_locale

desktoptype/channelname/clientPath

desktoptype/`provider`/*clientPath*

desktoptype/channelname

desktoptype/provider

desktoptype/clientPath

desktoptype

`default`_*locale/channelname/clientPath*

`default`_*locale/provider/clientPath*

`default`_*locale/channelname*

`default`_*locale/provider*

`default`_*locale/clientPath*

`default`_*locale*

`default`/*channelname/clientPath*

`default/`*provider*`/`*clientPath*

`default/`*channelname*

`default/`*provider*

`default/`*clientPath*

`default`

templateroot

If there is no `clientPath` specified, then the directory search order is as follows:

desktoptype_locale/channelname

desktoptype_locale/provider

desktoptype_locale

desktoptype/channelname

desktoptype/provider

desktoptype

`default`*_locale/channelname*

`default`*_locale/provider*

`default`*_locale*

`default/`*channelname*

`default/`*provider*

`default`

templateroot

The lookup scenario relies on the following parameters.

- Desktop type, for example `default` (set in the Identity Server software administration console). Note that desktop type is now a comma separated string list and so the look up will be based on the desktop type(s) that are defined in the desktop type attribute.
- Preferred Locale is the user's locale, for example, `en_US` (set by users through the Identity Server software administration console in the "User" setting)
- Client path is an optional file-path containing client-specific templates; for example, `html` (set through the Identity Server software administration console Client Detection service)
- Channel name is the name of the channel; for example, newSingleContainer (set in the display profile)

- Provider name is the provider name; for example, JSPSingleContainerProvider (set in the display profile)
- Template root as defined in the `desktopconfig.properties` file. The root of the search directory (default value of `/etc/opt/SUNWps/desktop/`) can be changed by modifying the `templateBaseDir` property in the `desktopconfig.properties` file.

Use this order to decide the final location of your own JSPs.

Deploying JSP or Template Files

When you create a container, you need to create a new subdirectory for your newly created container in the `/etc/opt/SUNWps/desktop/`*desktopType* directory. That is, the newly created container should be placed based on what the `desktopType` is. If sampleportal is installed, then the `desktopType` is, by default, `sampleportal`; so, create a new directory for the container under sampleportal so that any JSP and template that is being added can adopt the same look and feel as defined in the sampleportal. If sampleportal is not installed, and if you have set up a custom `desktopType`, for example, `foo`, then the new container directory must be created directory under `foo`.

Either copy the modified JSP or template files here, or place your newly created files here. If you use a sample container without changing any content or file names, you do not need to create a new subdirectory nor copy any files there. (In the example that follows, a new subdirectory is needed, because a new container is created.)

For example, let's say you create a new container called newSingleContainer whose display profile definition is the following:

```
<Container name="newSingleContainer" provider="JSPSingleContainer">
    <Properties>
        <String name="helpURL" value="desktop/newSingle.html"/>
        <String name="title" value="A new single container"/>
        <String name="contentPage" value="newsinglecontent.jsp"/>
        <Boolean name="isEditable" value="true"/>
        <String name="editType" value="edit_subset"/>
    </Properties>
    <Available/>
    <Selected/>
    <Channels/>
</Container>
```

Because the file specified for the contentPage property is different from the `contentPage` value for the provider definition, you need to create a new directory under the `/etc/opt/SUNWps/desktop/default` directory called `newSingleContainer`. You then only need to copy the `newsinglecontent.jsp` file to this new directory. The system is able to locate all other JSPs referenced by the JSPSingleContainer provider.

Creating Customized Organization JSP and Template Files

If desired, rather than customizing the sample portal JSP and template files directly, you can create a separate directory for your organization's customized files, and perform customizations on those files. This preserves the initially installed portal JSP and template files.

➤ **To Create Customized Organization JSP and Template Files**

1. Change directories to the Desktop JSP or template directory.

   For example,

   ```
   cd /etc/opt/SUNWps/desktop
   ```

2. Create a new directory for your organization's JSPs and templates.

   For example,

   ```
   mkdir sesta
   ```

3. Copy the JSPs and templates that you wish to modify into the new directory location, maintaining the same directory structure.

   For example, if your new Desktop type will modify `/etc/opt/SUNWps/desktop/default/JSPProvider/content.jsp`, copy this file to `/etc/opt/SUNWps/desktop/sesta/JSPProvider/content.jsp`, and customize the file for the new Desktop type in that location.

4. Customize the JSPs templates in the `sesta` directory as required.

5. Change the dynamic Desktop Type attribute in the Identity Server software administration console to use the newly created directory.

   See "Changing the Desktop Type" on page 86 for more information.

Using the Desktop Tag Library

Desktops based on JSPs enable a customization process without the necessity of changing the provider Java classes. The implementation of the JSP-based Desktop uses a tag library which the Portal Server software supplies. Not all Desktop channels need to be JSP-based. There are also channels using HTML-based templates.

A tag library is exposed through Tag Library Descriptors (TLD) files, so tags are in their appropriate functional area. See Appendix D, "JavaServer Pages Tag Library Reference" for more information.

What Is the Sample Portal?

Conceptually, the Desktop is split into the following three well-defined components (see Figure 1-2 also):

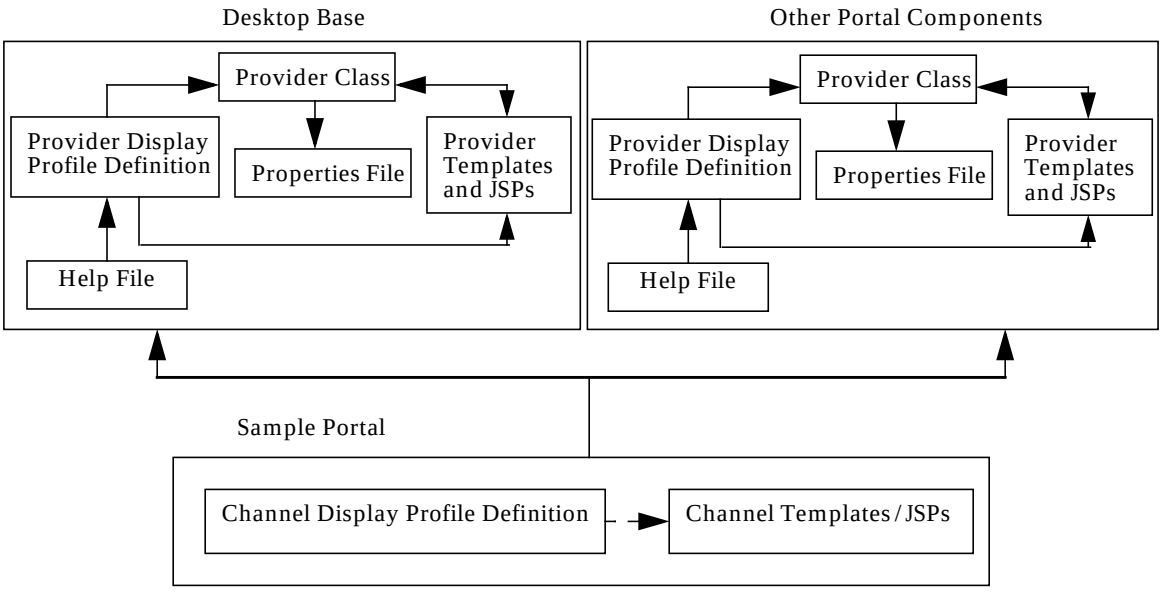
Base Desktop This includes Provider Java classes (based on the Provider API), provider display profile definitions and resource bundles (also referred to as properties files), display profile definitions of channels referenced by base Desktop XML or base Desktop JSPs and templates, default templates and JSPs (installed in `/etc/opt/SUNWps/desktop/default` directory), and help files.

Sample portal This includes organization level display profile definitions (such as themes and channel display profile definitions), templates and JSPs for the example Desktops in the sample portal (installed in `/etc/opt/SUNWps/desktop/sampleportal` directory), and user definition, Desktop display profile definitions, and templates and JSPs for the Authless user.

Other components This includes the Provider Java classes for their component specific providers, provider display profile definitions for their component specific providers and provider resource bundles (properties files), display profile definition for channels referenced by component display profile XML or component templates and JSPs, default templates and JSPs, and help files.

The sample portal has a dependency on the base Desktop and other components and cannot be installed if the base Desktop and other components are not installed. The base Desktop and other components are installed as part of the Portal Server software.

Figure 1-2 Desktop Components



When you install the Portal Server software, you can choose to install the sample portal. The sample portal includes five example Desktops that show the possibilities of the Portal Server software. In this way you can quickly get a feel for the kinds of containers that are possible to design.

Table 1-1 shows the five Desktops, which make up the sample portal. This two column table lists the containers (Desktops) in the first column and a brief description in the second column.

Table 1-1 Sun Java System Portal Server Sample Desktop Containers

Container (Desktop) Type	Description
JSPTabContainer	Generates a JSP-based tab Desktop.
JSPTableContainer	Generates a JSP-based table Desktop.
TemplateTableContainer	Generates a template-based table Desktop.
TemplateTabContainer	Generates a template-based tab Desktop.
FrameTabContainer	Generates a frame-based Desktop.

See Chapter 3, “Understanding the Sample Portal” for a complete description of the sample Desktop containers and how they function.

The sample portal can also serve as a place to start when building your own site's portal. You can customize the containers and use the building-block providers, such as XMLProvider and JSPProvider, to add customized content. The sample portal also includes content providers, such as BookmarkProvider, that cannot be extended but that can be used to provide content.

If the existing building-block and content providers do not meet your needs, you can either extend an existing building-block provider (content providers are not public and hence not extendible), or develop custom building-block providers. If either of these methods do not suit your needs, you can develop a custom provider.

NOTE The Portal Server software distinguishes between building-block providers, which you can extend using the Portal Server software APIs, and content providers, which you cannot extend. See the *Portal Server Developer's Guide* for more information on extending the providers.

Creating a New Desktop

Creating a new Desktop involves the following:

1. Developing the Container
2. Editing the Display Profile
3. Developing and Deploying JSP or Template Files
4. Loading the Display Profile at the Appropriate LDAP Nodes
5. (Optional) Creating a Resource Bundle
6. Accessing the Desktop, including Debugging the Desktop.

The following sections describe each of these steps in detail.

Developing the Container

You can develop a container by:

1. Defining a `<Container>` element in the display profile that references an existing `<Provider>` element.

2. Defining a <Provider> element in the display profile that references an existing provider class. You also must do Step 1 on page 37.
3. Defining a container provider class that extends an existing container provider such as JSPTableContainerProvider. You also must do Step 2.
4. Defining a container provider class from scratch that extends ContainerProviderAdapter. You also must do Step 2. If you create a container by extending the Provider class, then it also needs to implement the ContainerProvider interface.

NOTE You cannot create a container provider by just extending the Provider class. By definition, a container must implement the ContainerProvider interface. ContainerProviderAdapter does this.

If you write a new class file, it must reside in the /etc/opt/SUNWps/desktop/classes directory. You can change this location by editing the /etc/opt/SUNWps/desktop/desktopconfig.properties file.

NOTE You can also use the Identity Server software administration console to manipulate containers.

Editing the Display Profile

You need to edit the display profile XML and modify the following tag:

```
<Provider name="provider" class="provider class">
```

You also need to modify the following tag, which references the provider in the previous sentence:

```
<Container name="container" provider="provider">
```

For JSP files, the <Properties> tag for the provider contains the following property tag, which references the JSP Content page:

```
<String name="contentPage" value="value">
```

The <Properties> tag for the channel can have values that override the properties set in the <Provider> tag. Thus, if desired, you could set the JSP contentPage value here. You do not reference template-based providers, or other providers you might develop, in this way.

The <Available> and <Selected> tags are required for all containers in the display profile.

The JSP-based tab, table, and frametab containers have additional properties requirements. See Chapter 2, “Display Profile Properties” for more information.

NOTE There is a distinction between a provider element in the display profile and the Java class for the provider.

Provider element:

```
<Provider name="JSPTableContainer"
class=com.sun.portal.providers.containers.jsp.table.JSPTa
bleContainerProvider>
```

Java class:

```
com.sun.portal.providers.containers.jsp.table.JSPTableCon
tainerProvider
```

Developing and Deploying JSP or Template Files

You can modify existing JSP files (for example, tabs.jsp) or template files, or develop with your own. If you install the sample portal, the JSP and template files are located in the /etc/opt/SUNWps/desktop/sampleportal directory, in different subdirectories for each container.

See Appendix A, “Desktop Template Files” and Appendix C, “JavaServer Pages Reference” for more information on the sample JSP and template files.

For the JSPs, you can find compilation and runtime errors in the desktop debug log at /var/opt/SUNWam/debug/desktop.debug. Also, all JSPProvider based Desktop channels have a property called showExceptions. By default, this property is set to false; setting it to true causes the JSP exception to show up as the content of the channel.

When you create a container, you need to create a new subdirectory for your newly created container in the /etc/opt/SUNWps/desktop/*desktopType* directory. That is, the newly created container should be placed based on what the *desktopType* is. If sampleportal is installed, then the *desktopType* is, by default, sampleportal; so, create a new directory for the container under sampleportal so that any JSP and template

that is being added can adopt the same look and feel that as defined in the sampleportal. If sampleportal is not installed, and if you have set up a custom *desktopType*, for example, `foo`, then the new container directory must be created directory under `foo`.

Either copy the modified JSP or template files here, or place your newly created files here. If you use a sample container without changing any content or file names, you do not need to create a new subdirectory nor copy any files there. (In the example that follows, a new subdirectory is needed, because a new container is created.)

For example, let's say you create a new container called newSingleContainer whose display profile definition is the following:

```
<Container name="newSingleContainer" provider="JSPSingleContainer">
    <Properties>
        <String name="helpURL" value="desktop/newSingle.html"/>
        <String name="title" value="A new single container"/>
        <String name="contentPage" value="newsinglecontent.jsp"/>
        <Boolean name="isEditable" value="true"/>
        <String name="editType" value="edit_subset"/>
    </Properties>
    <Available/>
    <Selected/>
    <Channels/>
</Container>
```

Because the file specified for the `contentPage` property is different from the `contentPage` value for the provider definition, you need to create a new directory under the `/etc/opt/SUNWps/desktop/default` directory called `newSingleContainer`. You then only need to copy the `newsinglecontent.jsp` file to this new directory. The system is able to locate all other JSPs referenced by the JSPSingleContainer provider.

Loading the Display Profile at the Appropriate LDAP Nodes

Load the display profile at the appropriate LDAP node(s) by using the `dpadmin` command. You can also use the Edit Display Profile XML text box in the Identity Server software administration console (as long as you are not using Netscape 4.7x) or the Upload link.

See the *Portal Server Administration Guide* for more information on the `dpadmin` command, and the Edit Display Profile XML text box and Upload link.

(Optional) Creating a Resource Bundle

If you created a new provider, you may need to create a resource bundle file with the same name as the provider.

Accessing the Desktop

Access the Desktop in one of the following ways.

➤ **To Access the Desktop**

1. Use a specific container or channel reference by using the provider argument to the Desktop login URL:

   `http://hostname:port/portal/dt?provider=`*`providername`*

   where *providername* is one of the providers listed in Table 1-1 on page 36.

   For example, to access the JSP-based tab Desktop, in a browser, type:

   `http://hostname:port/portal/dt?provider=JSPTabContainer`

2. If no channel is referenced, the Desktop looks in the session for the last channel or container that was displayed. (This is stored in the session.)

3. If no channel is stored in the session, the Desktop looks in a Desktop service attribute for the top-level container to display (Default Channel Name attribute). This happens after an initial login.

Once this top-level container is determined, that container draws the containers or channels that it references (through the Selected list), until all of its leaves have been reached.

Debugging the Desktop

Use the following to help debug the Desktop environment:

The Desktop debug file is located at:

`/var/opt/SUNWam/debug/desktop.debug`

If you get an error message page on the Desktop, you can view the source to look at the stack trace. (For example, in Netscape Navigator, select Page Source from the View menu.)

Creating and Extending a Provider Class

A provider is responsible for delivering the content for a channel as well as defining the various channel properties such as title and description. A provider can optionally implement edit functionality.

You create a provider by creating a new provider class. The provider class is a Java class that implements all the methods defined in the Provider interface. The Portal Server software supplies basic implementation classes that implement the provider interface, ProviderAdapter. The ProfileProviderAdapter extends ProviderAdapter and adds some convenient methods that are indirectly defined in the ProviderContext class. Most of the time, you will extend ProfileProviderAdapter for your provider class. All building-block and content providers extend the ProfileProviderAdapter, either directly or indirectly.

See the *Portal Server Developer's Guide* for details about creating and extending a provider class.

Chapter 2

Display Profile Properties

This chapter describes the display profile global properties, and provider and channel specific properties. It contains the following sections:

- Overview of Display Profile Properties
- Display Profile Global Properties
- Display Profile Container Provider Properties
- Display Profile Leaf Provider Properties

Overview of Display Profile Properties

Display profile properties control all aspects of a channel, including:

- Content (available and selected channels)
- Position in the Desktop
- Controls

Display profile properties specify the per-channel configuration in the portal Desktop. Such properties define the visual representation of a channel in so much as the visual representation of the channel is affected by a display profile property.

The sample portal makes use of the following display profile definitions in the *portal-server-install-root*`/SUNWps/samples/desktop` directory:

`dp-org.xml`	Contains the display profile definitions for channels and containers.
`dp-providers.xml`	Contains the display profile definitions for providers.

`dp-anon.xml`	Contains the display profile definitions for channels and containers for the authlessanonymous and anonymous users in the default organization.

Display Profile Properties

The display profile properties are contained in a properties “bag.” A bag is simply a grouping mechanism for display profile entities such as channels, providers, and properties. The property itself does not have a properties bag associated with it.

You associate properties with the following display profile objects:

- `<Properties>` definition
- `<Provider>` definition
- `<Channel>` definition
- `<Container>` definition

There are four basic categories of properties; they are:

Global Global properties are accessible to all channels. You set global properties, which are shared by all channels, in the `<Properties> </Properties>` definition. Themes are an example of a global property. You define the theme data globally to share it among all channels. See “Display Profile Global Properties” on page 46 for more information.

NOTE Do not use global properties as defaults for all channels. Instead, use the `<Provider>` definition, as it sets the property interface used by the provider object that will use the `<Provider>` definition.

Provider Provider properties serve two purposes:

- They define a property template or schema, defining the properties that will be used by all channels based on the provider.
- The specific values in the provider serve as default values for channels.

If the property is not defined in channels based on this provider, the default value is used. If the default value is overridden by setting the value within the channel definition, then that value is used. By customizing a provider’s property values, you can customize all channels that the provider generates.

Channel Channel properties are available to the channel in which that properties are associated with. By customizing an individual channel's properties, you customize that particular channel.

NOTE Properties set in the `<Provider>` definitions are defaults for channels based on that provider. Properties set in `<Channel>` definitions override the defaults in the provider definition to customize the channel. For example, URLScaperProvider defines a `url` property. A default does not make sense here, thus a channel would naturally override this value.

Container Containers are simply channels that generate the majority of their content by executing other channels (or containers). Many of the properties defined for containers pertain to how to gather and arrange content from other channels. For example, properties set in the `<Container>` definition can describe how to display the contained channels in the container, including: the layout of the container (thin-wide, wide-thin, or thin-wide-thin), a list of the contained channels, the position of the channel (the row and column number), and the window state of the contained channels (maximized, minimized, or detached).

Lower priority display profile documents can overwrite properties of higher priority display profile documents using merge locking. That is, the lock stops the merge on a particular property or value. See the *Portal Server Administration Guide* for a complete discussion of the semantics of the display profile merging.

Display Profile Property Types

Table 2-1 on page 45 lists the property types for provider definitions. These can be used with leaf and container providers. This three column table lists the property types in the first column, a brief description in the second column, and an example in the third.

Table 2-1 Display Profile General Property Types

Property Type	Definition	Example
Boolean	An atomic object representing a Boolean value.	`<Boolean name="removable" value="true"/>`

Table 2-1 Display Profile General Property Types *(Continued)*

Property Type	Definition	Example

Collection	An object representing either a list or hash table. A collection is a type of property, or named bag, in which to put other properties.	`<Collection name="channelsRow">`
		`<String name="MailCheck" value="4"/>`
		`<String name="App" value="5"/>`
		`</Collection>`

ConditionalProperty	Defines the filtering criteria. The most common conditions are `locale` and `clientType`, but the API is generic in that it allows you to define and base properties on any sort of condition. `condition` and `value` are required attributes.
	In the administration console, the conditional properties are displayed as `condition-value` and can be edited like collections. The conditional properties can be nested and can be added to a channel or inside another conditional property. Use the Add Property page to add a new conditional property.
		`<ConditionalProperties condition="locale">`
		`<String name="en_US" value="English (United States)"/>`
		`</ConditionalProperties>`

Integer	An atomic object representing an integer value.	`<Integer name="numberOfHeadlines" value="7"/>`
Reference	An object representing a pointer to a channel definition (that is, to a channel name in a container's selected and available channel lists.) Reference is an unnamed string useful for design tools to be able to distinguish such things from strings.	`<Reference value="UserInfo"/>`
String	An atomic object representing a string value.	`<String name="title" value="Table Container Channel 1"/>`

Display Profile Global Properties

There are no global properties defined in the base Desktop. Global properties are added via the sample portal installation. So, if you did not install the sample portal, by default, you do not have any global properties defined in the base Desktop.

Use global properties to assign properties that apply to all channels. For example, Code Example 2-1 shows (a snippet of) the global properties defined in the dp-org.xml display profile file that is part of the sample portal. You assign the global properties inside the <Properties> </Properties> definition by using tags such as <Collection> </Collection>, <String> </String>, <Integer> </Integer>, and so on.

Code Example 2-1 Global Properties Sample in the Display Profile dp-org.xml File

```
<DisplayProfile version="1.0" priority="10">
    <Properties>

        <Collection name="GlobalThemes" propagate="false">
            <Collection name="SunTheme">
                ...
            </Collection>
        </Collection>
        <Collection name="UserTheme">
            ...
        </Collection>
        <String name="docroot" value=”/docs/”/>
        <ConditionalProperties condition="locale">
            <String name="en_US" value="English (United States)"/>
        </ConditionalProperties>
        <String name="helpURL" value="en/desktop/usedesk.htm" advanced="true"/>
        <Collection name="userDefinedChannels" propagate="false"/>
    </Properties>
```

The following is a list of all the global attributes available with the default installation of the sample portal. This two column table lists the attributes in the first (left) column and a brief description in the second (right) column.

GlobalThemes	Defines the global themes for the Desktop. Themes are mainly focused on channel decoration like background color, channel border color, border width, and font face. Custom themes give the end user the ability to change the look and feel of the Desktop beyond the preset themes. Global themes can be added in the display profile and changed by users in their Desktops. See “Customizing the Global Themes” on page 259 for information on adding global themes.

UserTheme	Defines the theme that shows up in the user's Desktop. The value must be one of the collection values defined in GlobalThemes. In dp-org.xml file, the value can be either theme1 or theme2. When users customize their Desktops, the value UserTheme will change.
locales	Used by UserInfoProvider to form the Language pull-down list.
docroot	Specifies the online help doc root relative to the installed portal/static location. See the Javadocs for more information on the getHelp() method in the ProviderContext API.
helpURL	Specifies a default help file that is used by all containers. If helpURL is specified in the container provider definition, then that one is used.
userDefinedChannels	Specifies the page to allow users to Create New Channel.

Display Profile Container Provider Properties

This section contains information on the display profile definitions and the properties of the building-block and internally used container providers that ship with Sun Java System Portal Server software.

Container providers enable you to aggregate channels inside the Desktop. The container building-block providers are building blocks in a sense since you can also customize them or use them differently by changing the container properties. They include:

- JSPTableContainerProvider - JSPTableContainerProvider is an extension of JSPProvider. This JSP table provider displays the content channels in a table.
- JSPTabContainerProvider - JSPTabContainerProvider is an extension of JSPProvider. This tab container provider displays a channel that is made up of a number of tabs with titles on them. By default, the JSPTabContainerProvider uses JSPTableContainer to lay out content for each tab. However, it can use JSPTableContainer, JSPSingleContainer, or JSPTabContainer to layout content for each tab.
- JSPSingleContainerProvider - JSPSingleContainerProvider is an extension of the JSP container provider. The single container provider displays one channel in it.

- TemplateTableContainerProvider - TemplateTableContainerProvider is the template-based table container. This provider displays the content channels in a table.
- TemplateTabContainerProvider - TemplateTabContainerProvider is the template-based tab container. TemplateTabContainerProvider contains support for a number of tabs in it.

See the Javadocs for more information on these containers.

Available and Selected List

All containers must define a list of available and selected channels. The presence of these is what mainly distinguishes a container from a channel.

Conceptually, the available list defines the set of channels that can be displayed in the container. The selected list defines those that are actually displayed in the container.

To take a specific example, consider the table container. Table containers use the available channel list to store channels that the user may add to their Desktop. The selected list is used to store the set of channels that are visible in their portal page. Typically, the selected channels are a subset of the available channels.

NOTE Containers are not required to make use of the available and selected channel lists in the display profile. A container may manage its contained channels in other implementation dependent ways. However, it is recommended that containers use the display profile available and selected channel lists in order to standardize how they are administrated.

The following is a list of the required container properties. This two column table lists the properties in the first (left) column and a brief description in the second (right) column.

`Available`	Defines a list of all available channels for this container. The `<Available>` and `</Available>` tags define the list, and the `<Reference value=>` tag defines the list items. For example:
	`<Available>`
	`<Reference value="UserInfo"/>`
	`<Reference value="MailCheck"/>`
	`<Reference value="App"/>`
	`<Reference value="Bookmark"/>`
	`</Available>`

`Selected`	Defines a list of selected channels for this container. Only selected channels are displayed on the Desktop. The `<Selected>` and `</Selected>` tags define the list, and the `<Reference value=>` tag defines the list items. For example:
	`<Selected>`
	`<Reference value="UserInfo"/>`
	`<Reference value="MailCheck"/>`
	`<Reference value="App"/>`
	`<Reference value="Bookmark"/>`
	`</Selected>`

Common Properties for Table Container

The following are the common properties for table containers. This two column table lists the property tags in the first column and a brief description in the second column.

The `<Collection name> </Collection>` tags define a list to contain these properties, which are set with the `<String>` tag.

Table 2-2 Table Container Properties

Property Tag	Description
parentTabContainer	Contained table containers have the parentTabContainer property whose value is the name of the tab container in which the contained table container is contained. If the contained table container has to be used in some other tab container, change this property value to the respective tab container name.
refreshParentContainerOnly	
layout	Defines the width of the table columns. Layout one (1) refers to thin-thick, layout two (2) refers to thick-thin, and layout three (3) refers to thin-thick-thin.
thin_popup_height	Defines the window height in pixels for the thin channel in the detached window.
thin_popup_width	Defines the window width in pixels for the thin channel in the detached window.
thick_popup_height	Defines the window height in pixels for the thick channel in the detached window.
thick_popup_width	Defines the window width in pixels for the thick channel in the detached window.
fullwidth_popup_height	Defines the window height in pixels for the full_top or full_bottom channel in the detached window.
fullwidth_popup_width	Defines the window width in pixels for the full_top or full_bottom channel in the detached window.
defaultChannelIsMinimizable	Defines the isMinimizable default value for the channels in this container. If you define a default value, then you do not have to define isMinimizable for all the leaf channels in the container. You can change the value for a leaf channel in the container if needed.
defaultchannelsIsMaximizable	Defines the isMaximizable default value for the channels in this container. If you define a default value, then you do not have to define isMaximizable for all the leaf channels in the container. You can change the value for a leaf channel in the container if needed.
defaultChannelIsMinimized	Defines the isMinimized default value for the channels in this container. If you define a default value, then you do not have to define isMinimized for all the leaf channels in the container. You can change the value for a leaf channel in the container if needed.
defaultChannelIsDetached	Defines the isDetached default value for the channels in this container. If you define a default value, then you do not have to define isDetached for all the leaf channels in the container. You can change the value for a leaf channel in the container if needed.

Table 2-2 Table Container Properties *(Continued)*

Property Tag	Description
defaultChannelIsDetachable	Defines the isDetachable default value for the channels in this container. If you define a default value, then you do not have to define isDetachable for all the leaf channels in the container. You can change the value for a leaf channel in the container if needed.
defaultChannelIsRemovable	Defines the isRemovable default value for the channels in this container. If you define a default value, then you do not have to define isRemoveable for all the leaf channels in the container. You can change the value for a leaf channel in the container if needed.
defaultChannelHasFrame	Defines the hasFrame default value for the channels in this container. If you define a default value, then you do not have to define hasFrame for all the leaf channels in the container. You can change the value for a leaf channel in the container if needed.
defaultChannelIsMovable	Defines the isMovable default value for the channels in this container. If you define a default value, then you do not have to define isMovable for all the leaf channels in the container. You can change the value for a leaf channel in the container if needed.
defaultChannelColumn	Defines the column number default value for the channels in this container. If you define a default value, then you do not have to define the column number for all the leaf channels in the container. You can change the value for a leaf channel in the container if needed.
defaultChannelRow	Defines the row number default value for the channels in this container. If you define a default value, then you do not have to define row number for all the leaf channels in the container. You can change the value for a leaf channel in the container if needed.
channelsIsMinimized	Defines a collection property to contain the isMinimized value for channels in this container.
channelsIsDetached	Defines a collection property to contain the isDetached value for channels in this container.
channelsHasFrame	Defines a collection property to contain the hasFrame value for channels in this container.
channelsIsMinimizable	Defines a collection property to contain the isMinimizable value for channels in this container.
channelsIsMaximizable	Defines a collection property to contain the isMaximizable value for channels in this container.
channelsRow	Defines a collection property to contain the row number value for channels in this container.
channelsColumn	Defines a collection property to contain the column number value for channels in this container.
channelsIsMovable	Defines a collection property to contain the isMovable value for channels in this container.

Table 2-2 Table Container Properties *(Continued)*

Property Tag	Description
channelsIsDetachable	Defines a collection property to contain the isDetachable value for channels in this container.
channelsIsRemovable	Defines a collection property to contain the isRemovable value for channels in this container.
borderlessChannels	Defines the collection property to contain the channel name and Boolean value pair for specifying borderless channels in this container. A value of true means the channel does not have border.
defaultBorderlessChannel	Defines the default value for the borderless channels in this container. If you define a default value, then you do not have to define borderlessChannels for all leaf channels in the container. You can change the value for a leaf channel in the container if needed.

Common Properties for Tab Container

The following is a list of the properties common to all TabContainerProviders. This two column table lists the properties in the first (left) column and a brief description in the second (right) column.

startTab	Tab that is displayed when the user logs in.
makeTabChannel	Container channel name to be used when the user creates a new tab.
makeTabProvider	Container provider to be used as a base provider when the user creates a new tab.
maxTabs	The maximum number of tabs that can be selected on the user's Desktop.
channelNumber	Used in the naming of newly created tabs by user.
contentChannel	The content channel to be used as the Content page for a user created tab.
TabProperties	The collection property TabProperties creates the new tab. There needs to be a one-to-one mapping between the contents of the TabProperties collection and the available or selected tabs. That is, for every tab specified in the available or selected list, a new collection needs to be defined inside TabProperties collection.

Other Container Properties

The following is a list of properties that are common to all container providers. This two column table lists the property in the first column and a brief description in the second column.

`presetThemeChannel`	Defines the preset theme channel for the container. The JSP™ defined in the channel displays the Theme->Preset Themes page.
`customThemeChannel`	Defines the custom theme channel for the container. The JSP defined in the channel displays the Theme-> Custom Theme page.
`editContainerName`	Defines the edit container channel for this container. When a leaf channel defined in this container is of the type `edit_subset`, then the edit container channel is used to display a frame for the Edit page for the leaf channel.

Display Profile Leaf Provider Properties

Providers have required properties, as well as general properties. This section describes each provider-specific property and the general properties of the providers.

By editing the provider properties in the display profile XML files, you can create customized display profile provider definitions. Any time you modify a display profile document, use the `dpadmin` command (or the Sun Java System Identity Server software administration console) to store it in LDAP. For information about using the `dpadmin` command, see the *Portal Server Administration Guide*.

Display Profile Properties of Building-Block Providers

This section describes the display profile definitions and the properties of the leaf building-block providers.

Leaf building-block providers generate their own content. They include:

- JSPProvider

- URLScraperProvider
- XMLProvider

JSPProvider

JSPProvider uses JavaServer Pages™ (JSP™). JSPProvider obtains content from one or more JSP files. A JSP file can be a static document (HTML only) or a standard JSP file with HTML and Java code. A JSP can include other JSP files. However, only the topmost JSP can be configured through the display profile. The topmost JSP files are defined through the `contentPage`, `editPage`, and `processPage` properties. See the *Portal Server Developer's Guide* for more information on how JSPProvider uses these JSPs.

If you need to make other customizations, you do so in the JSP files themselves. The following is a list of the properties specific to JSPProvider. This two column table lists the properties in the first (left) column and a brief description in the second (right) column.

`String name="contentPage"`	Specifies the JSP that is used to generate the channel content (by using the `getContent` method).
`String name="editPage"`	Specifies the JSP that is used to generate the Edit page content (by using the `getEdit` method).
`String name="processPage"`	Specifies the JSP that is used to process the results of an Edit page (by using the `processEdit` method).
`Boolean name="showExceptions"`	If true, makes JSPProvider show exceptions generated while processing the JSP as the channel output for the `getContent` and `getEdit` methods. This can be useful for developing and troubleshooting your portal.

URLScraperProvider

URLScraperProvider takes a URL, opens a connection to the URL, and reads the contents into a buffer. The contents are then sent to the Desktop servlet, which displays it. URLScraperProvider uses the Rewriter to construct the URL information and the content received contains the presentation markup (if applicable). See the *Portal Server Administration Guide* for more information on the Rewriter.

The following is a list of the properties specific to URLScraperProvider. This two column table lists the properties in the first (left) column and a brief description in the second (right) column.

`String name="url"`	Specifies the URL to be scraped. The default value is `/desktop/ipinfo.html`.
`String name="urlScraperRulesetID"`	Specifies the ID of the ruleset to be used by the Rewriter for rewriting content.
`Boolean name="cookiesToForwardAll"`	Specifies whether to forward cookies.
`String name="inputEncoding"`	Specifies the input encoding to be used by URLScraperProvider to encode the scraped content.
`Collection name="cookiesToForwardList"`	Specifies the list of cookies to be forwarded by URLScraperProvider if `cookiesToForwardAll` is set to `false`.
`Integer name="timeout"`	Specifies the timeout for which the provider should wait to fetch content before displaying the timed out message.

The `isEditable` property for URLScraperProvider cannot be turned on (set to `true`) as this channel is, by default, not editable. There are no `getEdit()` and `processEdit()` methods defined for this provider. If you want edit functionality for URLScraperProvider, define another provider that extends URLScraperProvider. In so doing, you would need to implement the `getEdit()` and `processEdit()` methods, and also define the `editType` property. See the *Portal Server Developer's Guide* for more information on extending the URLScraperProvider.

XMLProvider

XMLProvider transforms an XML document into HTML using an XSLT (XML Style Sheet Language) file. You must create the appropriate XSLT file to match the XML document type. XMLProvider is an extension of URLScraperProvider. This provider uses the JAXP 1.1 JAR files that come with Sun Java System Web Server software.

NOTE This guide does not discuss XML and XSL technologies. See `http://www.w3.org/TR/xslt` for more information.

The following is a list of the properties specific to XMLProvider. This two column table lists the properties in the first (left) column and a brief description in the second (right) column.

`String name="url"`	Specifies the URL that XMLProvider is to transform.
`String name="xslFileName"`	Specifies the path to the local file to be used as the XSL style sheet. The provider code tries to pick up the XSL file either from the XML channel directory (that is, `/etc/opt/SUNWps/desktop/default/SampleXML`), or if not specified here, from the XML provider directory (`/etc/opt/SUNWips/desktop/default/XMLProvider/xml`).
`String name="urlScraperRulesetID"`	Specifies the ID of the ruleset to be used by the Rewriter for rewriting content.
`Boolean name="cookiesToForwardAll"`	Specifies whether to forward cookies.
`String name="inputEncoding"`	Specifies the input encoding to be used by XMLProvider to encode the scraped content.
`Collection name="cookiesToForwardList"`	Specifies the list of cookies to be forwarded by URLScraperProvider if `cookiesToForwardAll` is set to `false`.
`Integer name="timeout"`	Specifies the timeout for which the provider should wait to fetch content before displaying the timed out message.

Display Profile Properties of Service Providers

Service providers are providers who provide a service, such as search service. The Sun Java System Portal Server software includes the following service providers:

- SearchProvider
- DiscussionProvider
- SubscriptionsProvider

SearchProvider

SearchProvider supplies the search function using the Sun Java System Portal Server software Search Engine. SearchProvider is a JSP-based provider. The resultant channel has three interfaces:

Basic search Enables users to search within the default document database or discussion database. Document and category matches are then displayed.

Advanced search Enables users to search for documents based on author, title, URL within the default document database or discussion database, discussion, and / or comment. Users can also search on the last-modified date of a document. Advanced search is a more complex user interface, and supports customization. See Chapter 14, “Customizing the Service Providers” for more information.

Browse Enables users to browse the category tree and search within categories.

Search results are displayed based on the `categorySearch` and `viewHits` properties. The following is a list of the properties specific to SearchProvider. This two column table lists the properties in the first (left) column and a brief description in the second (right) column.

`String name="contentPage"`	Specifies the JSP that is used to generate the channel content (by using the `getContent()` method).
`String name="editPage"`	Specifies the JSP that is used to generate the Edit page content (by using the `getEdit()` method).
`String name="processPage"`	Specifies the JSP that is used to process the results of an Edit page (by using the `processEdit()` method).
`Boolean name="showExceptions"`	If true, makes SearchProvider show exceptions generated while processing the JSP as the channel output for the `getContent()` and `getEdit()` methods. This can be useful for developing and troubleshooting your portal, and for debugging the Search provider.
`String name="searchServer"`	Specifies the Search server’s URL.
`Integer name="viewHits"`	Specifies the number of hits that should be displayed per page. The maximum desirable number is 25. (The Edit page specifies to choose a number between 1 and 100.) This property is user editable. Edit page displays allowable values as 5, 8, 10, 16.

`Boolean name="basicSearchDefault"`	If true, specifies that the default search mode should be basic. (Users can set this to advanced if desired.)
`String name="defaultMode"`	Specifies the default search mode. Allowable values are basic, advanced, or browse.
`Boolean name="categorySearch"`	Specifies the category search to be displayed by default. If set to false, category matches are not displayed.

DiscussionProvider

The DiscussionProvider is JSPProvider based and uses the Desktop themes. It retrieves data from the back end Search service using search taglibs and API. The discussions and comments are stored as separate Resource Descriptors (RDs) in the discussion database. Discussion RDs require special schema. See `schema.rdm` file in the `/var/opt/SUNWps/https-`*psserver*`/portal/config/` directory.

Discussions are stored in the discussion database specified in the `dbname` property in the display profile. Search server host (`searchServer` property) and database name (`dbname` property) are advanced properties that can be configured in the display profile.

The following is a list of properties specific to DiscussionProvider. This two column table lists the properties in the first (left) column and a brief description in the second (right) column.

`searchServer`	Specifies the path to the search server. By default, the value is `portal/search`.
`dbname`	Specifies the discussion database where discussions are stored. Any valid database can be specified.
`viewHits`	Specifies the number of discussions to display on the main discussion page (full view).
`defaultDiscussionDisplay`	Determines how the comment subtree is displayed. It can be set to flat or threaded to allow the comment subtree to be displayed as flat or threaded.

`defaultFilter`	Specifies the filter for searching and displaying discussions and this controls display of the subtree. It can be based on ratings such as irrelevant, routine, interesting, important, or must read. By default, its value is irrelevant; so all comments rated irrelevant and above are displayed. The Must read filter will highlight the highly rated comments.
`defaultExpansionThreshold`	It can be set to expand all or collapse all. By default, its value is set to collapse all. If set to expand all, it will expand all the filtered comments, show description, rating menu, and allow user to post reply via links.
`viewDiscussionWindow`	A user configurable property. If set to `true`, the discussion link gets displayed on an entire page; that is, JSPDynamicSingleContainer is invoked. If set to `false`, the discussion gets displayed within the channel within the tab.
`anonymousAuthor`	An anonymous user can submit comments. Default author value for an anonymous user is picked from this property. Default value is `anonymous`. For example, it can be set to `unknown author`.
`displaySearch`	Enable or disable Search in discussions.
`showDescription`	Specifies whether or not to show a description of the discussion.
`ratingText`	Specifies the type of rating that can be done on a discussion. By default, discussions can be rated as irrelevant, routine, interesting, important, or must read. This property is not used in this release.

SubscriptionsProvider

SubscriptionsProvider provides subscriptions service to users. The Subscriptions service enables users to create a set of profile of interest over a source of information. The source of information supported are categories, discussions, and searchable documents. The profile is updated with the latest information every time the user accesses the Subscriptions channel. The Subscriptions channel summarizes the number of hits (relevant information) that matches each profile entry the user defined for categorized document and/or discussions.

Display Profile Properties of Content Providers

This section provides definitions and examples for the following content providers that ship with the Portal Server software.

- AddressBookProvider
- AppProvider
- BookmarkProvider
- CalendarProvider
- IMProvider
- LoginProvider
- LotusNotesAddressBookProvider, LotusNotesCalendarProvider, and LotusNotesMailProvider
- MailCheckProvider
- MailProvider
- MSExchangeAddressBookProvider, MSExchangeCalendarProvider, and MSExchangeMailProvider
- NotesProvider
- SimpleWebServiceProvider and SimpleWebServiceConfigurableProvider
- UserInfoProvider

You cannot extend content providers as their APIs are not public.

AddressBookProvider

The address book provider works with the Sun Java System Messaging Server to provide simple personal address book functionality.

The following is a list of the properties specific to AddressBookProvider. This two column table lists the properties in the first (left) column and a brief description in the second (right) column.

`String name="sunPortalABSortBy"`	Specifies the value of the entries displayed to sort by.
`String name="sunPortalABSortOrder"`	Specifies the sort order of the entries displayed.

`String name = "ssoAdapter"`	Specifies the SSOAdapter configuration used by this provider / channels.
`Integer name="maxEntries"`	Specifies the limit for the number of address book entries to display.
`Integer name="numEntries"`	Specifies the number of entries to display.
`Boolean name="displayEntries"`	Specifies if the entries should be shown.
`Collection name="applicationHelperEdit"`	Specifies the mail application helpers that you can edit settings on.
`String name="applicationHelperURL"`	Specifies the default mail application helper.
`Collection name="ssoEditAttributes"`	Specifies the attributes that will appear on the 1st edit page for the provider. These are usually server settings and have nothing to do with display profile attributes.
`Collection name="dpEditAttributes"`	Specifies the attributes that will appear on the edit page for the application helper. These are usually display attributes and there can be multiple attributes based on number of clients, and so on.
`Collection name="sunPortalABSortBySelectOptions"`	Used to generate the drop down select boxes on the edit page. This specifies None and Full name.
`Collection name="sunPortalABSortOrderSelectOptions"`	Used to generate the drop down select boxes on the edit page. This specifies Ascending, Descending, and None.

AppProvider

AppProvider enables a user to add or remove applications from a list of applications.

The following is a list of the properties specific to AppProvider. This two column table lists the properties in the first (left) column and a brief description in the second (right) column.

`String name="windowPref"`	Specifies how to launch a link. The possible values are: • `all_new` (New window is opened for every link) • `one_new` (All links open on the same new window) • `same` (Desktop window)
`Collection name="targets"`	Specifies the list of application links in *name* | `URL` format, where *name* should match should match the entry in the `userApps` collection.
`Collection name="userApps"`	Specifies the list of applications that appear in the applications channel.

BookmarkProvider

BookmarkProvider enables a user to add or remove URLs from a list of bookmarks.

The following is a list of the properties specific to BookmarkProvider. This two column table lists the properties in the first (left) column and a brief description in the second (right) column.

`String name="windowPref"`	Specifies how to launch a link. The possible values are: • `all_new` (New window is opened for every link) • `one_new` (All links open on the same new window) • `same` (Desktop window)
`Collection name="targets"`	Specifies the list of bookmarks that is shown in the channel in the following format: *BookmarkName*|`URL`

CalendarProvider

The calendar provider works with the Sun Java System Calender Server so that you can view tasks and events and launch Calendar Express without having to sign in.

The following is a list of the properties specific to CalendarProvider. This two column table lists the properties in the first (left) column and a brief description in the second (right) column.

`String name="view"`	Specifies the view (day, week, or month) used.
`String name="calendar"`	Specifies the calendar to display.
`String name="ssoAdapter"`	Specifies the ssoAdapter configuration to use.
`Boolean name="loadSubscribedCalendars"`	If set to `true`, it will try to load all of the subscribed calendars and display them.
`Boolean name="disableTaskEventURLs"`	If set to `true`, it will not display links for tasks and events.
`Collection name="calendarSelectOptions"`	Specifies a list of all subscribed calendars.
`Collection name="applicationHelperEdit"`	Specifies the mail application helpers that you can edit settings on
`String name="applicationHelperURL"`	Specifies the default mail application helper
`Collection name="ssoEditAttributes"`	Specifies the attributes that will appear on the 1st edit page for the provider. These are usually server settings and have nothing to do with display attributes
`Collection name="dpEditAttributes"`	Specifies the attributes that will appear on the edit page for the application helper. These are usually display attributes and there can be multiple attributes based on number of clients, etc.
`Collection name="viewSelectOptions"`	Specifies the different Calendar views displayed in the Calendar edit page.

IMProvider

The IMProvider includes:

- Information needed to help the user decide whether to launch the IM client.
- The ability to launch the IM client using single-sign-on.

The information is gathered by accessing the Instant Messaging server through the use of the Instant Messaging APIs.

The following is a list of the properties specific to IMProvider. This two column table lists the properties in the first (left) column and a brief description in the second (right) column.

server	Specifies the name of the instant messaging server to use.
port	Specifies the port on which the instant messaging server listens.
mux	Specifies the name of instant messaging multiplexor to use (used by IM client.)
muxport	Specifies the port on which the instant messaging server listens.
codebase	Specifies where to find the instant messaging client.
netletRule	Specifies where to find the instant messaging client when using the netlet. By default, the value is IM.
clientRunMode	Specifies how the Instant Messaging server client must be run. The client can be run as either a plugin or jnlp. By default, the value is plugin.
authMethod	Specifies the authentication method. Clients can authenticate either via idsvr (for Identity Server) or ldap. By default, the value is idsvr.
authUsernameAttr	Specifies the LDAP attribute where instant messaging username is found. By default, the value is uid.
username	Specifies the username for LDAP authentication. This is not applicable if authMethod is set to idsvr.
password	Specifies the password for LDAP authentication. This is not applicable if authMethod is set to idsvr.
contactGroup	Specifies the contact group to display, or blank for all.

LoginProvider

LoginProvider enables the Login channel to show up in the anonymous user's Desktop. You can configure LoginProvider to enable users to log in and out using the Login channel. The system administrator can select one out of the three methods to enable users to log in: LDAP, Membership, or UNIX.

For the sample portal, if you type the following URL in a browser, you see the authlessanonymous user's Desktop, which has the login channel.

```
http://hostname:port/portal/dt
```

By default, LoginProvider uses Membership authentication. No additional setup is required to use this channel. From the authlessanonymous user page, valid users can use the login channel, and new users can register using the Sign me up link in the channel. You can change the authentication module for the login channel.

The following properties are specific to the LoginProvider. This two column table lists the property in the first (left) column and a brief description in the second (right) column.

`Boolean name="persistentCookie"`	Specifies if a persistent cookie is used to remember the user ID and password.
`Boolean name="federationEnabled"`	If set to true, the `libertyLogin.Template` is inserted.
`String name="preLoginURL"`	The value specified in the channel. This property is typically of the form: `http://www.siroe.con:80/amserver/preLogin?metaAlias=www.siroe.com&goto=http://www.siroe.com:80/portal/dt`

LotusNotesAddressBookProvider

The address book provider works with the Lotus Notes Server to provide simple personal address book functionality.

The following properties are specific to LotusNotesAddressBookProvider. This two column table lists the property in the first (left) column and a brief description in the second (right) column.

`String name="sunPortalABSortBy"`	Specifies the value of the entries displayed to sort by.
`String name="sunPortalABSortOrder"`	Specifies the sort order of the entries displayed.
`String name = "ssoAdapter"`	Specifies the SSOAdapter configuration used by this provider / channels.
`Integer name="maxEntries"`	Specifies the limit of address book entries to display.
`Integer name="numEntries"`	Specified the number of entries to display.
`Boolean name="displayEntries"`	Specifies if the entries should be shown.

`Collection name="applicationHelperEdit"`	Specifies the mail application helpers that you can edit settings on.
`String name="applicationHelperURL"`	Specifies the default mail application helper.
`Collection name="ssoEditAttributes"`	Specifies the attributes that will appear on the 1st edit page for the provider. These are usually server settings and have nothing to do with display attributes.
`Collection name="dpEditAttributes"`	Specifies the attributes that will appear on the edit page for the application helper. These are usually display attributes and there can be multiple attributes based on number of clients, etc.
`Collection name="sunPortalABSortBySelectOptions"`	Used to generate the drop down select boxes on the edit page. This specifies None and Full name.
`Collection name="sunPortalABSortOrderSelectOptions"`	Used to generate the drop down select boxes on the edit page. This specifies Ascending, Descending, and None.

LotusNotesCalendarProvider

The calendar provider works with the Lotus Notes Server so that you can view tasks and events and launch the web application without having to sign in.

The following properties are specific to LotusNotesCalendarProvider. This two column table lists the property in the first (left) column and a brief description in the second (right) column.

`String name="view"`	Specifies the view (day, week, or month) used.
`String name="calendar"`	Specifies the calendar to display.
`String name="ssoAdapter"`	Specifies the ssoAdapter configuration to use.
`Boolean name="loadSubscribedCalendars"`	If set to `true`, it will try to load all of the subscribed calendars and display them.
`Boolean name="disableTaskEventURLs"`	If set to `true`, it will not display links for tasks and events.
`Collection name="calendarSelectOptions"`	Specifies a list of all subscribed calendars.

`Collection name="applicationHelperEdit"`	Specifies the mail application helpers that you can edit settings on.
`String name="applicationHelperURL"`	Specifies the default mail application helper.
`Collection name="ssoEditAttributes"`	Specifies the attributes that will appear on the 1st edit page for the provider. These are usually server settings and have nothing to do with display attributes.
`Collection name="dpEditAttributes"`	Specifies the attributes that will appear on the edit page for the application helper. These are usually display attributes and there can be multiple attributes based on number of clients, etc.
`Collection name="viewSelectOptions"`	Specifies the different Calendar views displayed in the Calendar edit page.

LotusNotesMailProvider

The mail provider works with the Lotus Notes Server to provide simple mail functionality.

The following properties are specific to LotusNotesMailProvider. This two column table lists the property in the first (left) column and a brief description in the second (right) column.

`String name = "sortOrder"`	Specifies the sort order for the messages currently displayed.
`String name = "ssoAdapter"`	Specifies the SSOAdapter configuration used by this provider/channels.
`Integer name="numberHeaders"`	Specifies the limit of message headers to display. Hard limit is 30.
`Boolean name="displayHeaders"`	Specifies if the headers should be shown.
`Boolean name="sentFolderCopy"`	Specifies if sent messages should be copied to the Sent Folder (used by MA).
`Collection name="applicationHelperEdit"`	Specifies the mail application helpers that you can edit settings on.
`String name="applicationHelperURL"`	Specifies the default mail application helper.

`Collection name="ssoEditAttributes"`	Specifies the attributes that will appear on the 1st edit page for the provider. These are usually server settings and have nothing to do with display attributes.
`Collection name="dpEditAttributes"`	Specifies the attributes that will appear on the edit page for the application helper. These are usually display attributes and there can be multiple attributes based on number of clients, etc.
`Collection name="sortOrderSelectOptions"`	Specifies the different mail sort order options (recent at top or bottom).

MailCheckProvider

MailCheckProvider gives information about a user's mail status.

The following is a list of the properties specific to MailCheckProvider. This two column table lists the properties in the first (left) column and a brief description in the second (right) column.

`String name="IMAPServerName"`	Specifies the IMAP server name.
`String name="IMAPUserID"`	Specifies the IMAP user name.
`String name="IMAPPassword"`	Specifies the IMAP password.
`Collection name="targets"`	Specifies the list of application links in *name* `|` `URL` format, where *name* should match should match the entry in the `userApps` collection.
`Collection name="userApps"`	Specifies the list of mail applications that appear in the applications channel.

`Boolean name="defaultConfigParameters"`	MailCheckProvider uses the `defaultConfigParameters` property to read a user's mail settings. If set to `true`, MailCheckProvider reads the mail settings (server name, username, and password) from the NetMail service definition. If set to `false`, MailCheckProvider reads the mail settings from the properties defined in its display profile provider properties (`IMAPServerName`, `IMAPUserId`, and `IMAPPassword`). When `defaultConfigParameters` is set to `true`, the property `isEditable` is set to `false` implying that the Edit button is not available. If you change `defaultConfigParameters` to `false`, change `isEditable` to `true` for the Edit button to appear.

MailProvider

The mail provider works with the Sun Java System Messaging Server software to provide simple mail functionality and single sign-on services.

The following is a list of the properties specific to MailProvider. This two column table lists the properties in the first (left) column and a brief description in the second (right) column.

`String name = "sortOrder"`	Specifies the sort order for the messages currently displayed.
`String name = "ssoAdapter"`	Specifies the SSOAdapter configuration used by this provider / channels.
`Integer name="numberHeaders"`	Specifies the limit of message headers to display. Hard limit is 30.
`Boolean name="displayHeaders"`	Specifies if the headers should be shown.
`Boolean name="sentFolderCopy"`	Specifies if sent messages should be copied to the Sent Folder (used by MA).
`Collection name="applicationHelperEdit"`	Specifies the mail application helpers that you can edit settings on.
`String name="applicationHelperURL"`	Specifies the default mail application helper.

`Collection name="ssoEditAttributes"`	Specifies the attributes that will appear on the first edit page for the provider. These are usually server settings and have nothing to do with display attributes.
`Collection name="dpEditAttributes"`	Specifies the attributes that will appear on the edit page for the application helper. These are usually display attributes and there can be multiple attributes based on number of clients, etc.
`Collection name="sortOrderSelectOptions"`	Specifies the different mail sort order options (recent at top or bottom).

MSExchangeAddressBookProvider

The address book provider works with the Microsoft Exchange Server to provide simple personal address book functionality.

The following properties are specific to MSExchangeAddressBookProvider. This two column table lists the property in the first (left) column and a brief description in the second (right) column.

`String name="sunPortalABSortBy"`	Specifies the value of the entries displayed to sort by.
`String name="sunPortalABSortOrder"`	Specifies the sort order of the entries displayed.
`String name = "ssoAdapter"`	Specifies the SSOAdapter configuration used by this provider / channels.
`Integer name="maxEntries"`	Specifies the limit of address book entries to display.
`Integer name="numEntries"`	Specified the number of entries to display.
`Boolean name="displayEntries"`	Specifies if the entries should be shown.
`Collection name="applicationHelperEdit"`	Specifies the mail application helpers that you can edit settings on.
`String name="applicationHelperURL"`	Specifies the default mail application helper.
`Collection name="ssoEditAttributes"`	Specifies the attributes that will appear on the 1st edit page for the provider. These are usually server settings and have nothing to do with display attributes.

`Collection name="dpEditAttributes"`	Specifies the attributes that will appear on the edit page for the application helper. These are usually display attributes and there can be multiple attributes based on number of clients, etc.
`Collection name="sunPortalABSortBySelectOptions"`	Used to generate the drop down select boxes on the edit page. This specifies None and Full name.
`Collection name="sunPortalABSortOrderSelectOptions"`	Used to generate the drop down select boxes on the edit page. This specifies Ascending, Descending, and None.

MSExchangeCalendarProvider

The calendar provider works with the Microsoft Exchange Server so that you can view tasks and events and launch Exchanges web application.

The following properties are specific to MSExchangeCalendarProvider. This two column table lists the property in the first (left) column and a brief description in the second (right) column.

`String name="view"`	Specifies the view (day, week, or month) used.
`String name="calendar"`	Specifies the calendar to display.
`String name="ssoAdapter"`	Specifies the ssoAdapter configuration to use.
`Boolean name="loadSubscribedCalendars"`	If set to `true`, it will try to load all of the subscribed calendars and display them.
`Boolean name="disableTaskEventURLs"`	If set to `true`, it will not display links for tasks and events.
`Collection name="calendarSelectOptions"`	Specifies a list of all subscribed calendars.
`Collection name="applicationHelperEdit"`	Specifies the mail application helpers that you can edit settings on.
`String name="applicationHelperURL"`	Specifies the default mail application helper.
`Collection name="ssoEditAttributes"`	Specifies the attributes that will appear on the 1st edit page for the provider. These are usually server settings and have nothing to do with display attributes.

`Collection name="dpEditAttributes"`	Specifies the attributes that will appear on the edit page for the application helper. These are usually display attributes and there can be multiple attributes based on number of clients, etc.
`Collection name="viewSelectOptions"`	Specifies the different Calendar views displayed in the Calendar edit page.

MSExchangeMailProvider

The mail provider works with the Microsoft Exchange Server to provide simple mail functionality.

The following is a list of the properties specific to MSExchangeMailProvider. This two column table lists the properties in the first (left) column and a brief description in the second (right) column.

`String name="sortOrder"`	Specifies the sort order for the messages currently displayed.
`String name="ssoAdapter"`	Specifies the SSOAdapter configuration used by this provider / channels.
`Integer name="numberHeaders"`	Specifies the limit of message headers to display. Hard limit is 30.
`Boolean name="displayHeaders"`	Specifies if the headers should be shown.
`Boolean name="sentFolderCopy"`	Specifies if sent messages should be copied to the Sent Folder (used by MA).
`Collection name="applicationHelperEdit"`	Specifies the mail application helpers that you can edit settings on.
`String name="applicationHelperURL"`	Specifies the default mail application helper.
`Collection name="ssoEditAttributes"`	Specifies the attributes that will appear on the 1st edit page for the provider. These are usually server settings and have nothing to do with display attributes.
`Collection name="dpEditAttributes"`	Specifies the attributes that will appear on the edit page for the application helper. These are usually display attributes and there can be multiple attributes based on number of clients, etc.

`Collection name="sortOrderSelectOptions"`	Specifies the different mail sort order options (recent at top or bottom).

NotesProvider

NotesProvider enables the administrator or users the administrator has authorized to post a note to all users' Desktops in the Notes channel.

The following is a list of the properties specific to NotesProvider. This two column table lists the properties in the first (left) column and a brief description in the second (right) column.

`String name="location"`	Specifies the path to the text file, which contains the notes, in the file system.
`String name="lines"`	Specifies the number of lines of notes that is displayed in the channel.
`String name="maxLines"`	Specifies the maximum number of lines that can be displayed in the channel.
`Integer name="timeout"`	Specifies the time zone of the time stamp at which the notes were logged, either as an abbreviation such as `PST`, a full name such as `America/Los_Angeles`, or a custom ID such as `GMT-8:00`. Support of abbreviations is for JDK™ 1.1.x compatibility only and full names should be used.

Notes are stored and read in a text file in the following format:

`userid | date | message`

where | is the delimiter and date is the long value that denotes the time elapsed in milliseconds since January 1, 1970.

Following is a sample notes file.

```
User1|1007159465858|Message to Portal Desktop Team : Lets meet today at 2PM
User2|1007159465858|Information related to project is availble at home page
```

SimpleWebServiceProvider

SimpleWebServiceProvider, an extension of JSPProvider, makes simple web services available to an end user channel. SimpleWebServiceProvider dynamically constructs a user interface given a Web Services Description Language (WSDL) URL and a web service method name.

Using the URL, SimpleWebServiceProvider fetches the WSDL document, parses and validates it. Based on its content, SimpleWebServiceProvider generates input parameters to the method that return the information from the web service. The information is then displayed in the channel content window.

SimpleWebServiceProvider can generate channels that use the same web service, and the same method, so default parameter values can be stored using the Edit function.

SimpleWebServiceProvider supports basic data types such as String, int, and float as defined in the WSDL specification. It supports Complex Types if they are made up of only basic types (one level of nesting). There is no support for arrays.

SimpleWebServiceProvider can provide WSDL parsing for any other provider that needs it. SimpleWebServiceProvider is designed for stock quote or currency exchange rate content.

The following is a list of the properties specific to SimpleWebServiceProvider. This two column table lists the properties in the first (left) column and a brief description in the second (right) column.

`String name="wsdlURL"`	Specifies the URL to web service WSDL.
`String name="methodName"`	Specifies the web service method name that is going to be executed.
`Boolean name="isDefaultShowOutput"`	Specifies the default value. If `true`, the channel uses the default input value. If `false`, the channel uses the user input value.
`String name="contentPage"`	Specifies the JSP that is used to generate the channel content (by using the `getContent()` method).
`String name="editPage"`	Specifies the JSP that is used to generate the Edit page content (by using the `getEdit()` method).
`Boolean name="showExceptions"`	If `true`, makes SimpleWebServiceProvider show exceptions generated while processing the JSP as the channel output for the `getContent()` and `getEdit()` methods. This can be useful for developing and troubleshooting your portal.

`Boolean name="isDefaultAvailable"`	If `true`, the default value is available from the profile database.
`Collection name="defaultInput"`	Specifies the default input value.

SimpleWebServiceConfigurableProvider

SimpleWebServiceConfigurableProvider is similar to SimpleWebServiceProvider, except that it permits users to use the Edit function to change URLs and methods, hence, it is configurable.

The following is a list of the properties specific to SimpleWebServiceConfigurableProvider. This two column table lists the properties in the first (left) column and a brief description in the second (right) column.

`String name="wsdlURL"`	Specifies the URL to web service WSDL.
`String name="methodName"`	Specifies the web service method name that is going to be executed.
`Boolean name="isDefaultShowOutput"`	Specifies the default value. If `true`, the channel uses the default input value. If `false`, the channel uses the user input value.
`String name="contentPage"`	Specifies the JSP that is used to generate the channel content (by using the `getContent()` method).
`String name="editPage"`	Specifies the JSP that is used to generate the Edit page content (by using the `getEdit()` method).
`Boolean name="showExceptions"`	If `true`, makes SimpleWebServiceConfigurableProvider show exceptions generated while processing the JSP as the channel output for the `getContent()` and `getEdit()` methods. This can be useful for developing and troubleshooting your portal.
`Boolean name="isDefaultAvailable"`	If `true`, the default value is available from the profile database.
`Collection name="defaultInput"`	Specifies the default input value.

UserInfoProvider

UserInfoProvider collects information from the display profile and Identity Server software. It displays a greeting, the user's name, time zone, and locale, and has access to the user's IMAP and SMTP data.

The following is a list of the properties specific to UserInfoProvider. This two column table lists the properties in the first (left) column and a brief description in the second (right) column.

`Collection name="tags"`	Specifies the collection tags used in the user info template.
`String name="greeting"`	Specifies the tag name for greeting.
`String name="cn"`	Specifies the tag name for common name.
`String name="givenname"`	Specifies the given name.
`String name="sn"`	Specifies the surname.
`String name="uid"`	Specifies the user ID.
`String name="sunPortalNetmailIMAPServerName"`	Specifies the IMAP server name.
`String name="sunPortalNetmailSMTPServerName"`	Specifies the SMTP server name.
`String name="sunPortalNetmailIMAPUserid"`	Specifies the IMAP user ID.
`String name="sunPortalNetmailIMAPPassword"`	Specifies the IMAP password.
`String name="currentDate"`	Specifies the tag name for the current date.
`String name="timeLeft"`	Specifies the tag name for the time remaining.
`String name="maxIdle"`	Specifies the tag name for the maximum idle time.
`String name="preferredtimezone"`	Specifies the user's preferred time zone.
`String name="timezoneList"`	Specifies the tag name for the time zone.
`String name="locale"`	Specifies the user's locale.
`String name="localeList"`	Specifies the tag name for language.
`String name="preferredlocale"`	Specifies the user's preferred locale.
`String name="membershipNewPassword"`	Specifies the tag name for membership new password.

`String name="membershipConfirmPassword"`	Specifies the tag name for membership confirm password.
`String name="membershipOriginalPassword"`	Specifies the tag name for the original membership password.
`Collection name="tagModules"`	Specifies the collection of tag modules.
`String name="dp"`	Specifies the tag module to get and set user info channel properties.
`String name="attribute"`	Specifies the tag module to get and set user info channel attributes.
`String name="imappw"`	Specifies the tag module to get and set IMAP password.
`String name="datetime"`	Specifies the tag module to get current time.
`String name="timezone"`	Specifies the tag module to get and set time zone list.
`String name="localelist"`	Specifies the tag module to get and set local list.
`String name="membershippw"`	Specifies the tag module to get and set membership password.
`Collection name="authTypes"`	Specifies the collection of authentication types.
`String value="Membership"`	Specifies the value of authentication type.
`String name="netmailServiceName"`	Specifies the NetMail service name.

Display Profile Common Properties for Leaf Providers

The Portal Server software sample portal display profile XML fragments define the default values for all expected properties, so that channels that use the supplied providers do not have to define all the properties, only the ones that need to be different.

NOTE Depending on the merge priorities and locking assigned to the display profile documents that make up a user's display profile, the ultimate property values that are returned to a user's Desktop can change. See the *Portal Server Administration Guide* for more information on merging.

The following is a list of common properties for a <Provider> definition. In addition to the following properties, there are properties that are used in specific providers. For example, the lines and maxLines properties are required by the notes provider code.

The following two column table lists the general properties in the first (left) column and a brief description in the second (right) column.

title	Specifies the title that appears in the channel title bar in the Desktop.

description	The description is displayed on the content page to give the user a little more information about what the channel is more. For example:
	`title: Bookmarks`
	`description: manage your portal-specific bookmarks`

refreshTime	The refreshTime property controls how often a channel's content is reloaded.
editType	Specifies the edit type, either edit_complete or edit_subset. If edit_complete, the provider's getEdit() method is responsible for generating the complete Edit page content. If edit_subset, a generic edit provider container is used to put a frame around the Edit page. The provider's getEdit() page is then called, and displays the content of the Edit page.
isEditable	Determines if the provider has an edit view. If true, the Edit button is generated in the channel title bar. By clicking the Edit button, users can display an Edit page and customize channel settings such as whether the channel is minimized or detached. The Edit page is generated by the *Provider*.getEdit() method. If isEditable is false, no edit view is provided and no Edit button is generated in the title bar and users cannot change the settings for the channel. To implement an editable provider, the default no-operation implementations of ProviderAdapter.getEdit() and ProviderAdapter.processEdit() must be overridden. If the provider has the getEdit() and processEdit() methods defined, you can change the value of isEditable from false to true to cause the Edit button to appear in the channel title bar.

`width`	A channel's width setting is a suggestion for containing channels as to how much screen real estate the channel may require. This value is only a suggestion; a container is not required to utilize this value for its contained channels. Possible values are thin, thick, full_top, or full_bottom. In general, these values only make sense for an HTML-based Desktop.
`helpURL`	Specifies the online help URL, which can be either a fully qualified URL value or a relative path to the doc root location. For example, the online help URL for the bookmark channel is: `http://`*hostname*`:`*port*`/portal/docs/en/desktop/bkmark.html` This URL could also be defined as `desktop/bmark.html`. In this case, the provider context code figures out the doc root and the user locale, and locates the online help URL. A return value of null signifies that this provider does not have a help page. To have the provider code not generate a Help icon on the title bar for the channel, use a value of `""`.
`fontFace1`	Specifies the default font face for the channel, for example, `Sans-serif`.
`productName`	Specifies the name of the product, for example, Sun Java System Portal Server. This is not required by all providers. Required properties and their values for the `<Provider>` definition are based on the Provider interface. The required properties are necessary for the provider code if the provider class extends ProviderAdapter or the ProfileProviderAdapter class. Note that the channel can set its own properties that override these values.

Other Leaf Provider Display Profile Properties

authlessState

The `authlessState` property determines how client specific state is managed when the Desktop is operating in authless mode. Client specific state is accessed via the `ProviderContext.get`/`setClientProperty()` methods. The `authlessState` client type property can take on three values: client, server, and none. When set to:

- client, authless state is stored on the client.
- server, authless state is stored on the server.
- none, no authless state is recorded and the ProviderContext.get/setClientProperty() methods have no effect.

By default, the authlessState client type property is not present, and defaults to client for HTML devices, and none for non-HTML devices. To modify the default value for a specific client type, add the authlessState client type property and set its value to either client, server, or none.

encoderClassName

The encoderClassName client type property maps an encoding algorithm (class) to a specific client type. This information is used by the ProviderContext.escape() method to escape strings in a client type specific manner.

ConditionalProperties

This provides a generic operation for retrieving conditional properties. The most common conditions are locale and clientType, but the API is generic in that it allows you to define and base properties on any sort of condition.

In the administration console, the conditional properties are displayed as condition-value and can be edited like collections. The conditional properties can be nested and can be added to a channel or inside another conditional property. Use the Add Property page to add a new conditional property.

The <ConditionalProperties> tag must be used to define the filtering criteria. The tag contains the following required attributes:

condition	Specifies name of the filter
value	Specifies the value to be used in the filter

In the display profile, the <ConditionalProperties> tag can be defined as outlined in Code Example 2-2.

Code Example 2-2 <ConditionalProperties> Tag Usage Sample

```
<Properties>
    <String name="foo" value="bar">
    <ConditionalProperties condition="locale" value="de">
        <String name="foo" value="german bar">
        <String name="baz" value="a german baz value">
    </ConditionalProperties>
    <ConditionalProperties condition="client" value="nokia">
        <ConditionalProperties condition="locale" value="de">
            <String name="foo" value="nokia german bar">
        </ConditionalProperties>
    </ConditionalProperties>
</Properties>
```

Display Profile Channel Properties

The provider definition is the template that decides the properties for a channel. However, the display profile channel definition ultimately decides the values for the channel attributes. The display profile channel definition can define properties that overwrite the properties defined by the provider definition.

Container channels are channels that primarily generate its content by aggregating the content of other (its child) channels. A container channel allows for available and selected channel lists (see “Available and Selected List” on page 49) and can contain leaf channel definitions.

Both container and leaf channel properties can be configured from the administration console. See the provider-specific display profile properties for more information on the channel properties.

Chapter 3

Understanding the Sample Portal

This chapter describes the sample portal that you can choose to install on your system during the Sun Java System Portal Server installation. See the Installation Guide for more information on installing the sample portal.

This chapter assumes that the sample portal is installed. It contains the following sections:

- Overview of the Sample Portal
- JSP-Based Desktop
- Template-based Desktop
- Frame-based Desktop

Overview of the Sample Portal

The sample portal consists the following elements:

- Sample channels and containers (discussed in this chapter)
- Sample Portal templates and JSPs (see Appendix A, “Desktop Template Files” and Appendix C, “JavaServer Pages Reference”)
- Authless user definition and display profile XML fragment
- Authless templates and JSPs (see Appendix A, “Desktop Template Files” and Appendix C, “JavaServer Pages Reference”)

The Sample Desktop

This section provides an outline of the sample portal sample containers, each of which produces a different looking Desktop. A container is a channel that is able to hold and organize content from other channels.

Sample Containers for the Sample Portal

The sample portal is created from the following containers:

JSPTabContainer Creates a Desktop that contains multiple containers selected using different tabs. Normally, each tab is constructed by using the corresponding PredefinedTabPanelContainer. This container is JSP-based. Its provider is JSPTabContainerProvider.

JSPTableContainer Creates a Desktop that arranges a maximum of five sub-containers into the channel arrangement. This container is JSP-based. Its provider is JSPTableContainerProvider.

TemplateTabContainer Creates a Desktop that contains multiple containers selected using different tabs. Normally, each tab is constructed by using the corresponding PredefinedTemplatePanelContainer. This container is template-based. Its provider is TemplateTabContainerProvider.

TemplateTableContainer Creates a Desktop that arranges a maximum of five sub-containers into the channel arrangement. This container is template-based. Its provider is TemplateTableContainerProvider.

FrameTabContainer Creates a Desktop using frames. The left-hand frame enables you to navigate, and the right-hand frame displays the channels. This container is JSP-based. Its provider is JSPTabContainerProvider.

For more information on each container, see the appropriate section in this chapter.

Deriving the Sample Desktop

The container architecture uses the various containers, and JavaServer Pages™ (JSP™) or template files, to display the Desktop, and to build a hierarchy of containers that organize the Desktop content. Channels are defined at the global display profile level and are referenced by all the containers.

Determining the User's Default Desktop

The Portal Server software determines the user's default container by examining the Default Channel Name attribute for the Desktop page in the Identity Server software administration console.

When you install the sample portal, by default, the default channel is set to JSPTabContainer. Users can view this portal by typing:

`http://`*hostname*`:`*port*`/portal/dt`

You can easily view each sample portal Desktop by typing in the appropriate URL.

NOTE The Desktop service uses a dynamic attribute, `SunPortalDesktopDefaultChannelName`, to specify the default channel to execute when accessing the Desktop. This attribute is displayed as Default Channel Name in the administration console. This attribute is only used when the `provider=name` URL parameter is not specified. The default channel service attribute can be overridden by passing to the Desktop servlet the provider parameter and setting it to the name of a channel. Once the `provider=parameter` is passed to the Desktop, this value becomes the new default channel. The service attribute is no longer used for the duration of the current user's session.

Setting the service attribute for the default channel is useful in simple scenarios when you need to set a default channel per organization or per role. In situations where you need to set the default channel based on some programmatic logic, you should use a routing container.

A routing container is a channel that reads a display profile property specifying the user's preferred default Desktop channel to determine the Desktop to present. See the *Portal Server Developer's Guide* for more information on developing a custom routing container.

Guidelines for Using the Sample Portal

Editing the Default Sample Portal Files

Do not directly edit any of the files that make up the sample portal (display profile XML, JSP, and template files). Instead, make a copy of the sample portal to a new directory and then modify those copied files. In this way you preserve the integrity of the sample portal. Additionally, if you later apply a patch to the portal server, you won't loose any changes you might have made to the sample portal files, as the patch would only overwrite the initially installed sample files.

Changing the Desktop Type

You should also create a custom Desktop type for your users. The Desktop type attribute of the Desktop service is a comma-separated string. It is still a string type, but the Desktop uses it as an ordered Desktop type list. The list is used by the Desktop lookup operation when searching for templates and JSPs. The lookup starts at the first element in the list and each element represents a sub directory under the Desktop template base directory. If a template is not found in the first directory, then it proceeds to the next one in the list. This continues until the item is found (or not), for all Desktop type elements in the list.

If the default directory is not included in the list, it will be added at the end of the list implicitly. For example, if the Desktop type is `sampleportal`, the target template will be searched in the `sampleportal` sub directory, then the `default` sub directory.

By default, if the sample portal is installed, then the Desktop type attribute, `sunPortalDesktopType`, is set to `sampleportal`, meaning files are retrieved from the `sampleportal` subdirectory. If the sample portal is not installed, then the Desktop type attribute value is set to `default`. The authless user is created as part of the sample portal, and the Desktop type for the authless user is set to `anonymous,sampleportal`.

You can define a new set of templates by creating a new directory under the `/etc/opt/SUNWps/desktop/` directory, placing your template files in this directory, and making this directory the Desktop Type attribute for that organization.

➤ **To change the Desktop type**

1. Create a new subdirectory in the `/etc/opt/SUNWps/desktop` directory, or whatever directory `templateBaseDir` specifies in the `desktopconfig.properties` file.

   For example:

   ```
   mkdir /etc/opt/SUNWps/desktop/sesta
   ```

2. Manually copy only the template files that you wish to modify to the new directory location.

   For example, if your Desktop type will modify `content.jsp` file for JSPProvider, copy this file to `/etc/opt/SUNWps/desktop/sesta/JSPProvider/content.jsp`, and customize the file for the new Desktop type in that location.

You only need to copy the files that you have changed from the sample installation to the new directory tree. This structure enables you to tell at a glance which files have been modified from the original distribution. It also eliminates the need to back up copies of the original sample files.

3. Use the Identity Server software administration console to change the value of the Desktop Type attribute for the subdirectory created in Step 1.

   As this attribute is dynamic, you need to change it everywhere that it appears (organization, sub-organization, role, and user). Changing the Desktop Type at the organization level will not necessarily be reflected at the user level. This will be the case only if the user has not overwritten the Desktop Type in which case the Desktop Type value will be inherited from the organization level. If the user defines the Desktop Type at the user level, the value will remain the same even if the Desktop Type is changed at the organization level.

   In this example, in the administration console, you would specify `sesta,sampleportal` as the value for the `sunPortalDesktopType` attribute.

See the *Portal Server Administration Guide* for more information on editing Desktop attributes.

Restoring the Default Sample Portal Settings

To re-load the default (original) display profile for the sample portal providers, the `dp-providers.xml` and `dp-org.xml` files (in *portal-server-install-root*`/SUNWps/samples/desktop` directory) must be reloaded. The `dp-providers.xml` file goes in the global level and the `dp-org.xml` file goes in the organization level. For example, type:

***portal-server-install-root*/SUNWps/bin/dpadmin modify -u "uid=amAdmin,ou=People,dc=sesta,dc=com" -w *password* -g *portal-server-install-root*/SUNWps/samples/desktop/dp-providers.xml**

***portal-server-install-root*/SUNWps/bin/dpadmin modify -u "uid=amAdmin,ou=People,dc=sesta,dc=com" -w *password* -d "dc=sesta,dc=com" *portal-server-install-root*/SUNWps/samples/desktop/dp-org.xml**

Sample Portal Installation Directories

If you choose to install the sample portal, the installer locates the appropriate files in the following directories:

***portal-server-install-root*/SUNWps/samples/desktop**
This directory contains the following display profile documents:

`dp-org.xml`	Contains the display profile definitions for channels and containers.
`dp-anon.xml`	Contains the display profile definitions for channels and containers for the authlessanonymous and anonymous users in the default organization.

/etc/opt/SUNWps/desktop/sampleportal
Contains the JSP, template, and other support files for the Portal Server software sample portal.

/etc/opt/SUNWps/desktop/anonymous
Contains the JSP, template, and other support files for the Portal Server software Desktop anonymous user.

NOTE The sample portal has a dependency on the base Desktop and other components and cannot be installed if the base Desktop and other components are not installed. The base Desktop and other components are installed in the `/etc/opt/SUNWps/desktop/default` directory.

JSP-Based Desktop

Desktops based on JSPs enable a customization process without the necessity of changing the provider Java classes. The implementation of the JSP-based Desktop uses a tag library which Portal Server software supplies (see Appendix D for more information on the tags). Not all Desktop channels need to be JSP-based.

JSPTabContainer

The JSPTabContainer provides a JSP-based tabbed Desktop.

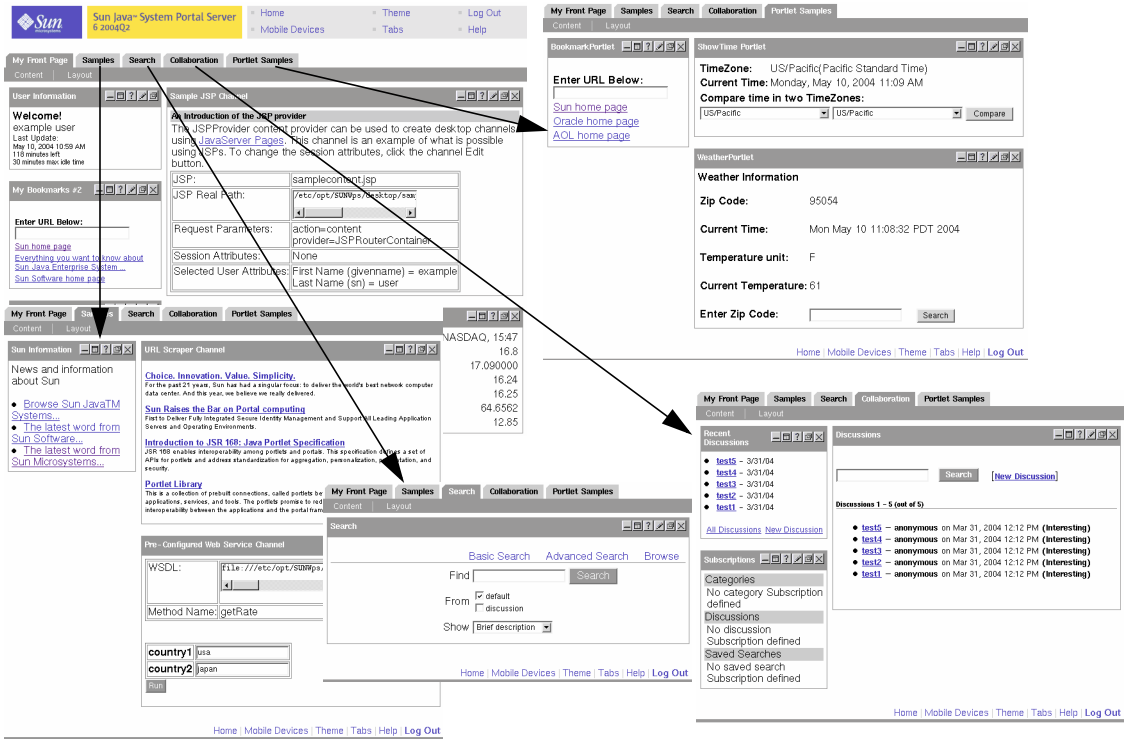
Sample Desktop

Default Layout

By default, the sample portal Desktop based on the JSPTabContainer (see Figure 3-1 on page 89) includes five tabs, My Front Page, Samples, Search, Collaboration, and Sample Portlet which includes the following channels:

- In the My Front Page tab: Login or User Information, Sun Information, Bookmark, Sample JSP, and XML Test channels
- In the Samples tab: Sun Information, URL Scraper, and Notes channels
- In the Search tab: Search channel
- In the Collaboration tab: Discussions Lite and Discussions channels
- In the Portlet Samples tab: Bookmark, Showtime, and Weather portlet channels

Figure 3-1 Sample Desktop Based on JSPTabContainer



Default Actions

The sample JSPTabContainer channel, by default, includes:

- Banner links to return Home, tabs to allow the user to remove, rename or select the start tab, and also create a new tab (the URL for this page is `action=edit&provider=JSPTabContainer`), theme to allow the user to set the color scheme and font type for the Desktop (the URL for the preset theme page is `action=edit&provider=JSPPresetThemeContainer`), help that displays the Desktop sample online help (the URL is `../docs/locale/desktop/helppage.htm`), Log Out link to log the user out of the Desktop (the URL is `action=logout`), and Search to allow the user to search.

  When you click the Tabs link in the Desktop, the Current Tabs Settings Edit page, where you can make changes, is displayed. Start Tab lets you set the starting tab; Tab Name specifies the name of the tabs in the container; and Action lets you rename or delete a tab from the Desktop. (JavaScript handles the action.)

  When you click Make a New Tab, the corresponding Edit page is shown. You decide what to name the tab and what the tab topics are. Content Page is displayed only when making a new tab from scratch. When other Tab Topics are selected, a new tab which looks similar to the TabTopic selected, is created and displayed.

- Links specific to the contained containers. The channels in each tab depend on the contained container of the JSPTabContainer. In the Sample Portal, these contained containers are JSPTableContainer and the channels are dependant on this container; but this does not have to be the case, they can be any container. The Content and Layout links provide the ability to customize the current selected contained container.

- Content and layout links. The top-most JSP in the table container defines the Content and Layout links. JSPContentContainer is the container that displays the Content page, and JSPLayoutContainer is the container that displays the Layout page.

Default Display Profile Settings

The provider responsible for generating the JSPTabContainer channel is JSPTabContainerProvider. The provider profile is the template which decides the properties for a container channel, but the container channel profile will ultimately decide the values for the container channel attributes.

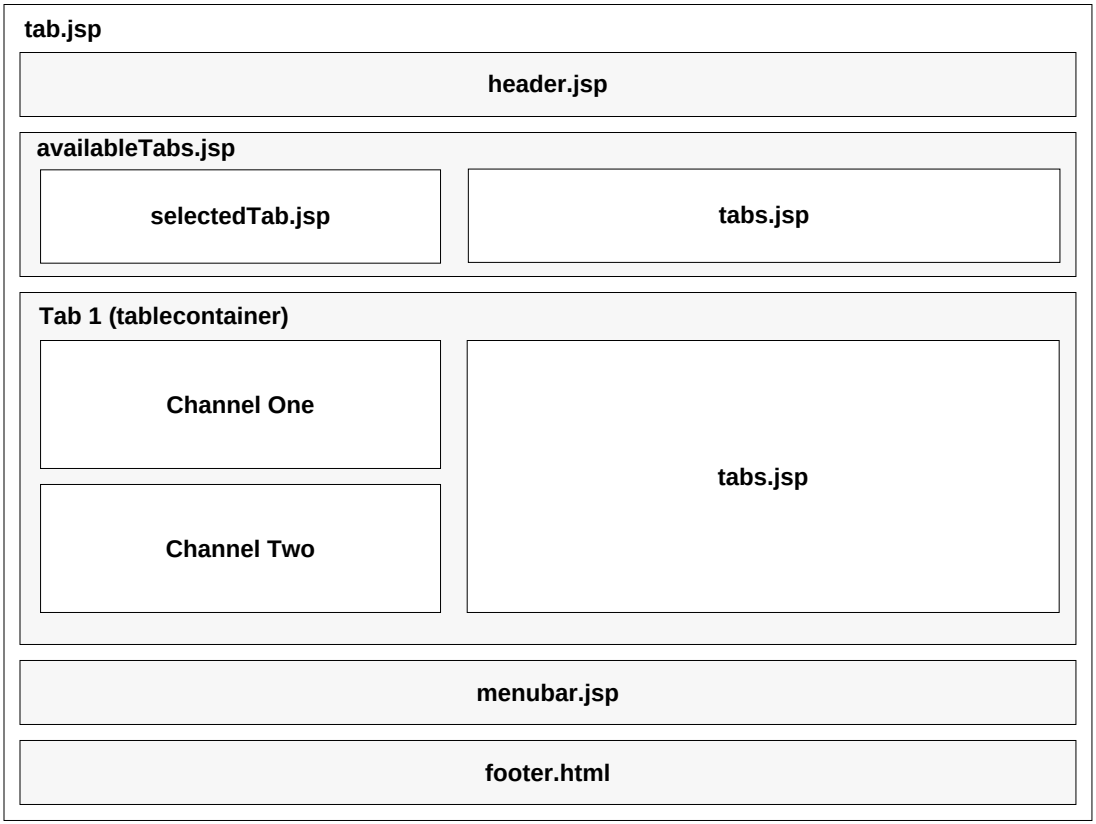
The properties that make up JSPTabContainer work as follows by default.

`contentPage`	Set to `tab.jsp`. This draws the Content Page for the tab container.
`editPage`	Set to `tabedit.jsp`. This displays the Edit page for the tab container where new tabs can be added, and existing tabs removed or renamed.
`startTab`	Sets the tab that opens first on the Desktop as MyFrontPageTabPanelContainer.
`maxTabs`	Allows six tabs to be created. As there are currently five tabs, one more can be added.
`makeTabProvider`	Used when creating a new tab from scratch.
`channelNumber`	Specifies that a number is appended to a newly created tab as the channel name. This number is increased each time a new tab is created, so that the new tab will have unique name. For example, to create a new tab based on MyFrontPageTabPanelContainer in JSPTabContainer, the new tab channel name would be JSPTabContainer / MyFrontPageTabPanelContainer1. (The new tab name is actually the `channelName` property in the display profile plus the value of the `channelNumber` property. The `channelNumber` is incremented by one each time a new tab is created.)
`contentChannel`	Specifies JSPContentContainer as the content channel that provides the Content page displaying channels to add to a user-created tab.
`presetThemeChannel`	Specifies JSPPresetThemeContainer as the channel that is displayed in the Theme - Preset Theme page.
`customThemeChannel`	Specifies JSPCustomThemeContainer as the channel that is displayed in the Theme - Custom Theme page.
`TabProperties`	This collection has `<Collection name=>` entries for each of the available tab defined in JSPTabContainer.
`Available`	This list describes all available channels for this container. The available channels are displayed in the Content Preference page for users to select from.
`Selected`	This list describes selected channels for this container. Only selected channels are displayed on the Desktop.

JSPTabContainer Architecture

Figure 3-2 shows the JSPTabContainer architecture. In this figure, `tab.jsp` is the top-level JSP file. The `tab.jsp` file makes include calls to the `header.jsp`, `availableTabs.jsp`, `menubar.jsp`, and `footer.html` files. The `availableTabs.jsp` file makes an include call to the `selectedTab.jsp` and `tabs.jsp` files.

Figure 3-2 JSPTabContainer Architecture



JSP Files Used by JSPTabContainer

The Portal Server software uses JSP files for a channel's presentation layer. JSPTabContainer references two main JSPs, `tab.jsp` and `tabedit.jsp`, through the `contentPage` and `editPage` properties.

Content template is responsible for the front page of the container channel and the file name for the tab container channel is `tab.jsp`. The `tab.jsp` file extensively uses the Desktop taglibs.

The Edit page is where you can add, remove, and rename tabs. The `tabedit.jsp` is used to display this page.

JSPTableContainer

The JSPTableContainer provides a JSP-based table Desktop.

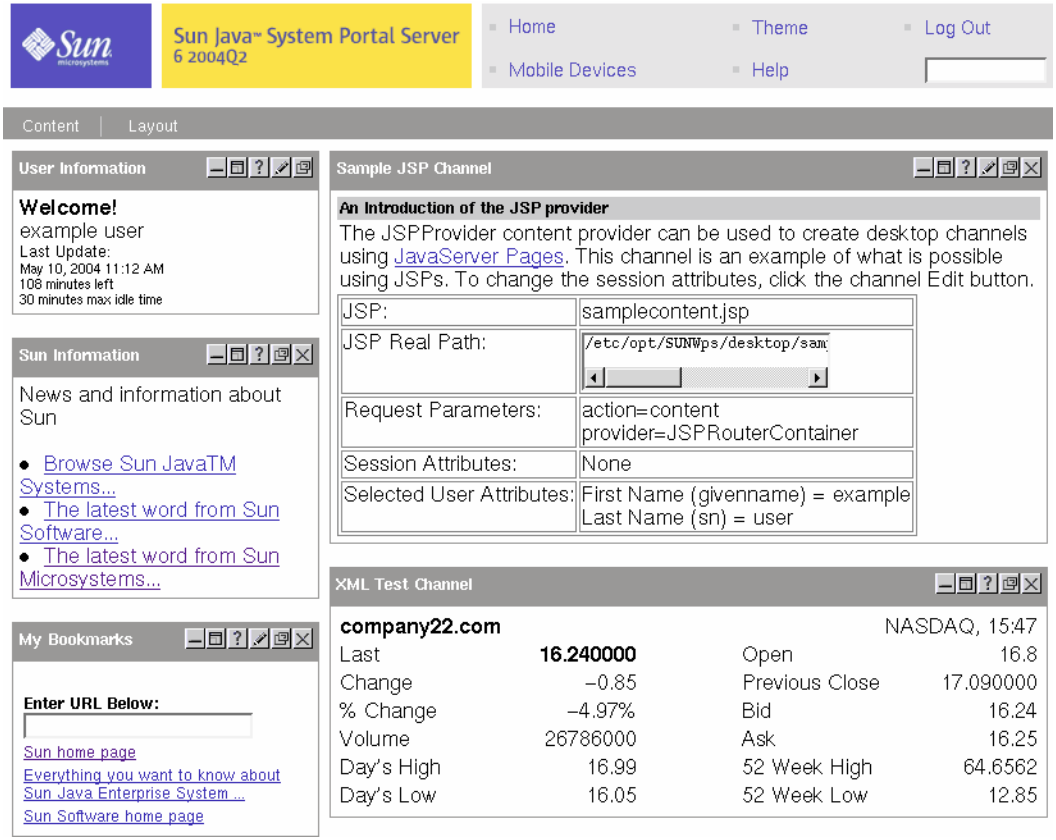
Sample Desktop

Default Layout

By default, the sample portal Desktop based on the JSPTableContainer (see Figure 3-3 on page 93) contains the following channels:

- Thin channels: User Information, Sun Information, My Bookmarks, Mailcheck Provider, and My Applications
- Wide channels: Sample JSP, XML Test, Notes, Personal Notes, and Preconfigured Web Service

Figure 3-3 Sample Desktop Based on JSPTableContainer



Default Actions

The sample JSPTableContainer channel, by default, includes:

- Banner links to return to the Desktop Home page, Desktop theme to allow the user to set the color scheme and font type for the Desktop (the URL for this page is `action=edit&provider=JSPPresetThemeContainer`), Log Out to allow the user to log out of the Desktop (the URL for this page is `action=logout`), Help to display the Desktop sample online help (the URL for this page is `../docs/locale/desktop/helppage.htm`), and Search to allow the user to search.
- Leaf channel. JSPTableContainer does not contain any contained containers, it only has leaf channels. This container uses JSPContentContainer and JSPLayoutContainer to edit the content and layout, respectively.

- Content and layout links. The `toptable.jsp` file defines the Content and Layout links. JSPContentContainer is the container that displays the Content page, and JSPLayoutContainer is the container that displays the Layout page.

  The Content link (`action=edit&provider=JSPContentContainer&container=JSPTableContainer`) allows the user to edit the content on the Content page and the Layout link (`action=edit&provider=JSPLayoutContainer&container=JSPTableContainer`) allows the user to edit the layout of the channels on the Layout page.

Default Display Profile Settings

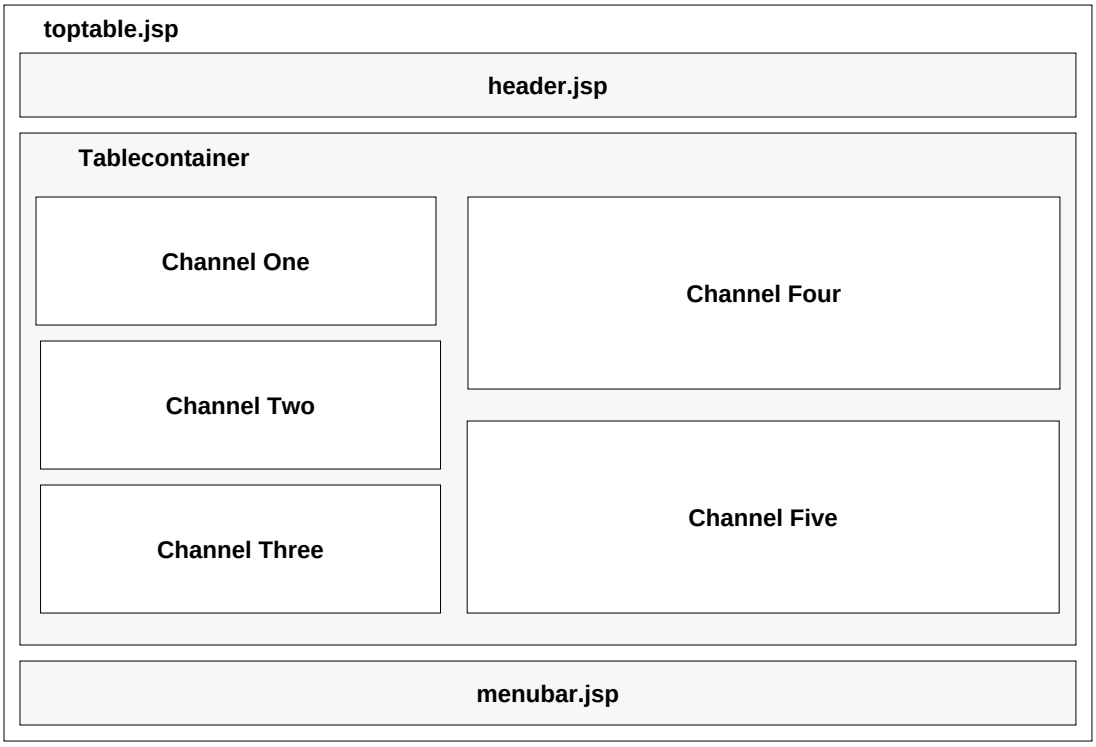
The provider responsible for generating the JSPTableContainer channel is JSPTableContainerProvider. The provider profile is the template which decides the properties for a container channel, but the container channel profile will ultimately decide the values for the container channel attributes. The default properties that make up JSPTableContainer work as follows:

`contentPage`	Set to `toptable.jsp`. This draws the Content Page for the table container.
`categories`	This collection defines the categories under which the available channels in the JSPTabContainer will be grouped in the Content page. Here there are three categories: Personal Channels, Sample Channels, and News Channels.
`channelsRow`	This collection and values that appear in this collection, contain the row number value for channels in this container. For example, the mail check channel is defined as row 4.
`channelsIsRemovable`	This collection defines a collection to contain the `isRemovable` value for channels in this container. Only one channel, user information, is defined, with a value of false, so that it cannot be removed.
`Available`	This list describes all available channels for this container. The available channels are displayed in the content preference page for users to select from.
`Selected`	This list describes selected channels for this container. Only selected channels shows up in the Desktop.

JSPTableContainer Architecture

Figure 3-4 shows the JSPTableContainer architecture. In this figure, `toptable.jsp` is the top-level JSP file. The `toptable.jsp` file makes include calls to the `header.jsp`, `launchPopup.jsp`, `leafWrapper.jsp`, and `menubar.jsp` files.

Figure 3-4 JSPTableContainer Architecture



JSP Files Used by JSPTableContainer

The Portal Server uses JSP files for a channel's presentation layer. JSPTableContainer references one main JSP, `toptable.jsp`, through the `contentPage` property.

Content template is responsible for the front page of the container channel and the file name for the tab container channel is `toptable.jsp`. The `toptable.jsp` file extensively uses the Desktop taglibs.

Template-based Desktop

TemplateTabContainer and TemplateTableContainer use the original template mechanism from iPlanet™ Portal Server 3.0. TemplateTableContainer is the same as the normal Desktop from iPlanet Portal Server 3.0, and TemplateTabContainer is the same as the tabbed Desktop from iPlanet Portal Server 3.0.

Frame-based Desktop

FrameTabContainer

The FrameTabContainer provides a frame-based table Desktop.

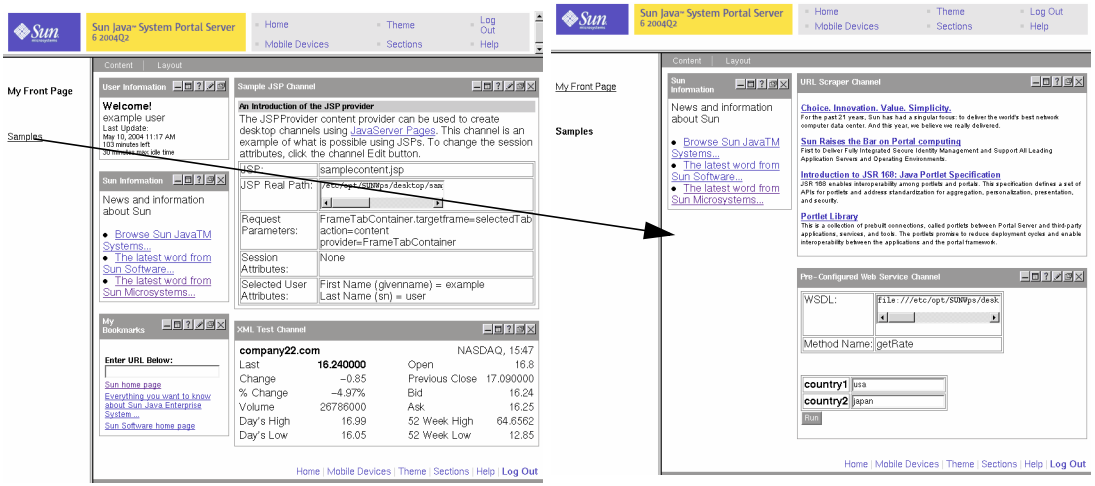
Sample Desktop

Default Layout

By default, the sample portal Desktop based on the FrameTabContainer provides a frame-based table Desktop consisting of two frames, My Front Page and Samples, with the following channels:

- In the My Front Page frame: User Information, Sun Information, My Bookmarks, Mailcheck Provider, My Applications, Sample JSP, and XML Test
- In the Samples frame: Sun Information, URL Scraper, Notes, and Preconfigured Web Service

Figure 3-5 Sample Desktop Based on FrameTabContainer



Default Actions

The sample FrameTabContainer channel, by default, includes:

- Banner links to return to the Desktop Home page, Sections (`action=edit&provider=FrameTabContainer`) that allow users to rename or select the start page, delete a page, and also create a new page, theme (`action=edit&provider=JSPPresetThemeContainer`) that allows users to set the color scheme and font type for the Desktop, Help (`../docs/locale/desktop/helppage.htm`) to display the Desktop sample online help, and Log Out (`action=logout`) that logs the user out from the Desktop.

  The Sections link on the Content page displays the Current Page Settings Edit page where you can make changes. Here, Start page allows the user to set the starting page, Rename allows the user to rename the page, and Delete allows the user to delete a page from the Desktop.

- Content and layout links. The top-most JSP in the table container defines the Content and Layout links. JSPContentContainer is the container that displays the Content page, and JSPLayoutContainer is the container that displays the Layout page.

The Content link (`action=edit&provider=JSPContentContainer&container=MyFrontPageFramePanelContainer`) allows the user to edit the content for this particular page on the Content page and the Layout link (`action=edit&provider=JSPLayoutContainer&container=MyFrontPageTabPanelContainer&selected=MyFrontPageFramePanelContainer`) allows the user to edit the layout of the channels for this particular page on the Layout page.

Default Display Profile Settings

The provider responsible for generating FrameTabContainer channel is JSPTabContainerProvider. The provider profile is the template which decides the properties for a container channel, but the container channel profile will ultimately decide the values for the container channel attributes. The properties that make up FrameTabContainer work as follows:

`contentPage`	Set to `frametab.jsp`. This draws the Content Page for the frame container.
`editPage`	Set to `frametabedit.jsp`. This displays the Edit page for the frame container where new pages can be added, or existing pages removed or renamed.
`startTab`	Sets the page that opens first on the Desktop as MyFrontPageFramePanelContainer.
`maxTabs`	Allows four pages to be created. As, by default, there are two pages, two more can be added.
`makeTabProvider`	Specifies JSPFrameCustomTableContainerProvider as the provider to create a new page on the Desktop.
`channelNumber`	Specifies that a number is appended to a newly created page as the channel name. This number is increased each time a new page is created, so that the new page will have unique name. For example, to create a new page based on MyFrontPageFramePanelContainer in FrameTabContainer, the new page channel name would be `FrameTabContainer/MyFrontPageFramePanelContainer1`. (The new page name is actually the `channelName` property in the display profile plus the value of `channelNumber` property. The `channelNumber` property value is incremented by one each time a new page is created.)

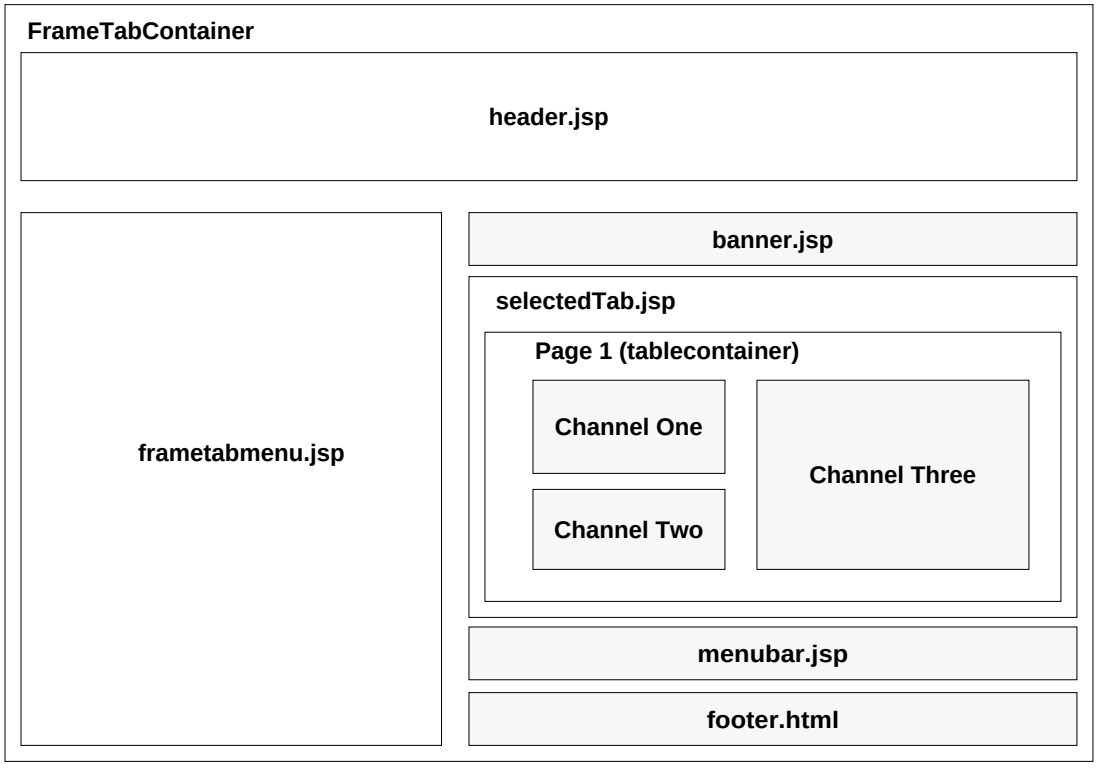
`contentChannel`	Specifies JSPContentContainer as the content channel that provides the Content page displaying channels to add to a user-created page.
`TabProperties`	This collection has entries for each of the four containers that are available in the FrameTabContainer.
`Available`	This list describes all available channels for this container. The available channels are displayed in the content preference page for users to select from.
`Selected`	This list describes selected channels for this container. Only selected channels shows up in the Desktop.

FrameTabContainer Architecture

Figure 3-6 on page 100 shows the FrameTabContainer architecture. In this figure, `frametab.jsp` is the top-level JSP file. The `frametab.jsp` file makes include calls to the `frametabmenu.jsp`, `header.jsp`, `banner.jsp`, `selectedTab.jsp`, `menubar.jsp`, `frameset.jsp`, and `footer.html` files.

FrameTabContainer is made up of two sub-containers, MyFrontPageFramePanelContainer and SamplesFramePanelContainer, as represented by Page 1 (tablecontainer) in the figure.

Figure 3-6 FrameTabContainer Architecture



JSP Files Used by FrameTabContainer

The Portal Server uses JSP files for a channel's presentation layer. FrameTabContainer references two main JSPs, `frametab.jsp` and `frametabedit.jsp`, through the `contentPage` and `editPage` properties.

Content template is responsible for the front page of the container channel and the file name for the tab container channel is `frametab.jsp`. The `frametab.jsp` file extensively uses the Desktop taglibs.

The Edit page is where you can add, remove, and rename pages. The `frametabedit.jsp` is used to display this page.

Chapter 4

Internally Used Containers

Often, when working with the sample portal, you need to modify the appropriate "contained" container, which is part of the top-level container. The contained containers are:

- MyFrontPageFramePanelContainer (parent = FrameTabContainer)
- MyFrontPageTabPanelContainer (parent = JSPTabContainer)
- MyFrontPageTemplatePanelContainer (parent = TemplateTabContainer)
- SamplesFramePanelContainer (parent=FrameTabContainer)
- SamplesTabPanelContainer (parent = JSPTabContainer)
- SearchTabPanelContainer (parent = JSPTabContainer)

Sun Java System Portal Server software uses other container providers internally to perform such tasks as creating new tabs and edit containers.

Table 4-1 Internally Used Containers

Container Name	Description
JSPTabCustomTableContainerProvider	JSPTabCustomTableContainerProvider is used when a new tab is created in the user's JSP tab-based Desktop, and is specified in the `makeTabProvider` property of JSPTabContainer. JSPTabCustomTableContainerProvider is based on the JSP table container provider.
JSPFrameCustomTableContainerProvider	JSPFrameCustomTableContainerProvider is used when a new frame is created in the user's JSP frameset-based Desktop, and is specified in the makeTabProvider property. JSPFrameCustomTableContainerProvider is based on the table container provider.
TemplateEditContainerProvider	TemplateEditContainerProvider is used by the template-based containers (TemplateTabContainer and TemplateTableContainer) as their edit provider. If a channel's editType is EDIT_SUBSET, this provider is used to draw the frame for the Edit page.

Table 4-1 Internally Used Containers

Container Name	Description
TemplateTabCustomTableContainerProvider	TemplateTabCustomTableContainerProvider is used when a new tab is created in the user's template-based tab Desktop. TemplateTabCustomTableContainerProvider is based on the template-based table container provider.
PredefinedFrontPageFramePanelContainerProvider	PredefinedFrontPageFramePanelContainerProvider is the provider for the predefined tab for MyFrontPage tab when the user creates a new page based on an existing page from the make New Page page on FrameTabContainer.
PredefinedSamplesFramePanelContainerProvider	PredefinedSamplesFramePanelContainerProvider is the provider for the predefined tab for Samples tab when the user creates a new page based on an existing page from the make New Page page on FrameTabContainer.
PredefinedToolsTemplatePanelContainerProvider	PredefinedToolsTemplatePanelContainerProvider is the provider for the Tools tab on TemplateTabContainer.
PredefinedFrontPageTemplatePanelContainerProvider	PredefinedFrontPageTemplatePanelContainerProvider is the provider for the predefined tab for MyFrontPage tab when the user creates a new tab based on an existing tab from the make New Tab page on TemplateTabContainer.
PredefinedSamplesTemplatePanelContainerProvider	PredefinedSamplesTemplatePanelContainerProvider is the provider for the predefined tab for Samples tab when the user creates a new tab based on an existing tab from the make New Tab page on TemplateTabContainer.
PredefinedSamplesTabPanelContainerProvider	PredefinedSamplesTabPanelContainerProvider is the provider for the predefined tab for Samples tab when the user creates a new tab based on an existing tab from the make New Tab page on JSPTabContainer.
PredefinedFrontPageTabPanelContainerProvider	PredefinedFrontPageTabPanelContainerProvider is the provider for the predefined tab for MyFrontPage tab used when the user creates a new tab based on an existing tab from the make New Tab page on JSPTabContainer.

The predefined container providers are not, by default, directly used for display on the sample portal.

Chapter 5

Customizing Container Tabs

This chapter provides a variety of tasks to customize the container tabs. It contains the following sections:

- Adding a Tab to JSPTabContainer
- Creating a Tab Within a Tab
- Stretching a Tab Across an Entire Container
- Changing the Tab Image for JSP-based Tab Containers
- Changing the Color of Tabs
- Making a Tab the Start Tab
- Adding a Role-Based Tab
- Adding a Channel to a User-defined Tab

NOTE The order that you list the tabs in the display profile is the order that tabs are displayed in the Desktop. So, to make a tab the first tab in the user's Desktop, you need to move it to be first in the selected list in the display profile.

Adding a Tab to JSPTabContainer

A tab can be any container type, but, by default, the sample portal uses table container. To add a new tab, you must first define the container, then register that container in JSPTabContainer, which "houses" the tabs.

➤ **To Add a Tab to JSPTabContainer**

1. Create the necessary display profile.

a. Define the new collection within `<Collection name="TabProperties">` in JSPTabContainer, for example:

```
...
    <Collection name="NewTabPanelContainer">
        <Boolean name="removable" value="false"/>
        <Boolean name="renamable" value="true"/>
        <Boolean name="predefined" value="true"/>
    </Collection>
    ...
</Collection>
...
```

b. Add entries to the `<Available>` and `<Selected>` tags, for example:

```
...
    <Available>
        <Reference value="NewTabPanelContainer"/>
        ...
    </Available>
...
    <Selected>
        <Reference value="NewTabPanelContainer"/>
        ...
    </Selected>
...
```

c. Define a container for NewTabPanelContainer, for example:

```
<Container name="NewTabPanelContainer"
provider="JSPTableContainerProvider">
    <Properties>
        <String name="title" value="New Container Channel"/>
        <String name="contentPage" value="tabtable.jsp"/>
        <String name="description" value="This is a test for front table
containers"/>
        <String name="Desktop-fontFace1" value="Sans-serif"/>
        <Collection name="categories">
            <String value="Personal Channels"/>
            <String value="Sample Channels"/>
        </Collection>
        <Collection name="Personal Channels">
            <String value="UserInfo"/>
```

```
            <String value="MailCheck"/>
        </Collection>
        <Collection name="Sample Channels">
            <String value="SampleJSP"/>
            <String value="SampleXML"/>
        </Collection>
    </Properties>
    <Available>
        <Reference value="UserInfo"/>
        <Reference value="MailCheck"/>
        <Reference value="SampleJSP"/>
        <Reference value="SampleXML"/>
    </Available>
    <Selected>
        <Reference value="UserInfo"/>
        <Reference value="MailCheck"/>
        <Reference value="SampleJSP"/>
        <Reference value="SampleXML"/>
    </Selected>
    <Channels>
        ...
    </Channels>
</Container>
```

d. If predefined property value is true in the `TabProperties` collection (Step a on page 106), then it is recommended to define a Provider for the container channel which is meant to be used as a predefined tab.

For example:

Code Example 5-1 PredefinedNewTabPanelContainerProvider Display Profile Definition

```
<Provider name="PredefinedNewTabPanelContainerProvider"
class="com.sun.portal.providers.containers.jsp.table.JSPTableContainerProvider"
version="2">
    <Properties>
        <ConditionalProperties condition="locale" value="en" >
            <ConditionalProperties condition="locale" value="US" >
                <String name="title" value="New Sample"/>
                <String name="description" value="New Tab"/>
            </ConditionalProperties>
        </ConditionalProperties>
        <String name="title" value="New Sample"/>
        <String name="description" value="New Tab"/>
        <String name="contentPage" value="tabtable.jsp"/>
        <String name="presetThemeChannel" value="JSPPresetThemeContainer" advanced="true"/>
        <String name="customThemeChannel" value="JSPCustomThemeContainer" advanced="true"/>
        <String name="parentTabContainer" value="JSPTabContainer" advanced="true"/>
        <String name="Desktop-fontFace1" value="Sans-serif"/>
```

Code Example 5-1 PredefinedNewTabPanelContainerProvider Display Profile Definition *(Continued)*

```
        <String name="refreshTime" value="" advanced="true"/>
        <String name="width" value="thin" advanced="true"/>
        <String name="fontFace1" value="Sans-serif"/>
        <String name="productName" value="Sun Java System Portal Server"/>
        <String name="maximizedChannel" value=""/>
        <Integer name="timeout" value ="240"/>
        <Integer name="layout" value="1"/>
        <Boolean name="showExceptions" value="false"/>
        <Boolean name="parallelChannelsInit" value="false"/>
        <Boolean name="refreshParentContainerOnly" value="false" advanced="true"/>
        <Boolean name="isEditable" value="true" advanced="true"/>
        <String name="editType" value="edit_complete" advanced="true"/>
        <String name="editContainerName" value="JSPEditContainer" advanced="true"/>
        <Integer name="thin_popup_height" value="200"/>
        <Integer name="thin_popup_width" value="500"/>
        <Integer name="thick_popup_height" value="300"/>
        <Integer name="thick_popup_width" value="600"/>
        <Integer name="fullwidth_popup_height" value="500"/>
        <Integer name="fullwidth_popup_width" value="600"/>
        <Boolean name="defaultChannelIsMinimizable" value="true"/>
        <Boolean name="defaultChannelIsMaximizable" value="true"/>
        <Boolean name="defaultChannelIsMinimized" value="false" advanced="true"/>
        <Boolean name="defaultChannelIsDetached" value="false" advanced="true"/>
        <Boolean name="defaultChannelIsDetachable" value="true"/>
        <Boolean name="defaultChannelIsRemovable" value="true"/>
        <Boolean name="defaultChannelHasFrame" value="true" advanced="true"/>
        <Boolean name="defaultChannelIsMovable" value="true"/>
        <Boolean name="defaultBorderlessChannel" value="false" advanced="true"/>
        <String name="defaultChannelColumn" value="1" advanced="true"/>
        <String name="defaultChannelRow" value="1" advanced="true"/>
        <Collection name="categories">
            <String value="Sample Channels"/>
        </Collection>
        <Collection name="Sample Channels">
            <String value="SampleRSS"/>
            <String value="SampleURLScraper"/>
            <String value="Notes"/>
            <String value="SampleSimpleWebService"/>
        </Collection>
        <Collection name="channelsColumn" advanced="true">
            <String name="SampleURLScraper" value="2"/>
            <String name="Notes" value="2"/>
            <String name="SampleSimpleWebService" value="2"/>
        </Collection>
        <Collection name="channelsRow" advanced="true">
            <String name="SampleURLScraper" value="2"/>
            <String name="Notes" value="3"/>
            <String name="SampleSimpleWebService" value="4"/>
        </Collection>
        <Collection name="channelsIsMinimized" advanced="true"/>
        <Collection name="channelsIsDetached" advanced="true"/>
        <Collection name="channelsHasFrame" advanced="true"/>
        <Collection name="channelsIsMinimizable"/>
        <Collection name="channelsIsMaximizable"/>
```

Code Example 5-1 PredefinedNewTabPanelContainerProvider Display Profile Definition *(Continued)*

```
        <Collection name="channelsIsMovable"/>
        <Collection name="channelsIsRemovable"/>
        <Collection name="channelsIsDetachable"/>
        <Collection name="borderlessChannels"/>
    </Properties>
</Provider>
```

e. Define the container channel based on the PredefinedNewTabPanelContainerProvider.

When the user creates a new tab based on the predefined tab, all the properties for this tab are picked up from the Provider definition. For example:

Code Example 5-2 PredefinedNewTabPanelContainer Channel Properties

```
<Container name="PredefinedNewTabPanelContainer"
provider="PredefinedNewTabPanelContainerProvider">
    <Properties/>
    <Available>
        <Reference value="SampleRSS"/>
        <Reference value="SampleURLScraper"/>
        <Reference value="Notes"/>
        <Reference value="SampleSimpleWebService"/>
    </Available>
    <Selected>
        <Reference value="SampleRSS"/>
        <Reference value="SampleURLScraper"/>
        <Reference value="Notes"/>
        <Reference value="SampleSimpleWebService"/>
    </Selected>
    <Channels>
    </Channels>
</Container>
```

2. Load the display profile into LDAP by using the `dpadmin` command.

See "Editing the Display Profile" on page 38.

3. Bring up the Desktop and verify that the tab was added.

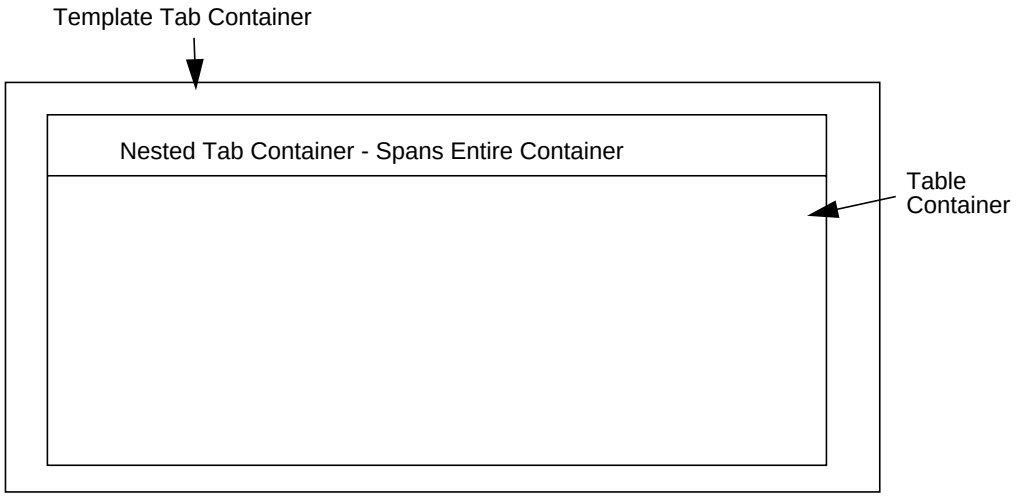
Creating a Tab Within a Tab

This is similar to “Adding a Tab to JSPTabContainer” on page 105, except that instead of defining the tab based on JSPTableContainerProvider, you base the new tab on JSPTabContainerProvider.

Stretching a Tab Across an Entire Container

Figure 5-1 shows an example of a tab that spans an entire container. In this figure, the top-level container is a template tab container. Nested within it is a table container, and nested within that is a tab container, in which the tab spans the entire container.

Figure 5-1 Tab Spanning Entire Container



➤ **To Stretch a Tab Across an Entire Container**

1. Edit the display profile and make the width of the Nested Tab Container full_top in the Table Container so that it stretches across the entire page.
2. Load the display profile into LDAP by using the `dpadmin` command.

   See “Editing the Display Profile” on page 38.

Changing the Tab Image for JSP-based Tab Containers

You can customize the look of tabs as they use images.

➤ **To Change the Tab Image for JSP-based Tab Containers**

1. Log in to the Sun Java System Identity Server administration console.
2. Select Services from your Organization View pull-down menu and select Portal Desktop.
3. Select Edit XML to directly edit the display profile XML fragment. Or,
   a. Select Manage Channels and Containers Link and Edit Properties for Display Profile.
   b. Select Global Themes and the theme you wish to modify.
4. Change the value of the `tabNotchImage` property to the new image name.

   By default, the value for this property is `tabNotch.gif`.
5. Copy the new image into *portal-server-install-root*`/SUNWps/web-src/desktop/tabs/images` directory.
6. Run the *portal-server-install-root*`/SUNWps/bin/deploy redeploy -deploy_admin_password` *password* command to deploy the new image.
7. Reload the Desktop to verify the change.

Changing the Color of Tabs

The background color of tabs are part of the themes.

➤ **To Change the Color of Tabs**

1. Log in to the Sun Java System Identity Server administration console.
2. Select Services from your Organization View pull-down menu and select Portal Desktop.
3. Select Edit XML to directly edit the display profile XML fragment. Or,
   a. Select Container and Channel Management Link and Edit Properties for Display Profile.

b. Select Edit Collection for Global Themes and the theme you wish to modify.

4. Change the value of the `titleBarColor` property to change the color of the selected tab and / or change the value of `tabColor` property to change the color of an unselected tab.

   The selected tab background color is the same as the title bar color.

5. Reload the Desktop to verify the change.

Making a Tab the Start Tab

The “Start tab” is the tab that is highlighted when user first logs in.

➤ **To make the tab the start tab**

1. Edit the display profile for the appropriate container.
2. Change the `startTab` property to the tab to highlight when the user logs in. For example:

   ```
   <String name="startTab" value="MyFrontPageTabPanelContainer"/>
   ```

3. Load the display profile into LDAP by using the `dpadmin` command.

   See “Editing the Display Profile” on page 38.

Adding a Role-Based Tab

When you have a display profile that provides a set of JSP nested tabs, and you want a user who belongs to this role and another role to add an additional sub-tab, you can merge multiple display profiles to accomplish this. In the additional display profile, you specify the additional sub-tab in the like-named container for JSPTabContainer with `merge=fuse` and JSPTableContainer for the new tab.

➤ **To Add a Role-based Tab**

1. To add a role-based tab for a user, define a role level display profile, which has the JSPTabContainer definition.

2. Add the role-based tab to the available and selected list with merge=fuse in the channel definition for the table container.

   When the user is added to this role, the new tab is visible. The following display profile XML fragments show the role definitions.

role1

```
<Container name="JSPTableContainer" provider="JSPTableContainerProvider"
merge="fuse">
    <Properties> ... </Properties>
    <Available> ... </Available>
    <Selected merge="fuse">
        <Reference value="Outages"/>
        <Reference value="SolarisAdmin"/>
        <Reference value="AdminTipoftheDay"/>
    </Selected>
</Container>
```

role2

```
<Container name="Front" provider="front" merge="fuse">
    <Properties> ... </Properties>
    <Available> ... </Available>
    <Selected merge="fuse">
        <Reference value="Benefits"/>
        <Reference value="EmployeeNews"/>
    </Selected>
</Container>
```

The user belonging to both role1 and role2 receives the following display profile:

```
Container name="JSPTableContainer" provider="JSPTableContainerProvider"
merge="fuse">
    <Properties> ... </Properties>
    <Available> ... </Available>
    <Selected merge="fuse">
        <Reference value="Outages"/>
        <Reference value="SolarisAdmin"/>
        <Reference value="AdminTipoftheDay"/>
        <Reference value="Benefits"/>
        <Reference value="EmployeeNews"/>
    </Selected>
</Container>
```

Adding a Channel to a User-defined Tab

Users can add a new tab to their Desktop by using the Tabs link and then by clicking the Make a New Tab link. The channel list that gets displayed on the content page which is shown when the user selects to create a new tab from scratch is picked up from the JSPTabCustomTableContainer's Available list.

Chapter 6

Customizing Channels

This chapter describes how to customize channels. This chapter contains the following sections:

- Customizing Channel Refresh Times and Container Caching
- Customizing Window Preference
- Removing a Button
- Changing the Channel Layout for a Table Container
- Removing the Title Bar from a Channel
- Changing the Channel Border Width
- Customizing the Channel Border

See the *Portal Server Administration Guide* for instructions on adding a channel to the Desktop.

Customizing Channel Refresh Times and Container Caching

The `refreshTime` property controls how often a channel's content is reloaded. When `refreshTime` is set to `0` (the default) for the container, the browser refresh (or reload) causes the page to be reloaded and the `getContent()` method is called again for every channel.

The following applies to a single channel:

- It is not possible to refresh only the content of the single channel within a container because a channel is an HTML table cell.

- It is possible to use the `DesktopURL()` method in the PAPI. The provider can use `getDesktopURL()` to get the Desktop servlet's URL, append arguments to it, and generate a new URL (or link).

The following applies to controlling and configuring container caching:

- Use the `refreshTime` property for the container along with the `refreshTime` for individual channels within the container.
- If the `refreshTime` for the container is blank, it is calculated to be the minimum time for all of the contained channels. If you want to override that calculated time, set a `refreshTime` for the container and then the content for the whole container will be cached.

NOTE If you have a large number of channels, utilize the provider caching by setting the `refreshTime` to a large number so that the portal page can use cached content. This makes sense when most of your channels have static content. The way the `refreshTime` works is if the container's `refreshTime` is set, it will use it. If `refreshTime` is set to an empty string, it will try to get and use the minimum of the `refreshTime` of its selected channels.

Customizing Window Preference

For channels that include links that launch another browser, you can control how this browser window is opened.

➤ To Customize the Channel Window Preference

1. Define the display profile (either for the channel, to make the change for only that channel, or for the provider, to make the change for every channel that uses the provider) so that it includes the windowPref property.

   For example:

```
<Properties>
    ...
    <String name="windowPref" value="all_new"/>
    ...
</Properties>
```

The values are:

- `all_new` (New window is opened for every link)
- `one_new` (All links open on the same new window)
- `same` (Desktop window)

2. Load the display profile into LDAP by using the `dpadmin` command.

   See "Editing the Display Profile" on page 38.

NOTE The intelligence has to be built with the help of JavaScript for that particular channel. Out of the box, this is supported by BookmarkProvider only.

Removing a Button

➤ **To Remove a Button From All Channels in a Container**

1. Find the container you want to work with. If you are working with one of the sample portals, you need to modify the appropriate "contained" container, which is part of the top-level container. See Chapter 4, "Internally Used Containers."

2. Add the appropriate property (within the `<Properties> </Properties>`) tags from Table 6-1 to the container's display profile for the button you want to remove. This two column table lists the button in the first column and the property to hide the button in the second column.

   The order of the buttons in this table corresponds to the order they appear in the channel, from left to right: Minimize, Maximize, Help, Edit, Detach, and Remove.

Table 6-1 Channel Buttons and Corresponding Properties

Button	Property to Hide the Button
Minimize	`<Boolean name="defaultChannelIsMinimizable" value="false"/>`
Maximize	`<Boolean name="defaultChannelIsMaximizable" value="false"/>`
Help	`<String name="helpURL" value=""/>`
Edit	`<Boolean name="isEditable" value="false"/>`

Table 6-1 Channel Buttons and Corresponding Properties *(Continued)*

Button	Property to Hide the Button
Detach	`<Boolean name="defaultChannelIsDetachable" value="false"/>`
Remove	`<Boolean name="defaultChannelIsRemovable" value="false"/>`

NOTE For the Help and Edit buttons, You must insert the respective property for each channel. You cannot insert the property within the container's `<Properties> </Properties>` tags.

Make sure the following properties are not defined in the container:

```
<Collection name="channelsIsRemovable">..</Collection>
<Collection name="channelsIsMinimizable"/>..</Collection>
<Collection name="channelsIsMaximizable"/>..</Collection>
<Collection name="channelsIsDetachable"/>..</Collection>
```

3. Load the display profile into LDAP by using the `dpadmin` command.

   See "Editing the Display Profile" on page 38.

➤ To Remove a Button From a Single Channel

1. For the channel from which you want to remove a button, add the appropriate property to a `Collection` tag in the container that contains the channel. See Table 6-2 on page 119, for the button you want to remove. This two column table lists the button in the first column and the property to hide the button in the second column

   The order of the buttons in this table corresponds to the order they appear in the channel, from left to right: Minimize, Maximize, Help, Edit, Detach, and Remove.

Table 6-2 Channel Buttons and Corresponding Properties

Button	Property to Hide the Button

Minimize	`<Collection name="channelsIsMinimizable">`
	`<Boolean name="channelname" value="false"/>`
	`</Collection>`

Maximize	`<Collection name="channelsIsMaximizable">`
	`<Boolean name="channelname" value="false"/>`
	`</Collection>`

Detach	`<Collection name="channelsIsDetachable">`
	`<Boolean name="channelname" value="false"/>`
	`</Collection>`

Remove	`<Collection name="channelsIsRemovable">`
	`<Boolean name="channelname" value="false"/>`
	`</Collection>`

2. For the channel in which you want to remove a button, add the appropriate property to a Collection tag in the controlling container.

   For example, use the following XML to hide the Remove button for the Sample JSP channel in the JSP table container, MyFrontPageTabPanelContainer, whose container is JSPTabContainer.

```
<Container name="MyFrontPageFramePanelContainer"
provider="JSPTableContainerProvider">
    <Properties>
        ...
         <Collection name="channelsIsRemovable">
             <Boolean name="SampleJSP" value="false"/>
         </Collection>
    </Properties>
    ...
```

3. Load the display profile into LDAP by using the `dpadmin` command.

   See "Editing the Display Profile" on page 38.

Changing the Channel Layout for a Table Container

➤ **To Change the Channel Layout for a Table Container**

1. You can change the layout for a particular table container by modifying (or adding) the following property in the table container's display profile:

   ```
   <Integer name="layout" value="value"/>
   ```

   where value is:

   1 = Thin-wide, two columns

   2 = Wide-thin, two columns

   3 = Thin-wide-thin, three columns

2. Reload the display profile to LDAP by running the `dpadmin` with the `modify` command, loading it at the top-most node in the directory by using the `-g` option.

   For example:

   ```
   dpadmin modify -u "uid=amAdmin,ou=People,dc=sesta,dc=com" -w password
   -g dp-providers.xml
   ```

NOTE You can also use the Sun Java System Identity Server software administration console Channel and Container Management link for the Desktop service to change the channel layout.

Removing the Title Bar from a Channel

➤ **To Remove the Title Bar from a Channel**

1. Add the following to the table container display profile in which the channel is present.

```
<Collection name="channelsHasFrame">
<Boolean name="channelname" value="false"/>
</Collection>
```

2. Load the display profile into LDAP by using the `dpadmin` command.

   See "Editing the Display Profile" on page 38.

Changing the Channel Border Width

➤ **To Change the Border Width for all Channels in a Container**

1. Change directories to the appropriate table container (JSPFrameCustomTableContainerProvider, JSPTabCustomTableContainerProvider, or JSPTableContainerProvider) for which you want to change the border width.

   For example:

   ```
   cd /etc/opt/SUNWps/desktop/default/JSPTableContainerProvider
   ```

2. Edit the `leafWrapper.jsp` file.

   Change the cellpadding to not use the `borderWidth` property from theme. The line to change is:

   ```
   CELLPADDING=<dttheme:getAttribute name="borderWidth"/>
   ```

   For example, you might use `CELLPADDING=2` in place of this line.

3. Run the `touch` command on the `tablecolumn.jsp` file, which includes `leafWrapper.jsp`.

   ```
   touch tablecolumn.jsp
   ```

NOTE You can also change the `borderWidth` property for the GlobalThemes Collection. This changes the width of the channel borders for a theme. Users can then select the theme from the Themes page. (This method removes the need to change JSPs.) After making the change to the display profile, which in the sample portal is `dp-org.xml`, you need to upload it to LDAP.

Customizing the Channel Border

➤ **To Customize Channel Borders to Have Bevelled Edges, Shadows, Curved Corners, and So On**

1. Change directories to the appropriate table container (JSPFrameCustomTableContainerProvider, JSPTabCustomTableContainerProvider, or JSPTableContainerProvider) for which you want to customize the border width.

   For example:

   ```
   cd /etc/opt/SUNWps/desktop/default/JSPTableContainerProvider
   ```

2. Edit leafWrapper.jsp.

   Change the appropriate HTML. You change border width and color by modifying the following lines:

   ```
   CELLPADDING=<dttheme:getAttribute name="borderWidth"/>
   BGCOLOR="<dttheme:getAttribute name="borderColor"/>"
   ```

   To add curved borders, shadows, and so on, you use a combination of image maps and tables. While possible, such an example is beyond the scope of this document.

3. Run the `touch` command on the `tablecolumn.jsp` file, which includes `leafWrapper.jsp`.

   ```
   touch tablecolumn.jsp
   ```

Chapter 7

Customizing Instant Messaging

This chapter contains the following sections:

- Disabling the User from Editing Instant Messaging Server Information
- Automatically Closing the Instant Messaging Invite Window
- Customizing Display of Instant Messaging Contacts

Disabling the User from Editing Instant Messaging Server Information

When a site has only a single Instant Messaging server, there is no need for the end-users to be able to edit the server information in the channel edit page. The server information can be removed by editing the `IMEdit.jsp` file so that part of the page is not displayed.

Code Example 7-1 contains the Instant Messaging Server Information (snippet) in the `IMEdit.jsp` file.

Code Example 7-1 Instant Messaging Server Information in IMEdit.jsp File

```
...
<%-- ------------------- Instant Messaging Server Information --------------%>
<tr>
    <td width="100%" bgcolor="#333366" colspan="3">
        <font size="+1" face="<%=fontFace%>" color="#ffffff"><b>Instant Messaging Server
Information</b></font>
    </td>
</tr>
<tr>
    <td colspan="2" width="70%"></td>
    <td valign="top" rowspan="5">
```

Code Example 7-1 Instant Messaging Server Information in IMEdit.jsp File *(Continued)*

```
        <table border="0" cellpadding="2" cellspacing="1" width="100%" bgcolor="#eeeeee">
            <tbody>
                <tr>
                    <td valign="top" bgcolor="#dcdcdc" height="15"
                        <font face="<%=fontFace%>"><b>Why?</b></font>
                    </td>
                </tr>
                <tr>
                    <td valign="top">
                        <font face="<%=fontFace%>" size="-1">In order to access a Sun Java
System Instant Messaging server, you must enter the information needed to contact the
server.</font>
                    </td>
                </tr>
            </tbody>
        </table>
    </td>
</tr>
<tr>
    <td valign="middle" align="right"><font face="<%=fontFace%>" color="#000000"><label
for="commonname">
        <b>Server:</b>
    </label></font><br></td>
    <td valign="middle">
        <input type="text" name="server" size="25" maxlength="40"
value="<dtpc:getStringProperty key="server"/>">
    </td>
</tr>
<tr>
    <td valign="middle" align="right"><font face="<%=fontFace%>" color="#000000"><label
for="commonname">
        <b>Server Port:</b>
    </label></font><br></td>
    <td valign="middle">
        <input type="text" name="port" size="25" maxlength="40"
value="<dtpc:getStringProperty key="port"/>">
    </td>
</tr>
<tr>
    <td valign="middle" align="right"><font face="<%=fontFace%>" color="#000000"><label
for="commonname">
        <b>Multiplexor:</b>
    </label></font><br></td>
    <td valign="middle">
        <input type="text" name="mux" size="25" maxlength="40"
value="<dtpc:getStringProperty key="mux"/>">
    </td>
</tr>
<tr>
    <td valign="middle" align="right"><font face="<%=fontFace%>" color="#000000"><label
for="commonname">
        <b>Multiplexor Port:</b>
```

Code Example 7-1 Instant Messaging Server Information in IMEdit.jsp File *(Continued)*

```
    </label></font><br></td>
    <td valign="middle">
        <input type="text" name="muxport" size="25" maxlength="40"
value="<dtpc:getStringProperty key="muxport"/>">
    </td>
</tr>
<dtpc:getStringProperty id="authMethod" key="authMethod"/>
<jx:if test="$authMethod == 'ldap'">
<tr>
    <td valign="middle" align="right"><font face="<%=fontFace%>" color="#000000"><label
for="commonname">
        <b>Username:</b>
    </label></font><br></td>
    <td valign="middle">
        <input type="text" name="username" size="25" maxlength="40"
value="<dtpc:getStringProperty key="username"/>">
    </td>
</tr>
<tr>
    <td valign="middle" align="right"><font face="<%=fontFace%>" color="#000000"><label
for="commonname">
        <b>Password:</b>
    </label></font><br></td>
    <td valign="middle">
        <input type="password" name="password" size="25" maxlength="40" value="<%=
JSPProvider.getDummyPassword() %>">
    </td>
</tr>
</jx:if>
...
```

The change to this file depends on the type of authentication that is being used for the channel. With Sun Java System Identity Server software authentication (`authMethod=idsvr`), the entire Instant Messaging Server Information section can be removed. With LDAP authentication (`authMethod=ldap`), the `username` and `password` fields are still necessary. So, remove only the `server` and `port` fields and keep the rest of the Instant Messaging Server Information section.

Automatically Closing the Instant Messaging Invite Window

The `invite.jsp` file generates the content for the popup window that is created when a user is invited to a conference in an instant messaging client that is already running. It is necessary to open this window (the window is opened before the server checks if the instant messaging client is running), but the window can be closed automatically. The `invite.jsp` file has the Javascript code for doing this commented out (see Code Example 7-2). Merely uncomment this Javascript code (shown in Code Example 7-2 in bold) to cause the window to close automatically.

Code Example 7-2 Instant Messaging `invite.jsp` File

```
<%@ taglib uri="/tld/jx.tld" prefix="jx" %>
<%@ taglib uri="/tld/im.tld" prefix="im" %>
<%@ taglib uri="/tld/desktop.tld" prefix="dt" %>
<%@ taglib uri="/tld/desktopProviderContext.tld" prefix="dtpc" %>
<dt:obtainChannel channel="$JSPProvider">
<html>
    <head>
        <title>Sun Java System Instant Messenger</title>
    </head>
    <body bgcolor="#ffffff">
        <p>
            User <%= request.getParameter("username") %> is being invited to join a
conference using the IM client that you already have running.
        <p>
            <center>
                <form>
                    <input type=button value="Ok" onClick="window.close();">
                </form>
            </center>
        <%-- uncomment this script if you want the window to close automatically
             <script>window.close();</script>
        --%>
```

Customizing Display of Instant Messaging Contacts

By default, the `IMContent.jsp` file only displays presence information for on-line contacts. There is code in the JSP file for displaying presence information for all contacts in the selected contact group(s). If you wish to display all of this information, uncomment the code (shown in Code Example 7-3 on page 127) in `IMContent.jsp` file.

Code Example 7-3 Instant Messaging Offline Contact Information in `IMContent.jsp` File

```
...
<%--Offline States - Uncomment this if you want users to see offline contacts -
<jx:when test="$p=='CLOSED'">
<jx:expr value="$anchortag"/>
    <img src="im/images/Offline_13Padded.gif" alt="Offline - Click to chat" border=0
align=absmiddle></a>
    <im:getContactName/></br>
</jx:when>
<jx:when test="$p=='AWAY'">
<jx:expr value="$anchortag"/>
    <img src="im/images/Away_13Padded.gif" alt="Away - Click to chat" border=0
align=absmiddle></a>
    <im:getContactName/></br>
</jx:when>
<jx:when test="$p=='FORWARDED'">
<jx:expr value="$anchortag"/>
    <img src="im/images/Forwarded_13Padded.gif" alt="Offline - Click to chat" border=0
align=absmiddle></a>
    <im:getContactName/></br>
</jx:when>
<jx:when test="$p=='OTHER'">
<jx:expr value="$anchortag"/>
    <img src="im/images/Offline_13Padded.gif" alt="Other - Click to chat" border=0
align=absmiddle></a>
    <im:getContactName/></br>
</jx:when>
<jx:otherwise>
<jx:expr value="$anchortag"/>
    <img src="im/images/Offline_13Padded.gif" alt="<%=p%> - Click to chat" border=0
align=absmiddle></a>
    <im:getContactName/></br>
</jx:otherwise>
--  End of offline states --%>
```

Chapter 8

Customizing the Anonymous Desktop

This chapter describes customizations you can make for the anonymous Desktop. This chapter contains the following sections:

- Configuring Anonymous Authentication
- Accessing the Anonymous Desktop
- Disabling the Initial Identity Server Software Login Page and Always Use Anonymous Log In
- Modifying the Anonymous Banner and Menu Bar
- Adding the Login Channel to the Anonymous Desktop of a Newly Created Organization

When you install the sample portal, a copy of the anonymous Desktop display profile is located in the *portal-server-install-root*`/SUNWps/samples/desktop/dp-anon.xml` file, with the support files located in the `/etc/opt/SUNWps/desktop/anonymous` directory.

Configuring Anonymous Authentication

Sun Java System Portal Server software supports two methods for implementing anonymous authentication:

- Authentication-less User ID attributes - Users accessing the Desktop URL are granted access to the default Desktop.
- Anonymous user session - Users select Anonymous from the Authentication menu, log in as the user anonymous, and are granted access to the Desktop.

When you install Portal Server software, by default the installation program enables anonymous authentication to the Desktop of the default organization using the Authentication-less User ID attributes. To implement this feature, the installation program creates a user account, authlessanonymous, and sets up access for this user within the following two Desktop Services global attributes:

- Authorized Authentication-less User IDs
- Default Authentication-less User ID

This section describes how to enable and disable both types of anonymous authentication. See the Administering Users And Services chapter in the *Portal Server Administration Guide* for more information on enabling and disabling anonymous authentication.

➤ To Enable Anonymous Log In

1. Log in to the Sun Java System Identity Server software administration console as administrator.
2. Register the Anonymous service for the selected organization and create its template.
3. Add Anonymous to the Authentication menu in the Core service (for the selected organization).
4. Create the anonymous user account for the selected organization.

➤ To Disable Anonymous Log In

1. Log in to the Identity Server software administration console as administrator.
2. Unregister the Anonymous service for the selected organization.
3. Remove Anonymous from the Authentication menu in the Core service (for the selected organization).
4. Remove the anonymous user account for the selected organization.

➤ To Enable Authentication-less (authlessanonymous) Log In

1. Log in to the Identity Server software administration console as administrator.
2. Create the auhthlessanonymous account with a password of authlessanonymous for the selected organization.
3. Select the Service Configuration tab.
4. Click on the Desktop node.

   The Desktop attributes page appears in the data pane.

5. Add the following value to the Authorized Authentication-less user IDs attribute:

   ```
   uid=authlessanonymous,ou=People,dc=organization|authlessanonymous
   ```

   Substitute the appropriate organization name for organization.

6. Set the Default Authentication-less user ID attribute to the following:

   ```
   uid=authlessanonymous,ou=People,dc=organization
   ```

   Substitute the appropriate organization name for organization.

7. Log out from the Identity Server software administration console.

8. Verify that authentication-less authentication works. That is, close all current browsers and start a new browser with the following URL:

   ```
   http://hostname:port/portal/dt
   ```

   The anonymous Desktop will be displayed.

➤ To Disable Authentication-less (authlessanonymous) Log In

By default, the sample portal is registered for Authentication-less (authlessanonymous) authentication. This is different from Anonymous authentication, which the sample portal, by default, is not registered for. The Anonymous Desktop uses Portal Server software for authentication; the Authless Desktop does not pass through the authentication process at all and is handled internally in the Desktop servlet.

To disable authentication-less log in:

1. Log in to the Identity Server software administration console as administrator.

2. Select the Service Configuration tab.

3. Click on the Desktop node.

   The Desktop attributes page is displayed in the data pane.

4. Remove the value(s) from the Authorized Authentication-less user IDs attribute.

5. Remove the value from the Default Authentication-less user ID attribute so that it is blank.

6. Log out from the Identity Server software administration console.

7. Verify that you cannot reach the Anonymous Desktop. That is, close all current browsers and start a new browser with the following URL:

`http://hostname:port/portal/dt`

The Anonymous Desktop should not appear. Instead, the Login page, presenting the various authentication modules you have configured, should appear.

Accessing the Anonymous Desktop

➤ **To Access the Anonymous Desktop through the Identity Server Host Name (obj.conf File)**

To enable users to access the Anonymous Desktop without typing the fully qualified domain name, you need to modify the *Web-Container-Instance*`/config/obj.conf` file.

1. Edit the web server's `obj.conf` file.
2. After the line Object `name=default`, which is at the top, add the following lines, depending on whether you want authentication-less (authless anonymous) or anonymous access.

   For authentication-less (authlessanonymous):

```
NameTrans fn="redirect" from="/index.html"
url="http://hostname:port/portal/dt?desktop.suid=uid=authlessanonymous,ou=Peop
le,dc=organization"
```

   For anonymous:

```
NameTrans fn="redirect" from="/index.html"
url=http://hostname:port/amserver/login?org=organization&module=Anonymous
```

   For a specific organization:

```
NameTrans fn="redirect" from="/index.html"
url="http://hostname:port/amserver/login?organization
```

Make sure psservername is the fully qualified domain name of your Portal Server software host, and organization is the name of the appropriate Identity Server software organization.

NOTE For anonymous, you must also make sure that only Anonymous authentication is enabled in the Identity Server software administration console.

3. Save the file and restart Portal Server software.

   ```
   /etc/init.d/amserver start
   ```

   Users can now view the Anonymous Desktop by typing the Portal Server software host name in their browser. The fully qualified domain name is no longer required.

➤ To Access the Anonymous Desktop through the Portal Server Host Name (index.html File)

To access the Desktop login page using a URL in the following form http://*psservername*, add some JavaScript™ to the web server's index.html file.

1. Add the following Javascript to the index.html file.

```
<HTML>
    <HEAD>
        <SCRIPT>
            document.location.href="/portal/dt?desktop.suid=uid=authlessano
nymous,ou=People,dc=organization,dc=com" <-- for authless anonymous
        </SCRIPT>
    </HEAD>
</HTML>
```

This example assumes that /portal/dt is the user's redirect URL.

2. Verify that you can now access the Desktop by just typing the server name in the browser.

Disabling the Initial Identity Server Software Login Page and Always Use Anonymous Log In

➤ **To always use Anonymous Log In**

1. Log in to the Identity Server software administration console as administrator.
2. Navigate to the default organization or sub-organization.
3. Choose Services from the View menu.
4. Click the Properties icon next to Core.
5. For the Authentication Menu, make sure Anonymous is selected and deselect all other entries.
6. Click Save.
7. Create the anonymous user. With the desired organization selected, choose Users from the Show menu.
8. Click New.
9. Select the services for the anonymous user.

   Typically, you select Desktop and NetMail.
10. Type in the Create User screen with the following information.
    - UserID - anonymous
    - First Name - (blank)
    - Last Name - anonymous
    - Full Name - anonymous
    - Password - anonymous
    - Password (confirm) - anonymous

11. Click the create button to create the user.

    When users type the URL to access the portal server in a browser, the anonymous Desktop comes up, bypassing the Identity Server software login page. This Desktop will have the login channel, where users can log in if desired.

Modifying the Anonymous Banner and Menu Bar

To change the banner for the Anonymous Desktop, you need to modify the `/etc/opt/SUNWps/desktop/anonymous/banner.template` file. To modify the menu bar, you need to modify the `/etc/opt/SUNWps/desktop/anonymous/menubar.template` file.

➤ **To Change the Banner for the Anonymous Desktop**

1. Edit the `banner.template` file.
2. Make your modifications.

   For example, you could change the following line to a background color or image of your choice:

```
<td bgcolor="#333366"><img src="[surl:/images/productName.gif]" width="274
" height="38" alt="Sun Java System Portal Server"></td>
```

   Replace `[surl:/images/productName.jpg]` with a reference to an alternate image. For example, if you use *identity-server-install-root*`/SUNWam/public_html/images/newimage.gif`, then use `/images/newimage.gif` as your replacement text. The `[surl:]` tag references image files from the Portal Server software web application archive. Your own custom images need to be placed elsewhere, so the `[surl:]` tag is not used.

3. Place your file in the appropriate directory.

   You can place your custom image files under the web server document root or you can deploy them in a custom web application archive. See the web server documentation for information on how to deploy a web application archive.

   You could also make a new `banner.template` file to replace the default one.

4. Modify the `menubar.template` file. You could also make a new `menubar.template` file to replace the default one.

Adding the Login Channel to the Anonymous Desktop of a Newly Created Organization

The default organization in the sample portal is configured with the login channel on the Anonymous Desktop. This enables new users who do not already have a membership user account to sign up for a membership user account. The login channel is also the only way a user can log in when anonymous is the sole authentication module selected.

As you add new organizations, you might want to set up the login channel on the Anonymous Desktop of the new organization.

➤ **To Add the Login Channel to the Anonymous Desktop of a Newly Created Organization**

1. Use the Identity Server software administration console to create the new organization (this example uses `company22.com` as the initial organization and `sesta.com` as the new one), register the appropriate services (Core, Membership, LDAP, Desktop, NetMail, User, and so on), create the service templates, and assign policies to execute Desktop and NetMail.

   See the *Portal Server Administration Guide* for details.

TIP Make sure that the Desktop policy contains the rule to execute the Desktop, and that in the Core service you add Membership to the Authentication Menu.

2. In the Identity Server software administration console, choose Organizations from the View menu in the Identity Management tab.
3. Navigate to the newly created organization.
4. Create a user account for the authless session.
   a. Choose Users from the View menu then click New.
   b. Select Desktop and NetMail for services then click Next.

      The Create User page opens in the data pane.

c. Type values for the required fields. This example uses authlessanonymous as the user ID and authlessanonymous as the password.

d. When done click Create.

The authlessanonymous user ID appears in the list of users.

5. Add the authlessanonymous user ID to the list of authorized users for the global Desktop service.

a. Choose Service Configuration tab.

b. Click the Properties arrow icon next to Portal Desktop.

The Desktop attributes page opens in the data pane.

c. Type the following for the Authorized Authentication-less User IDs attribute:

`uid=authlessanonymous,ou=People,dc=sesta,dc=com|authlessanonymous`

d. Click Add.

e. Click Save.

6. Load the display profile for the organization by using the `dpadmin` command.

This example uses the `dp-org.xml` file as the display profile for the new organization, `sesta.com`.

```
/opt/SUNWps/bin/dpadmin add -u "uid=amAdmin,ou=People,dc=sesta,dc=com" -w
password -d "dc=sesta,dc=com" /opt/SUNWps/samples/desktop/dp-org.xml
```

7. Copy the sample anonymous display profile, `dp-anon.xml`, to a new file.

For example,

```
<Reference value="Login"/>
...
<String name="Login" value="1"/>
...
<String value="Login"/>
...
<Boolean name="Login" value="false"/>
...
<Channel name="Login" provider="LoginProvider">
```

You do not want to modify the sample `dp-anon.xml` file, as you may want to have it as a backup in case need it for reloading that for your default organization.

8. Edit the `dp-anon-sesta.xml` display profile file to change every instance of the Login channel to LoginSesta.

The lines of the `dp-anon-sesta.xml` display profile to be changed look like this:

```
<Reference value="Login"/>
...
<String name="Login" value="1"/>
...
<String value="Login"/>
...
<Boolean name="Login" value="false"/>
...
<Channel name="Login" provider="LoginProvider">
```

NOTE Do not change LoginProvider to LoginSestaProvider. The provider name must stay the same.

9. Load the anonymous display profile for the authless user ID by using the `dpadmin` command.

```
/opt/SUNWps/bin/dpadmin add -u "uid=amAdmin,ou=People,dc=sesta,dc=com" -w
password -d "uid=authlessanonymous,ou=People,dc=sesta,dc=com"
dp-anon-sesta.xml
```

10. Create the channel templates for the new login channel.

    a. Change directories to the `/etc/opt/SUNWps/desktop/`*desktoptype* directory.

       ```
       cd /etc/opt/SUNWps/desktop/desktoptype
       ```

    b. Copy the `Login` directory contents to a new directory, `LoginSesta`.

       ```
       cp -r Login LoginSesta
       ```

    c. Change directories to the `LoginSesta` directory.

       ```
       cd /etc/opt/SUNWps/desktop/desktoptype/LoginSesta
       ```

    d. Change the Form action value from `/amserver/login` to `/amserver/login?org=sesta.com` in all the display template files (`display.html`, `display_AuthLDAP.html`, and `display_AuthUnix.html`).

    e. Change the "Sign me up" URL from `<A HREF="/amserver/login?module...>` to `<A HREF="amserver/login?org=sesta.com&module...>` in all the display template files.

11. Set the Desktop type for the authless user.

    a. In the Identity Server software administration console, select the newly created organization.

    b. Choose Users from the View menu.

    c. Click the Properties arrow icon next to the authlessanonymous user ID.

    d. Select Edit at the end of the Desktop line in the data pane.

    e. In the popup window, type anonymous in the Desktop Type field and select Customize in the drop-down menu next to the text field.

    f. Click Save.

    g. Access the authless anonymous Desktop for the new organization by typing the following URL:

    ```
    http://psserver:port/portal/dt?desktop.suid=uid=authlessanonymous,ou=People,dc=sesta,dc=com
    ```

Modifying the Default Desktop (Container) for authlessanonymous User

To change the default channel name for authlessanonymous user from JSPTabContainer to another container, for example, JSPTableContainer, perform the following:

➤ **To Change the Default Channel Name for Authlessanonymous User**

1. Log in to the administration console and select Users View for your organization.
2. Select authlessanonymous and Portal Desktop from the View pull-down menu for authlessanonymous users.
3. Select the Edit link.
4. Change the Default Channel Name and select Customize from the pull-down menu.
5. Select Save.
6. Validate the change to the Desktop.

Chapter 9

Customizing Authentication

Sun Java System Portal Server supports a number of authentication schemes, including LDAP, anonymous, membership, UNIX, and more. See the *Portal Server Administration Guide* for information on configuring authentication, as well as users and roles.

This chapter contains instructions for:

- Using UNIX Authentication with LoginProvider
- Configuring LDAP Authentication for UserInfoProvider

Using UNIX Authentication with LoginProvider

➤ **To Use UNIX Authentication with LoginProvider**

1. Change directories to the default/Login directory.

   For example:

   ```
   cd /etc/opt/SUNWps/desktop/default/Login
   ```

2. Copy the `display_UnixAuth.html` file to `display.html`.

   For example,

   ```
   cp display_AuthUnix.html display.html
   ```

3. Register and enable UNIX authentication service for the organization.

   See the *Portal Server Administration Guide* for details.

4. Add Unix to the Non Interactive Modules in the Core service.

NOTE To use LDAP authentication, the authentication module is already enabled for the default organization. You only need to copy `display_AuthLDAP.html` to `display.html`.

Configuring LDAP Authentication for UserInfoProvider

Out of the box, the UserInfo channel allows the user to edit and maintain their Membership password (change their own password). To change the user's authentication module to only LDAP, the administrator has to customize the UserInfoProvider to acknowledge LDAP authenticated users.

➤ **To Enable End User Password Maintenance for LDAP Authentication**

1. Create an LDAP `passwordHandler` template. The template name format is `passwordHandler-`*authType*`.template`.

   You can copy an existing template in the Userinfo template directory. For example,

   ```
   cd /etc/opt/SUNWps/desktop/default/UserInfo/html
   ```

   ```
   cp passwordHandler-Membership.template passwordHandler-LDAP.template
   ```

2. Optionally, modify the descriptive text within `passwordHandler-`*authType*`.template`.

   For example, in the `passwordHandler-LDAP.template` file, change the `Membership` to `LDAP`.

3. Add the authentication module name to the channel's `authTypes` display profile Collection.

   Use the `dpadmin` utility to add the entry to the UserInfoProvider `<Provider>` element. For example:

   a. Type ***portal-server-install-root*/SUNWps/bin/dpadmin list -u *uid* -w *password* -g > provider.xml**.

      Make a backup copy of provider.xml. For example, type **`cp provider.xml provider-original.xml`**.

   b. Add the entry `LDAP` to the `authTypes` collection for the UserInfoProvider in the `provider.xml` file as shown (in bold) below:

```
<Collection name="authTypes" advanced="true">
    <String value="Membership"/>
    <String value="LDAP"/>
</Collection>
```

Here, based on the user's SSOToken authentication type, the appropriate authType will be used.

c. Import the modified display profile document. For example, type

***portal-server-install-root*/SUNWps/bin/dpadmin modify -u *uid* -w *password* -g provider.xml**

4. Restart the web container.
5. Access the portal desktop as an LDAP authenticated user and edit the user info channel.

   Verify that the password field is displayed.
6. Modify the user's password and select finished
7. Logout and login to the Desktop with the new credentials.

Chapter 10

Modifying the Desktop Layout

This chapter describes how to modify the channel arrangement in the Desktop. It contains the following sections:

- Deriving More Desktop Layouts
- Changing Content Layout to Support Categorizing the Available and Selected Lists
- Adding New Layouts
- Changing the Desktop Column Layout

Deriving More Desktop Layouts

The Desktop Layout page provides a way for users to set the arrangement of the channels by moving them up or down, and right or left. The Desktop Layout page also provides users with the option to set column layout, where they can arrange columns by channel width.

Channel widths are defined as thin, wide, full_top, and full_bottom. A thin channel takes less Desktop area than a wide channel. A full_top channel spans the entire Desktop width above all the other channels. A full_bottom channel spans the entire Desktop width below all other channels. The available layouts, which use different combinations of channel widths, are:

- thin-wide (two columns)
- wide-thin (two columns)
- thin-wide-thin (three columns)
- full_top (one column spans the width of the Desktop at the top of the page)

- full_bottom (one column spans the width of the Desktop at the bottom of the page)

You can derive more Desktop layouts from the existing layouts by modifying display profile properties and the JSPs for the table container, when one of the contained channel's width is specified as either full_top or full_bottom.

For example, you could come up with the following:

- full_top-thin-wide / full_top-wide-thin / full_top-thin-wide-thin
- thin-wide-full_bottom
- fulltop-thin-wide-full_bottom (This layout is a combination of the first two with full_top at the top and full_bottom at the bottom.)

To do so involves modifying the appropriate display profile. That is, to derive more desktop layouts, use the appropriate display profile for the desired layout. After making a change to the display profile, load the display profile into LDAP by using the `dpadmin` command.

To use a full_top-thin-wide / full_top-wide-thin / full_top-thin-wide-thin layout, modify a channel's width in the display profile as follows:

```
<Channel name="Search" provider="SearchProvider">
    <Properties>
        <String name="title" value="Search"/>
        <String name="description" value="This is a search provider example"
/>
        <String name="searchServer" value=""/>
         <String name="width" value="full_top"/>
    </Properties>
</Channel>
```

To use a thin-wide-full_bottom layout, modify a channel's width in the display profile as follows:

```
<Channel name="Search" provider="SearchProvider">
    <Properties>
        <String name="title" value="Search"/>
        <String name="description" value="This is a search provider example"
/>
        <String name="searchServer" value=""/>
```

```
        <String  name="width" value="full_bottom"/>
    </Properties>
</Channel>
```

To use a full_top-thin-wide-full_bottom layout, modify the width of one of the channels as full_bottom and one of the other channels as full_top in the display profile.

Changing Content Layout to Support Categorizing the Available and Selected Lists

For sites with a great number of channels, and a need to categorize and sub-categorize these into a hierarchical kind of structure—to make channel selection easier to navigate, and channels easier to find—customization options include:

- Customizing Existing JSPs
- Writing a New Content Channel

Customizing Existing JSPs

You can customize `contentedit.jsp`, which generates the Content page, and `contentdoedit.jsp`, which processes the page. The Content page uses categories to group channels into different sections. If you want this page to look different, for example, to use a pull-down menu instead of showing all the channels for one category, to save on page space, you can customize the two JSPs mentioned. (The main reason for a JSP-based content and layout channels is to support ease of customization.)

Writing a New Content Channel

You can write a new content channel by extending JSPSingleContainerProvider to support your site's needs. You would then only need to modify the Content link in `table.jsp` under JSPTableContainerProvider to point to this new content channel. You can also create your own JSP to do whatever specific implementation you need, and change the link from `table.jsp` to `new.jsp`.

Adding New Layouts

The following example is intended to show some of the customization possibilities for the Desktop. Details are not provided. See the *Portal Server Developer's Guide* for more information on how to add new layouts.

In this scenario, the Desktop has three rows. The first row contains one full-width channel; the second row contains 2 channels, a thin plus a thick channel; and the third row contains 3 thin channels of equal width.

To create such a Desktop requires a custom container channel, created from the JSPs for the table container (JSPTableContainer).

To enable the Layout link to work with this container, you need a new layout channel with a customized JSP for editing and processing the Edit page. You build this by starting from the `layoutedit.jsp` and `layoutdoedit.jsp` files.

You also need to create a new custom table container provider by extending JSPTableContainerProvider.

Changing the Desktop Column Layout

➤ **To Change the Desktop Column Layout from the Command Line**

1. Modify the `width` property to change the layout of the channel column. Specify one of the following:
   - thin - The channel appears in one of the thin columns.
   - thick - The channel appears in one of the wide columns.
   - full_top - The channel appears at the top spanning the entire horizontal space.
   - full_bottom - The channel appears at the bottom spanning the entire horizontal space.

   For example, the following specifies that a channel appears at the top spanning the entire horizontal space:

   ```
   <String name="width" value="full_top"/>
   ```

2. Load the display profile into LDAP by using the `dpadmin` command

   See "Editing the Display Profile" on page 38.

➤ To Change the Desktop Column Layout from the Administration Console

1. Log in to the Identity Server software administration console as administrator.
2. Navigate to User Management by choosing View User Management.
3. Select the appropriate organization or suborganization.
4. Choose Services from the Show menu.
5. Select the properties arrow next to Desktop in the navigation pane.

   The Desktop attributes appear in the data pane.
6. Click Channel and Container Management.

   The Channels page appears, with the container path set at the root. The defined channels appear in a list.
7. Select the Edit link beside the channel to be modified.

   The Properties page appears.
8. Modify the width property.

   The possible values are thin, thick, full_top, and full_bottom.
9. Click Save.

➤ To Modify Column Widths Directly (Using Scriptlets)

To modify the column widths directly, you can add the following code to the `JSPTableContainerProvider/toptable.jsp` file.

```
<%@ page import="com.sun.portal.search.providers.util.Layout"%>
<%
    JSPTableContainerProvider tcp =
(JSPTableContainerProvider)pageContext.getAttribute("JSPProvider");
    int layout = tcp.getLayout();
    int centerWidth = -1;
    int rightWidth = -1;
    int leftWidth = -1;
    switch (layout) {
        case Layout.LAYOUT_THIN_THICK:
            leftWidth = 40;
            rightWidth = 60;
            break;
        case Layout.LAYOUT_THICK_THIN:
            rightWidth = 40;
            leftWidth = 60;
            break;
        case Layout.LAYOUT_THIN_THICK_THIN:
            rightWidth = 30;
```

```
            centerWidth= 40;
            leftWidth = 30;
            break;
        default:
            rightWidth = 40;
            leftWidth = 60;
            break;
    }
%>
and then replace the widths
<!-- BEGIN LEFT CHANNELS -->
<TD WIDTH="<%=leftWidth%>" VALIGN=TOP>
<!-- BEGIN CENTER CHANNELS -->
<TD WIDTH="<%=centerWidth%"  VALIGN=TOP>
<!-- BEGIN RIGHT CHANNELS -->
<TD WIDTH="<%=rightWidth%>"  VALIGN=TOP>
```

Chapter 11

Branding the Desktop

This chapter describes how to brand the Desktop with your site's logo and name. It contains the following sections:

- Changing the HTML Title (Title That Appears in the Browser)
- Changing the Logo (Image) in the Banner Header
- Changing the Header and Footer of the Theme, Content, and Layout Pages

Changing the HTML Title (Title That Appears in the Browser)

The title is in the `productName` property in the display profile definition for all the providers and channels. Edit this property to change the HTML title.

Changing the Logo (Image) in the Banner Header

The logo image is defined in the themes in the display profile. The related theme properties are:

`brandImage`	The brand image on the left of the header.
`brandImage2`	The brand image in the center of the header; if there's no need to have the second image, then use `spacer.gif`.
`brandImageBgColor`	The background color for the left image file.
`brandImage2BgColor`	The background color for the center image file.

`brandImageWidth`	The width of the left image file.
`previewImage`	The image that is displayed in the Theme / preset Themes page.

➤ **To Change the Logo (image) in the Banner**

1. Log in to the Sun Java System Identity Server administration console.
2. Select Services from your Organization View pull-down menu and select Portal Desktop.
3. Select Edit XML to directly edit the display profile XML fragment. Or,
   a. Select Container and Channel Management Link and Edit Properties for Display Profile.
   b. Select Edit Collection for Global Themes and the theme you wish to modify.
4. Modify the relevant theme properties.
5. Copy the new image into *portal-server-install-root*`/SUNWps/web-src/images` directory.
6. Run *portal-server-install-root*`/SUNWps/bin/deploy redeploy -deploy_admin_password` *password* command to deploy the new image.

   The images will be deployed to the *web_container_install_root*`/portal_web_application_install_root/images` directory.
7. Reload the Desktop to verify the change.

Changing the Header and Footer of the Theme, Content, and Layout Pages

Depending on the Desktop, the particular header and footer files for in the Theme, Content, and Layout pages are determined by the container that contains the Theme, Content, and Layout pages.

For example, when you access the Content page for JSPTabContainer, JSPContentContainer is the container that is used to include the header and footer files that the tab container is using. The `contentedit.jsp` file, located in the JSPContentContainer directory, uses logic, based on the container, to access the appropriate header and footer files.

To brand the header and footer of the Theme, Content, and Layout pages:

Use Table 11-1 to determine the appropriate header and footer JSP files to edit. This two column table lists the Desktop containers in the first column and the corresponding header and footer files in the second column.

Table 11-1 Header and Footer Files for Theme, Content, and Layout Pages

Desktop Container	Header and Footer Files for Theme, Content, and Layout Pages
FrameTabContainer	`framePreferenceHeader.jsp` and `framePreferenceMenubar.jsp`
JSPSingleContainer	`singlePreferenceHeader.jsp` and `singlePreferenceMenubar.jsp`
JSPTabContainer	`tabPreferenceHeader.jsp` and `tabPreferenceMenubar.jsp`
JSPTableContainer	`tablePreferenceHeader.jsp` and `tablePreferenceMenubar.jsp`

Containers not listed in Table 11-1 use `defaultHeader.jsp` and `defaultMenubar.jsp` files. These two files are actually the same as `singlePreferenceHeader.jsp` and `singlePreferenceMenubar.jsp` files. If you want a default look and feel for the container's header and menubar, customize these two JSPs. Currently, the sample portal does not use `defaultHeader.jsp` and `defaultMenubar.jsp` files.

➤ To Change the Header and Footer of the Theme, Content, and Layout Pages

1. Change to the appropriate directory.

   That is, change to `/etc/opt/SUNWps/desktop/sampleportal` (if sampleportal is installed) or change to the specific desktop type subdirectory associated with the target user or organization.

2. Edit the JSP files.

   For example, change the HTML title and logo in the header file, and change the product name in the footer.

3. Run the `touch` command.

   For example, type **`touch *.jsp`**.

4. Reload the Desktop to verify the change.

Chapter 12

Changing Desktop Colors

This chapter describes how to change the color for various Desktop components, such as header, footer, font color in the header and footer, and so on. This chapter contains the following:

- Changing Desktop Colors
- Changing the Default Color Scheme for an Organization

Changing Desktop Colors

Most of these colors are part of the global theme attributes. See Chapter 13, “Customizing the Global Themes” for more information.

➤ **To Change the Desktop Colors**

1. Use Table 12-1 on page 156 to determine what you want to change and what file you need to change. This two column table lists the Desktop component to customize in the first column, the files to edit in the second column, and the corresponding theme attribute in the third column.

Table 12-1 Files to Customize to Change Desktop Colors

Desktop Component to Customize	File to Edit in /etc/opt/SUNWps/desktop/default/	Theme Attribute
Header background color	JSP-based containers: • ./JSPSingleContainerProvider/header.jsp • ./JSPTabContainer/header.jsp • ./JSPTableContainerProvider/header.jsp • ./PredefinedFrontPageFramePanelContainerProvider/header.jsp • ./PredefinedFrontPageTabPanelContainerProvider/header.jsp • ./PredefinedSamplesTabPanelContainerProvider/header.jsp Frame-based containers: • ./FrameTabContainer/banner.jsp • ./PredefinedSamplesFramePanelContainerProvider/header.jsp	brandBgColor
Footer background color	JSP-based containers: • ./JSPSingleContainerProvider/menubar.jsp • ./JSPTabContainer/menubar.jsp • ./JSPTableContainerProvider/menubar.jsp • ./PredefinedFrontPageFramePanelContainerProvider/menubar.jsp • ./PredefinedFrontPageTabPanelContainerProvider/menubar.jsp • ./PredefinedSamplesTabPanelContainerProvider/menubar.jsp Frame-based containers: • ./FrameTabContainer/menubar.jsp • ./PredefinedSamplesFramePanelContainerProvider/menubar.jsp	brandBgColor
Font color in the header and footer	The related JSPs are the `header.jsp` and `menubar.jsp` that listed in header background color and footer background color.	headerFontColor

Table 12-1 Files to Customize to Change Desktop Colors *(Continued)*

Desktop Component to Customize	File to Edit in /etc/opt/SUNWps/desktop/default/	Theme Attribute
Selected tab color	JSP-based containers: ./JSPTabContainer/selectedTab.jsp Frame-based containers: ./FrameTabContainer/selectedTab.jsp	titleBarColor
Content Page color	./JSPContentContainer/contentLayoutBar.jsp ./JSPEditContainer/contentLayoutBar.jsp ./JSPLayoutContainer/contentLayoutBar.jsp ./TabJSPEditContainer/contentLayoutBar.jsp	(none)
Layout Page color	./JSPLayoutContainer/layoutedit.jsp	(none)
Desktop body	./JSPTableContainerProvider/tabtable.jsp	tableBgColor

2. Edit the appropriate file.

   In almost all case, make modifications to the `bgcolor=value` statement to change the color. In the case of the font color in the header and footer, change the color inside of the `FONT` tag for the specific link.

3. In the directory where you make the change, run the following command:

   ```
   touch *.jsp
   ```

   (Or, if you know the parent JSP file, just run the `touch` command on that file.)

4. Reload the Desktop.

Changing the Default Color Scheme for an Organization

There are two ways in which to provide a new color scheme and layout for an organization:

- Define a new set of templates - You can define a new set of templates in `/etc/opt/SUNWps/desktop/new/` and make this directory (new) the Desktop Type attribute for that organization. See the *Portal Server Administration Guide* for more details.

- Define a new theme - In the display profile, you can define your own theme in the GlobalThemes collection. See “Adding and Customizing Global Themes” on page 170 for more information.

Chapter 13

Customizing the Global Themes

This chapter contains the following:

- What is a Theme?
- Customization Overview
- GlobalThemes Display Profile Definition
- Theme Properties
- Glossary of Terms
- Adding and Customizing Global Themes

What is a Theme?

The Desktop theme provides the capability of creating a customizable user interface that allows the end users to select different look and feel for their Desktop.

The definition of a theme in Sun Java System Portal Server software Desktop is a collection of user interface attributes that are used in the markup output from the Desktop. The attributes can be colors, fonts, and images. Out of the box, there are eight themes that come with the sample portal and each theme contains thirty eight (38) attributes.

Customization Overview

There are two levels of customization for the themes:

The number of themes and theme attributes are configurable by the administrators. Theme and theme attributes are display profile properties; so they can be modified through the Sun Java System Identity Server software administration console, or they can be edited in the display profile directly. The theme properties are defined as global properties in the organization level in the sample portal. So, when a new theme is created, all users in the organization will see it.

The end user can select one of the preset themes that are defined by the administrator, or customize some theme attribute values inside of the theme page in the Desktop. When the theme changes, it applies to all the containers in the Desktop, and also, the changed property will be stored in the user level display profile.

There are tag library functions defined to allow JSPs to retrieve the theme related values from the display profile (see Appendix D, "JavaServer Pages Tag Library Reference" for more information.) Behind the scene, the tag library functions use the Theme Java class to get the theme properties. For more information on the Theme Java class, please see the Java docs for `com.sun.portal.providers.context.Theme`.

GlobalThemes Display Profile Definition

The following display profile XML fragment shows the eight themes defined in *portal-server-install-root*`/SUNWps/samples/desktop/dp-org.xml` file.

Code Example 13-1 Display Profile Definition for GlobalThemes

```
<Collection name="GlobalThemes" propagate="false">
    <Collection name="SunTheme">
        <String name="description" value="Sun"/>
        <String name="bgColor" value="white"/>
        <String name="titleBarColor" value="#999999"/>
        <String name="fontColor" value="black"/>
        <String name="borderColor" value="#999999"/>
        <String name="borderWidth" value="1"/>
        <String name="fontFace" value="Sans-serif"/>
        <String name="fontSize" value="3"/>
        <String name="activeBulletImage" value="branded_bullet_on.gif"/>
        <String name="inactiveBulletImage" value="branded_bullet_off.gif"/>
        <String name="brandImage" value="logo_sun.gif"/>
        <String name="brandImage2" value="sunONE_productlogo.gif"/>
        <String name="brandImageBgColor" value="#594FBF"/>
        <String name="brandImage2BgColor" value="#FBE249"/>
        <String name="brandBgColor" value="#ffffff"/>
        <String name="brandImageWidth" value="110"/>
        <String name="headerBgColor" value="#e5e5e5"/>
```

Code Example 13-1 Display Profile Definition for GlobalThemes *(Continued)*

```
        <String name="headerFontColor" value="#594FBF"/>
        <String name="headerText" value="sansSerif11Font"/>
        <String name="tabNotchImage" value="tabNotch.gif"/>
        <String name="tabColor" value="#cccccc"/>
        <String name="tabFontColor" value="#000000"/>
        <String name="contentLayoutLinkColor" value="#FFFFFF"/>
        <String name="contentLayoutText" value="sansSerif10Font"/>
        <String name="linkSeparatorColor" value="#CCCCCC"/>
        <String name="tableBgColor" value="#FFFFFF"/>
        <String name="titleFontColor" value="#ffffff"/>
        <String name="titleText" value="sansSerif10Font"/>
        <String name="channelHighlightColor" value="#ffffff"/>
        <String name="channelLinkColor" value="#3a2eb5"/>
        <String name="previewImage" value="Sun_preview.gif"/>
        <String name="helpImage" value="b_help.gif"/>
        <String name="removeImage" value="b_remove.gif"/>
        <String name="minimizeImage" value="b_minimize.gif"/>
        <String name="maximizeImage" value="b_maximize.gif"/>
        <String name="normalizeImage" value="b_normal.gif"/>
        <String name="attachImage" value="b_attach.gif"/>
        <String name="detachImage" value="b_new_window.gif"/>
        <String name="editImage" value="b_edit.gif"/>
    </Collection>
    <Collection name="FieryTheme">
        <String name="description" value="Fiery"/>
        <String name="bgColor" value="white"/>
        <String name="titleBarColor" value="#990000"/>
        <String name="fontColor" value="black"/>
        <String name="borderColor" value="#990000"/>
        <String name="borderWidth" value="1"/>
        <String name="fontFace" value="Sans-serif"/>
        <String name="fontSize" value="3"/>
        <String name="activeBulletImage" value="fiery_bullet_on.gif"/>
        <String name="inactiveBulletImage" value="fiery_bullet_off.gif"/>
        <String name="brandImage" value="Fiery_000000.gif"/>
        <String name="brandImage2" value="spacer.gif"/>
        <String name="brandImageBgColor" value="#000000"/>
        <String name="brandImage2BgColor" value="#000000"/>
        <String name="brandBgColor" value="#000000"/>
        <String name="headerBgColor" value="#000000"/>
        <String name="brandImageWidth" value="243"/>
        <String name="headerFontColor" value="#FFFFFF"/>
        <String name="headerText" value="sansSerif11Font"/>
        <String name="tabNotchImage" value="black_tabend.gif"/>
        <String name="tabColor" value="#FF3300"/>
        <String name="tabFontColor" value="#000000"/>
        <String name="contentLayoutLinkColor" value="#ff3300"/>
        <String name="contentLayoutText" value="sansSerif10Font"/>
        <String name="linkSeparatorColor" value="#000000"/>
        <String name="tableBgColor" value="#999999"/>
        <String name="titleFontColor" value="#ffffff"/>
        <String name="titleText" value="sansSerif10Font"/>
        <String name="channelHighlightColor" value="#ffffff"/>
        <String name="channelLinkColor" value="#3a2eb5"/>
```

Code Example 13-1 Display Profile Definition for GlobalThemes *(Continued)*

```
        <String name="previewImage" value="Fiery_preview.gif"/>
        <String name="helpImage" value="b_help.gif"/>
        <String name="removeImage" value="b_remove.gif"/>
        <String name="minimizeImage" value="b_minimize.gif"/>
        <String name="maximizeImage" value="b_maximize.gif"/>
        <String name="normalizeImage" value="b_normal.gif"/>
        <String name="attachImage" value="b_attach.gif"/>
        <String name="detachImage" value="b_new_window.gif"/>
        <String name="editImage" value="b_edit.gif"/>
    </Collection>
    <Collection name="BreezyTheme">
        <String name="description" value="Breezy"/>
        <String name="bgColor" value="white"/>
        <String name="titleBarColor" value="#336699"/>
        <String name="fontColor" value="black"/>
        <String name="borderColor" value="#336699"/>
        <String name="borderWidth" value="1"/>
        <String name="fontFace" value="Sans-serif"/>
        <String name="fontSize" value="3"/>
        <String name="activeBulletImage" value="breezy_bullet_on.gif"/>
        <String name="inactiveBulletImage" value="breezy_bullet_off.gif"/>
        <String name="brandImage" value="Breezy_FFFFFF.gif"/>
        <String name="brandImage2" value="spacer.gif"/>
        <String name="brandImageBgColor" value="#FFFFFF"/>
        <String name="brandImage2BgColor" value="#FFFFFF"/>
        <String name="brandBgColor" value="#FFFFFF"/>
        <String name="headerBgColor" value="#FFFFFF"/>
        <String name="brandImageWidth" value="243"/>
        <String name="headerFontColor" value="#000000"/>
        <String name="headerText" value="sansSerif11Font"/>
        <String name="tabNotchImage" value="tabNotch.gif"/>
        <String name="tabColor" value="#CCCC66"/>
        <String name="tabFontColor" value="#000000"/>
        <String name="contentLayoutLinkColor" value="#FFFFFF"/>
        <String name="contentLayoutText" value="sansSerif10Font"/>
        <String name="linkSeparatorColor" value="#ED471E"/>
        <String name="tableBgColor" value="#CCCCFF"/>
        <String name="titleFontColor" value="#ffffff"/>
        <String name="titleText" value="sansSerif10Font"/>
        <String name="channelHighlightColor" value="#ffffff"/>
        <String name="channelLinkColor" value="#3a2eb5"/>
        <String name="previewImage" value="Breezy_preview.gif"/>
        <String name="helpImage" value="b_help.gif"/>
        <String name="removeImage" value="b_remove.gif"/>
        <String name="minimizeImage" value="b_minimize.gif"/>
        <String name="maximizeImage" value="b_maximize.gif"/>
        <String name="normalizeImage" value="b_normal.gif"/>
        <String name="attachImage" value="b_attach.gif"/>
        <String name="detachImage" value="b_new_window.gif"/>
        <String name="editImage" value="b_edit.gif"/>
    </Collection>
    <Collection name="SereneTheme">
        <String name="description" value="Serene"/>
        <String name="bgColor" value="white"/>
```

Code Example 13-1 Display Profile Definition for GlobalThemes *(Continued)*

```
        <String name="titleBarColor" value="#EAE7E7"/>
        <String name="fontColor" value="black"/>
        <String name="borderColor" value="#EAE7E7"/>
        <String name="borderWidth" value="0"/>
        <String name="fontFace" value="Sans-serif"/>
        <String name="fontSize" value="3"/>
        <String name="activeBulletImage" value="serene_bullet_on.gif"/>
        <String name="inactiveBulletImage" value="serene_bullet_off.gif"/>
        <String name="brandImage" value="Serene_330099.gif"/>
        <String name="brandImage2" value="spacer.gif"/>
        <String name="brandImageBgColor" value="#330099"/>
        <String name="brandImage2BgColor" value="#330099"/>
        <String name="brandBgColor" value="#330099"/>
        <String name="headerBgColor" value="#330099"/>
        <String name="brandImageWidth" value="243"/>
        <String name="headerFontColor" value="#FFFFFF"/>
        <String name="headerText" value="sansSerif11Font"/>
        <String name="tabNotchImage" value="serene_tabend.gif"/>
        <String name="tabColor" value="#9999FF"/>
        <String name="tabFontColor" value="#000000"/>
        <String name="contentLayoutLinkColor" value="#330099"/>
        <String name="contentLayoutText" value="sansSerif10Font"/>
        <String name="linkSeparatorColor" value="#9999FF"/>
        <String name="tableBgColor" value="#FFFFFF"/>
        <String name="titleFontColor" value="#330099"/>
        <String name="titleText" value="sansSerif10Font"/>
        <String name="channelHighlightColor" value="#ffffcc"/>
        <String name="channelLinkColor" value="#3a2eb5"/>
        <String name="previewImage" value="Serene_preview.gif"/>
        <String name="helpImage" value="b_help.gif"/>
        <String name="removeImage" value="b_remove.gif"/>
        <String name="minimizeImage" value="b_minimize.gif"/>
        <String name="maximizeImage" value="b_maximize.gif"/>
        <String name="normalizeImage" value="b_normal.gif"/>
        <String name="attachImage" value="b_attach.gif"/>
        <String name="detachImage" value="b_new_window.gif"/>
        <String name="editImage" value="b_edit.gif"/>
    </Collection>
    <Collection name="SportingTheme">
        <String name="description" value="Sporting"/>
        <String name="bgColor" value="white"/>
        <String name="titleBarColor" value="#009999"/>
        <String name="fontColor" value="black"/>
        <String name="borderColor" value="#009999"/>
        <String name="borderWidth" value="1"/>
        <String name="fontFace" value="Sans-serif"/>
        <String name="fontSize" value="3"/>
        <String name="activeBulletImage" value="sporting_bullet_on.gif"/>
        <String name="inactiveBulletImage" value="sporting_bullet_off.gif"/>
        <String name="brandImage" value="Sporting_000000.gif"/>
        <String name="brandImage2" value="spacer.gif"/>
        <String name="brandImageBgColor" value="#000000"/>
        <String name="brandImage2BgColor" value="#000000"/>
        <String name="brandBgColor" value="#000000"/>
```

Code Example 13-1 Display Profile Definition for GlobalThemes *(Continued)*

```
            <String name="headerBgColor" value="#000000"/>
        <String name="brandImageWidth" value="243"/>
        <String name="headerFontColor" value="#FFFFFF"/>
        <String name="headerText" value="sansSerif11Font"/>
        <String name="tabNotchImage" value="black_tabend.gif"/>
        <String name="tabColor" value="#CCCCCC"/>
        <String name="tabFontColor" value="#000000"/>
        <String name="contentLayoutLinkColor" value="#000000"/>
        <String name="contentLayoutText" value="sansSerif10Font"/>
        <String name="linkSeparatorColor" value="#CCCCCC"/>
        <String name="tableBgColor" value="#FFFFFF"/>
        <String name="titleFontColor" value="#000000"/>
        <String name="titleText" value="sansSerif10Font"/>
        <String name="channelHighlightColor" value="#ffffff"/>
        <String name="channelLinkColor" value="#3a2eb5"/>
        <String name="previewImage" value="Sporting_preview.gif"/>
        <String name="helpImage" value="b_help.gif"/>
        <String name="removeImage" value="b_remove.gif"/>
        <String name="minimizeImage" value="b_minimize.gif"/>
        <String name="maximizeImage" value="b_maximize.gif"/>
        <String name="normalizeImage" value="b_normal.gif"/>
        <String name="attachImage" value="b_attach.gif"/>
        <String name="detachImage" value="b_new_window.gif"/>
        <String name="editImage" value="b_edit.gif"/>
    </Collection>
    <Collection name="DustyTheme">
        <String name="description" value="Dusty"/>
        <String name="bgColor" value="white"/>
        <String name="titleBarColor" value="#330000"/>
        <String name="fontColor" value="black"/>
        <String name="borderColor" value="#330000"/>
        <String name="borderWidth" value="1"/>
        <String name="fontFace" value="Sans-serif"/>
        <String name="fontSize" value="3"/>
        <String name="activeBulletImage" value="dusty_bullet_on.gif"/>
        <String name="inactiveBulletImage" value="dusty_bullet_off.gif"/>
        <String name="brandImage" value="Dusty_FFFFFF.gif"/>
        <String name="brandImage2" value="spacer.gif"/>
        <String name="brandImageBgColor" value="#FFFFFF"/>
        <String name="brandImage2BgColor" value="#FFFFFF"/>
        <String name="brandBgColor" value="#FFFFFF"/>
        <String name="headerBgColor" value="#FFFFFF"/>
        <String name="brandImageWidth" value="243"/>
        <String name="headerFontColor" value="#330000"/>
        <String name="headerText" value="sansSerif11Font"/>
        <String name="tabNotchImage" value="tabNotch.gif"/>
        <String name="tabColor" value="#999966"/>
        <String name="tabFontColor" value="#000000"/>
        <String name="contentLayoutLinkColor" value="#CC9900"/>
        <String name="contentLayoutText" value="sansSerif10Font"/>
        <String name="linkSeparatorColor" value="#ED471E"/>
        <String name="tableBgColor" value="#CCCC99"/>
        <String name="titleFontColor" value="#FFFFCC"/>
        <String name="titleText" value="sansSerif10Font"/>
```

Code Example 13-1 Display Profile Definition for GlobalThemes *(Continued)*

```
        <String name="channelHighlightColor" value="#ffffff"/>
        <String name="channelLinkColor" value="#3a2eb5"/>
        <String name="previewImage" value="Dusty_preview.gif"/>
        <String name="helpImage" value="b_help.gif"/>
        <String name="removeImage" value="b_remove.gif"/>
        <String name="minimizeImage" value="b_minimize.gif"/>
        <String name="maximizeImage" value="b_maximize.gif"/>
        <String name="normalizeImage" value="b_normal.gif"/>
        <String name="attachImage" value="b_attach.gif"/>
        <String name="detachImage" value="b_new_window.gif"/>
        <String name="editImage" value="b_edit.gif"/>
    </Collection>
    <Collection name="CollegiateTheme">
        <String name="description" value="Collegiate"/>
        <String name="bgColor" value="white"/>
        <String name="titleBarColor" value="#000033"/>
        <String name="fontColor" value="black"/>
        <String name="borderColor" value="#000033"/>
        <String name="borderWidth" value="1"/>
        <String name="fontFace" value="Sans-serif"/>
        <String name="fontSize" value="3"/>
        <String name="activeBulletImage" value="collegiate_bullet_on.gif"/>
        <String name="inactiveBulletImage" value="collegiate_bullet_off.gif"/>
        <String name="brandImage" value="Collegiate_FFFFFF.gif"/>
        <String name="brandImage2" value="spacer.gif"/>
        <String name="brandImageBgColor" value="#FFFFFF"/>
        <String name="brandImage2BgColor" value="#FFFFFF"/>
        <String name="brandBgColor" value="#FFFFFF"/>
        <String name="headerBgColor" value="#FFFFFF"/>
        <String name="brandImageWidth" value="243"/>
        <String name="headerFontColor" value="#000000"/>
        <String name="headerText" value="sansSerif11Font"/>
        <String name="tabNotchImage" value="tabNotch.gif"/>
        <String name="tabColor" value="#EED23A"/>
        <String name="tabFontColor" value="#000000"/>
        <String name="contentLayoutLinkColor" value="#EED23A"/>
        <String name="contentLayoutText" value="sansSerif10Font"/>
        <String name="linkSeparatorColor" value="#ED471E"/>
        <String name="tableBgColor" value="#CCCCCC"/>
        <String name="titleFontColor" value="#ffffff"/>
        <String name="titleText" value="sansSerif10Font"/>
        <String name="channelHighlightColor" value="#ffffff"/>
        <String name="channelLinkColor" value="#3a2eb5"/>
        <String name="previewImage" value="Collegiate_preview.gif"/>
        <String name="helpImage" value="b_help.gif"/>
        <String name="removeImage" value="b_remove.gif"/>
        <String name="minimizeImage" value="b_minimize.gif"/>
        <String name="maximizeImage" value="b_maximize.gif"/>
        <String name="normalizeImage" value="b_normal.gif"/>
        <String name="attachImage" value="b_attach.gif"/>
        <String name="detachImage" value="b_new_window.gif"/>
        <String name="editImage" value="b_edit.gif"/>
    </Collection>
    <Collection name="NeonTheme">
```

Code Example 13-1 Display Profile Definition for GlobalThemes *(Continued)*

```
        <String name="description" value="Neon"/>
        <String name="bgColor" value="white"/>
        <String name="titleBarColor" value="#CCFF33"/>
        <String name="fontColor" value="black"/>
        <String name="borderColor" value="#CCFF33"/>
        <String name="borderWidth" value="1"/>
        <String name="fontFace" value="Sans-serif"/>
        <String name="fontSize" value="3"/>
        <String name="activeBulletImage" value="neon_bullet_on.gif"/>
        <String name="inactiveBulletImage" value="neon_bullet_off.gif"/>
        <String name="brandImage" value="Neon_000000.gif"/>
        <String name="brandImage2" value="spacer.gif"/>
        <String name="brandImageBgColor" value="#000000"/>
        <String name="brandImage2BgColor" value="#000000"/>
        <String name="brandBgColor" value="#000000"/>
        <String name="headerBgColor" value="#000000"/>
        <String name="brandImageWidth" value="243"/>
        <String name="headerFontColor" value="#FFFFFF"/>
        <String name="headerText" value="sansSerif11Font"/>
        <String name="tabNotchImage" value="black_tabend.gif"/>
        <String name="tabColor" value="#FF0000"/>
        <String name="tabFontColor" value="#000000"/>
        <String name="contentLayoutLinkColor" value="#000000"/>
        <String name="contentLayoutText" value="sansSerif10Font"/>
        <String name="linkSeparatorColor" value="#FF0000"/>
        <String name="tableBgColor" value="#FFFFFF"/>
        <String name="titleFontColor" value="#000000"/>
        <String name="titleText" value="sansSerif10Font"/>
        <String name="channelHighlightColor" value="#ffffff"/>
        <String name="channelLinkColor" value="#3a2eb5"/>
        <String name="previewImage" value="Neon_preview.gif"/>
        <String name="helpImage" value="b_help.gif"/>
        <String name="removeImage" value="b_remove.gif"/>
        <String name="minimizeImage" value="b_minimize.gif"/>
        <String name="maximizeImage" value="b_maximize.gif"/>
        <String name="normalizeImage" value="b_normal.gif"/>
        <String name="attachImage" value="b_attach.gif"/>
        <String name="detachImage" value="b_new_window.gif"/>
        <String name="editImage" value="b_edit.gif"/>
    </Collection>
    </Collection>
    <String name="UserTheme" value="SunTheme" propagate="false"/>
    <Collection name="CustomTheme" propagate="false">
        <String name="bgColor" value=""/>
        <String name="titleBarColor" value=""/>
        <String name="fontColor" value=""/>
        <String name="borderColor" value=""/>
        <String name="borderWidth" value=""/>
        <String name="fontFace" value=""/>
        <String name="fontSize" value=""/>
        <String name="activeBulletImage" value=""/>
        <String name="inactiveBulletImage" value=""/>
        <String name="brandImage" value=""/>
        <String name="brandImage2" value=""/>
```

Code Example 13-1 Display Profile Definition for GlobalThemes *(Continued)*

```
        <String name="brandImageBgColor" value=""/>
        <String name="brandImage2BgColor" value=""/>
        <String name="brandBgColor" value=""/>
        <String name="headerBgColor" value=""/>
        <String name="brandImageWidth" value=""/>
        <String name="headerFontColor" value=""/>
        <String name="headerText" value=""/>
        <String name="tabNotchImage" value=""/>
        <String name="tabColor" value=""/>
        <String name="tabFontColor" value=""/>
        <String name="contentLayoutLinkColor" value=""/>
        <String name="contentLayoutText" value=""/>
        <String name="linkSeparatorColor" value=""/>
        <String name="tableBgColor" value=""/>
        <String name="titleFontColor" value=""/>
        <String name="titleText" value=""/>
        <String name="channelHighlightColor" value=""/>
        <String name="channelLinkColor" value=""/>
        <String name="previewImage" value=""/>
        <String name="helpImage" value=""/>
        <String name="removeImage" value=""/>
        <String name="minimizeImage" value=""/>
        <String name="maximizeImage" value=""/>
        <String name="normalizeImage" value=""/>
        <String name="attachImage" value=""/>
        <String name="detachImage" value=""/>
        <String name="editImage" value=""/>
    </Collection>
</Collection>
```

Theme Properties

The following is a list of theme properties that can be defined, modified, and / or customized in the display profile document. Please reference to the "Glossary of Terms" on page 170 for more detailed description for the theme properties.

In the following table, column one lists the theme property name and column two provides a brief description of the corresponding theme property.

`activeBulletImage`	activeBulletGraphics
`inactiveBulletImage`	inactiveBulletGraphics
`brandImage`	logo image
`brandImage2`	Product name image
`brandImageBgColor`	header logo bg color

`brandImage2BgColor`	header product name bg color
`brandBgColor`	header link box bg color
`headerBgColor`	header bg color and footer bg color
`headerFontColor`	header font color
	footer font color
`headerText`	header font size
	footer font size
	header font face
	footer font face
`tabNotchImage`	tabNotch image
`titleText`	selected tab font face
	selected tab font size
	unselected tab font face
	unselected tab font size
	channel title font face
	channel title font size
`titleFontColor`	selected tab font color
	channel title font color
`fontSize`	channel font size
	channel link font size
`fontFace`	channel font face
	channel link font face
`titleBarColor`	selected tab bg color
`borderColor`	channel title bar bg color
	content / layout bar color
	channel border color
	page piping color (bottom)
	button bg color

`tabColor`	unselected tab bg color
	secondary channel title bar color
`tabFontColor`	unselected tab font color
`bgColor`	channel bg color
`fontColor`	channel font color
`borderWidth`	channel border width
`tableBgColor`	table bg color
	page piping, top
`channelHightlightColor`	highlight color for channel content (as seen in the Placida theme JSP channel)
`linkSeparatorColor`	link separator color (in the toolbar, between Content and Layout)
	footer link separator color
`channelLinkColor`	channel link color
`contentLayoutLinkColor`	content / layout link color
`contentLayoutText`	content / layout link font size
	content / layout link font face
`brandImageWidth`	brand image width
`previewImage`	preview image (on preset themes page)
`removeImage`	remove image (for the channel title bar)
`detachImage`	detach image (for the channel title bar)
`helpImage`	help image (for the channel title bar)
`editImage`	edit image (for the channel title bar)
`minimizeImage`	minimize image (for the channel title bar)
`maximizeImage`	maximize image (for the channel title bar)
`normalizeImage`	normalize image (in the maximized channel)
`attachImage`	attach image (in the popup window)

Glossary of Terms

The following table give some detailed description of where the theme attributes are actually used in the desktop.

Table 13-1 Glossary of Terms

Term	Description
Header	The banner area at the top of the portal page. Contains the branding and the global links.
Bullet graphics	The “dot” graphics that go next to the global links in the header
Logo, product name, link	The three areas in the header for the Sun theme
Footer	The narrow banner at the bottom of the portal page. Contains the global links
Tab notch	The graphic that goes in the upper left corner of the tab table cells
Selected tab	The tab whose contents are displayed
Unselected tab	The other tabs whose contents are not seen
Content/layout bar	The tool bar underneath the tabs that contains the content and layout links
Channel	The data containers displayed inside each tab
Page piping	The narrow bands of color at the top and bottom of the portal page
Table background	The areas the channels sit in
Highlight color for channel content	A contrasting color for tables inside channels
Secondary channel title bar	An extra color bar beneath the standard channel title bar
Link separator	A pipe used between links in the content/layout bar

Adding and Customizing Global Themes

When you add (or customize) a global theme, all channels see the change, as themes are a global property for all channels.

➤ To Add a Theme to the Sample Portal

1. Develop the display profile XML definition for the new theme and ensure that the new collection has all thirty eight (38) properties defined in the display profile.

   The collection can be added either using the Identity Server administration console, or via the `dpadmin` command.

2. Copy new images into the *portal-server-install-root*/SUNWps/web-src. That is:
   - activeBulletImage, inactiveBulletImage, brandImage, brandImage2, previewImage: these images must be copied in to the *portal-server-install-root*/SUNWps/web-src/images directory.
   - helpImage, removeImage, minimizeImage, maximizeImage, normalImage, attachImage, detachImage, editImage: these images must be copied in to the *portal-server-install-root*/SUNWps/web-src/desktop/images directory.
   - tabNotchImage must be copied in to the *portal-server-install-root*/SUNWps/web-src/desktop/tabs/images directory.
3. Run the deploy command to deploy the image files. For example, type:

   portal-server-install-root/SUNWps/bin/deploy redeploy -deploy_admin_password *password*
4. Verify that the new theme shows up on the Desktop's Theme page.

➤ **To Customize the Current Themes**

Change the theme values in the display profile. You can modify the theme properties from the administration console, or by using the dpadmin command to load the XML fragment.

Changing in the administration console is easier, since you want to pin point some specific properties, and change the values. See "Editing the Display Profile" on page 38 for loading the display profile into LDAP by using the dpadmin command.

In the sample portal, the *portal-server-install-root*/SUNWps/samples/desktop/dp-org.xml file defines eight themes. See also, "GlobalThemes Display Profile Definition" on page 160 for the default display profile XML fragment for GlobalThemes.

➤ **To Change the Text**

The font families and font sizes are combined and defined in the following theme attributes:

- headerText
- titleText
- contentLayoutText

The value of these attributes is actually a class defined in the desktop *Web-Container-Instance*/desktop/style.css file. In the Desktop style.css file, there are predefined font family + font size as follows:

```
.sansSerif12Font { font-family: sans-serif; font-size: 12pt }
.sansSerif11Font { font-family: sans-serif; font-size: 11pt }
.sansSerif10Font { font-family: sans-serif; font-size: 10pt }
.sansSerif9Font { font-family: sans-serif; font-size: 9pt }
.sansSerif8Font { font-family: sans-serif; font-size: 8pt }
.sansSerif6Font { font-family: sans-serif; font-size: 6pt }
.monospace12Font { font-family: monospace; font-size: 12pt }
.monospace11Font { font-family: monospace; font-size: 11pt }
.monospace10Font { font-family: monospace; font-size: 10pt }
.monospace9Font { font-family: monospace; font-size: 9pt }
.monospace8Font { font-family: monospace; font-size: 8pt }
.monospace6Font { font-family: monospace; font-size: 6pt }
.serif12Font { font-family: serif; font-size: 12pt }
.serif11Font { font-family: serif; font-size: 11pt }
.serif10Font { font-family: serif; font-size: 10pt }
.serif9Font { font-family: serif; font-size: 9pt }
.serif8Font { font-family: serif; font-size: 8pt }
.serif6Font { font-family: serif; font-size: 6pt }
.verdana12Font { font-family: verdana; font-size: 12pt }
.verdana11Font { font-family: verdana; font-size: 11pt }
.verdana10Font { font-family: verdana; font-size: 10pt }
.verdana9Font { font-family: verdana; font-size: 9pt }
.verdana8Font { font-family: verdana; font-size: 8pt }
.verdana6Font { font-family: verdana; font-size: 6pt }
```

To change the font for the header text, title text, and content and layout text, please use one of the predefined class name, or, add new class definition in the `style.css` file, and then use it.

➤ To Change the Sample Anonymous Desktop Theme

Anonymous users cannot choose their own theme as this is determined by the display profile. However, an administrator who has permission to edit the anonymous user's display profile can change the theme for the anonymous Desktop.

1. Create a temporary file containing the theme definition.

   For example,

   `vi newtheme.xml`

```
<?xml version="1.0" encoding="utf-8" standalone="no"?>
<!DOCTYPE DisplayProfile SYSTEM "jar://resources/psdp.dtd">
    <Collection name="GlobalThemes">
        <Collection name="NeonTheme">
            <String name="description" value="Neon"/>
```

```
            <String name="bgColor" value="white"/>
            <String name="titleBarColor" value="#CCFF33"/>
            <String name="fontColor" value="black"/>
            <String name="borderColor" value="#CCFF33"/>
            <String name="borderWidth" value="1"/>
            <String name="fontFace" value="Sans-serif"/>
            <String name="fontSize" value="3"/>
            <String name="activeBulletImage" value="neon_bullet_on.gif"/>
            <String name="inactiveBulletImage"
value="neon_bullet_off.gif"/>
            <String name="brandImage" value="Neon_000000.gif"/>
            <String name="brandImage2" value="spacer.gif"/>
            <String name="brandImageBgColor" value="#000000"/>
            <String name="brandImage2BgColor" value="#000000"/>
            <String name="brandBgColor" value="#000000"/>
            <String name="headerBgColor" value="#000000"/>
            <String name="brandImageWidth" value="243"/>
            <String name="headerFontColor" value="#FFFFFF"/>
            <String name="headerText" value="sansSerif11Font"/>
            <String name="tabNotchImage" value="black_tabend.gif"/>
            <String name="tabColor" value="#FF0000"/>
            <String name="tabFontColor" value="#000000"/>
            <String name="contentLayoutLinkColor" value="#000000"/>
            <String name="contentLayoutText" value="sansSerif10Font"/>
            <String name="linkSeparatorColor" value="#FF0000"/>
            <String name="tableBgColor" value="#FFFFFF"/>
            <String name="titleFontColor" value="#000000"/>
            <String name="titleText" value="sansSerif10Font"/>
            <String name="channelHighlightColor" value="#ffffff"/>
            <String name="channelLinkColor" value="#3a2eb5"/>
            <String name="previewImage" value="Neon_preview.gif"/>
            <String name="helpImage" value="b_help.gif"/>
            <String name="removeImage" value="b_remove.gif"/>
            <String name="minimizeImage" value="b_minimize.gif"/>
            <String name="maximizeImage" value="b_maximize.gif"/>
            <String name="normalizeImage" value="b_normal.gif"/>
            <String name="attachImage" value="b_attach.gif"/>
            <String name="detachImage" value="b_new_window.gif"/>
            <String name="editImage" value="b_edit.gif"/>
        </Collection>
    </Collection>
    <String name="UserTheme" value="NeonTheme" propagate="false"/>
```

2. Run the `dpadmin` command with the `add` sub-command to load the display profile.

For example, to load the XML fragment to the anonymous user's node,

portal-server-install-root`/SUNWps/bin/dpadmin add -u "uid=amAdmin,ou=People,o=sesta.com,o=isp" -w` *password* `-d "uid=anonymous,ou=people,o=sesta.com,o=isp" newtheme.xml`

3. Log in to the administration console and change the global property. That is:
   a. Select Users from the View pull-down menu.
   b. Select Authlessanonymous
   c. Select View->Portal Desktop on the right panel and select the Edit link.

      A popup window displayed.
   d. Select Channel and Container Management.
   e. Select Display Profile: Edit Properties and change User Theme to “NeonTheme”.
   f. Click on Save to save the settings.
4. Log out of the administration console.

Chapter 14

Customizing the Service Providers

This chapter provides common customization tasks for modifying the Search provider and Discussion provider.

This chapter contains the following sections:

- Overview of Customizing the Service Providers
- Tips for Customizing the Service Providers
- Customizing the Search Provider
- Customizing the Discussion Channels

Overview of Customizing the Service Providers

The Sun Java System Portal Server software includes a search service and discussion service provider.

Overview of Customizing the Search Provider

The Search provider (SearchProvider) furnishes a basic reference user interface that contains both search and browse functionality. Search functionality includes basic search mode, and advanced search for more complex searches. You can perform specific field searches in advanced search mode. For example, while in advanced mode, you can search within the title, URL, last modified date, author, and so on.

SearchProvider provides a link for category browsing. In addition, you can create a taxonomy for the Search Engine along with category filter rules. You can browse through the taxonomy tree and view documents within a category through the Search provider interface.

Search Provider Design

The Search provider uses JSPProvider to access the Portal Server back end services. The Search provider users JavaServer Pages™ (JSP™) helper tag libraries to avoid using Java™ scriptlets. The `searchServer` is a global service list type attribute that is configured and updated at installation time. The Search provider is responsible for directing the search request to the appropriate back end Search Engine server.

See Chapter 2, "Display Profile Properties" for the display profile properties you can set for the provider.

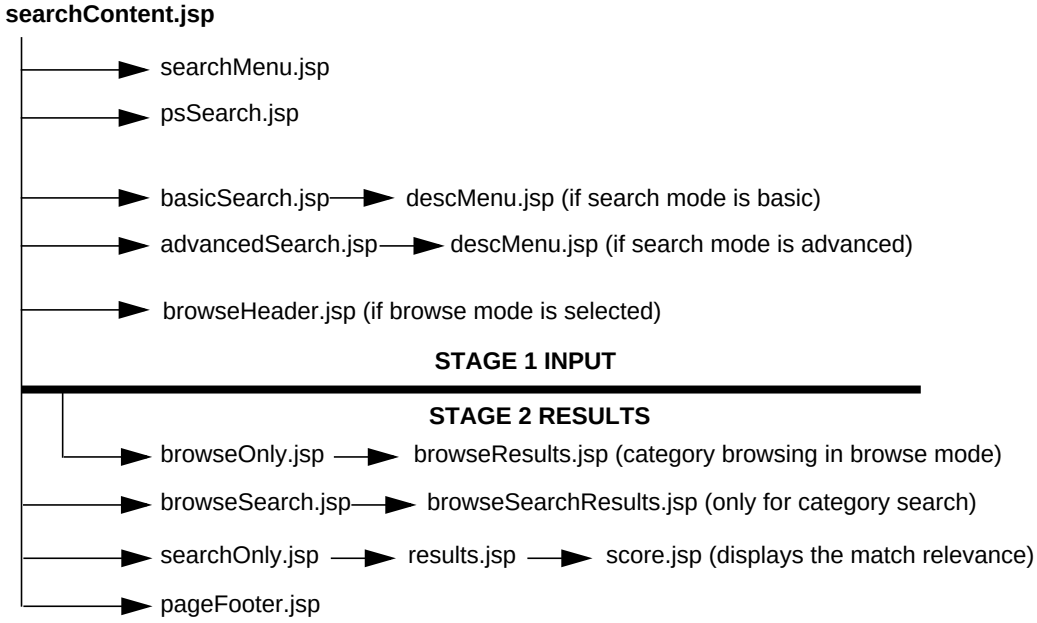
Search JavaServer Pages and Tag Libraries

The Search provider consists of two stages—input form and results—and the JSPs used by the Search provider fall into one of those two stages.

Figure 14-1 explains the JSP layout for `searchContent.jsp`. In the input stage (Stage 1), `searchContent.jsp` makes use of `searchMenu.jsp` and `psSearch.jsp` to set up the initial interface. The `basicSearch.jsp` file is used for a basic search and `advancedSearch.jsp` file for an advanced search. The description menu— that is, the Full, Brief, and Title menus—is displayed for both basic and advanced searches by `descMenu.jsp` file. The `browseHeader.jsp` file defines the browse interface.

In the results stage (Stage 2), one of three JSPs is used: `browseOnly.jsp` file sets and executes the parameters for category browsing using the Search tag library, and includes the `browseResults.jsp` page; `browseSearch.jsp` file sets and executes the parameters for searching and browsing within categories using the Search tag library and includes `browseSearchResults.jsp`; and `searchOnly.jsp` file sets and executes the parameters for search using search the Search tag library and includes `results.jsp` and `score.jsp` for the match relevance. The `pageFooter.jsp` file displays the list of pages, Next, and Previous links.

Figure 14-1 JSP Layout for `searchContent.jsp`



See Appendix C, "JavaServer Pages Reference" for more information on the Search JSP files.

The Search JSPs use the following tag libraries, which ship with the Portal Server software:

- Jakarta tag library, for any generic tags
- Desktop tag library, for the Portal Server software related information
- Search tag library to cover all Search Engine server access functionality and to provide a tag-based wrapper for the existing public search and SOIF API

See Appendix D, "JavaServer Pages Tag Library Reference" for more information.

Overview of Customizing the Discussion Provider

The DiscussionProvider is JSPProvider based and hence customizable. It uses the Desktop themes. It retrieves data from the back end Search service using search taglibs and API. The discussions and comments are stored as separate Resource Descriptors (RDs) in the discussion database.

The DiscussionProvider includes features such as discussion threads, starting discussions based on documents or new topics, searching discussions, and rating discussions. By default, the Discussions channel is available on the sample portal for anonymous users. However, an anonymous user cannot subscribe to a discussion or edit the Discussion channel.

The DiscussionProvider supports a full view (via the Discussions channel) and a lite view (via the DiscussionLite channel.) It has the following main functions:

- Start a new discussion from the discussion channel.
- Start a new discussion based on web documents from the search channel.
- Add a comment to an existing discussion or post a reply to an existing discussion.
- Rate all discussions and comments. Note that the displayed ratings are based on an algorithm such that the rating for any comment goes up gradually. For example, a comment has to be rated important three times before it is marked as important.
- Search all discussions and search within a discussion. These functions are routed to the search provider. The `displaySearch` property can be disabled if the search feature is not required in the discussion channel. Users can also search by rating in Advance Search.
- Authenticated users can choose to subscribe to a particular discussion by selecting the subscribe link. The request is handled by the SubscriptionProvider. The `displaySubscription` property can be disabled if the feature is not required. By default, the value is true.

A Discussion Lite view retrieves main posts sorted by last-modified date and has pagination so users can access older discussions. View discussion displays each discussion subtree. The main item is displayed in detail and the subtree is displayed below the main item. View discussion includes:

- Several filters on the page. A document display can be based on filters such as document rating (irrelevant, routine, interesting, important, and must read).
- Display preference can be set to threaded or flat display.

Expansion threshold helps to control displayed items in the subtree. The users can choose to expand only highly rated documents, or expand all or collapse all. Default value is collapse all. Expand all will expand all the filtered comments. It will also show a description of the discussion, provide a menu for rating the discussion, and allow the user to post a reply.

Discussion Service Channels

The DiscussionLite channel and the Discussions channel are based on the DiscussionProvider.

DiscussionLite Channel

The DiscussionLite channel displays the top twenty discussion titles (which can be reconfigured) and the date. The discussions are sorted by creation date (last modified) and the newest discussion is displayed first. The DiscussionLite channel view has links to view each discussion, view all discussions, and start a new discussion. All these links target the Discussions channel which gets displayed in the JSPDynamicSingleContainer.

Properties for this channel can be configured from the administration console. By default, there are no user editable properties for this channel.

Discussions Channel

The Discussions channel includes a full view that:

- Shows detailed descriptions for the top eight discussions sorted in descending order. This can be reconfigured via the channel edit page.
- Includes pagination so that users can see all the discussions.
- Supports search. The search returns discussion and comment results.

Discussion JavaServer Pages and Tag Libraries

Similar to the search channel JSPs, the discussion channel JSPs have a query portion, a display portion, and use Desktop themes.

For more information on the channel specific JSP files, see Appendix C, “JavaServer Pages Reference.”

Tips for Customizing the Service Providers

This section provides some basic tips for customizing the search and discussion providers.

Debugging the Service Providers

The Portal Server software provides files to help debug the Search and Discussion providers.

The following directory contains various search log files:

`/var/opt/SUNWps/https-psservername/portal`

The following search log file records the search query sent to the Search Engine by the Search server:

`/var/opt/SUNWps/https-psservername/portal/logs/rdm.log`

See the *Portal Server Administration Guide* for more information about the Search log files.

Location of JavaServer Pages

JavaServer Pages for the Search channel are in the `/etc/opt/SUNWps/desktop/default/Search` directory.

JavaServer Pages for the DiscussionLite channel are in the `/etc/opt/SUNWps/desktop/default/DiscussionLite` directory.

JavaServer Pages for the Discussions channel are in the `/etc/opt/SUNWps/desktop/default/Discussions` directory.

Modifying JavaServer Pages

When you modify statically included JavaServer Pages, be sure to run the touch command, otherwise no changes are reflected. You need to either run the touch command on the top-level JSP file or on all JSP files. For example,

```
touch searchContent.jsp
```

or

```
touch *.jsp
```

See also "JavaServer Page Caching Information" on page 298 and "Recompiling JSPs" on page 297.

Accessing Channels Directly

You can access the search channel directly at the following URL:

`http://`*`server`*`:`*`port`*`/portal/dt?provider=JSPDynamicSingleContainer&JSPDynamicSingleContainer.selectedChannel=Search&last=false&action=content`

Modify all the links to use these extra parameters in the URL. For example, edit `searchMenu.jsp` file as follows:

```
<nobr>  <a class=noUnderline href="<%=dpurl%>?mode=basic">Basic
Search</a>  </nobr>
```

Replace the bold portion with:

```
http://server:port/portal/dt?provider=JSPDynamicSingleContainer&JSPDynamicSing
leContainer.selectedChannel=Search&last=false&action=content
```

You can access the Discussion channel directly at the following URL:

```
http://server:port/portal/dt?provider=JSPDynamicSingleContainer&JSPDynamicSing
leContainer.selectedChannel=Discussions&last=false&action=content
```

Customizing the Search Provider

This section describes how to perform some common customizations on the Search provider.

➤ **To Modify the Default Search Server**

When you install the Portal Server software, the Search provider is linked back to the Search server by default. That is to say, the Search provider can connect to any Search server, but by default the value for the `searchServer` property is set to the following:

```
http://server:port/portal/search
```

The `searchServer` property is normally initialized to the default portal server instance during the sample portal installation but this property is not set when you create a new organization or a new custom search channel. See *Portal Server Administration Guide* for information on configuring the `searchServer` property for the Search provider.

1. Log in to the Sun Java System Identity Server software administration console as administrator.
2. Choose Organizations from the Show menu in User Management.

   All created organizations display in the navigation pane.
3. Navigate to the organization or sub-organization that you want to configure Search for.
4. Choose Services from the Show menu.

5. Click the properties arrow next to Desktop in the Navigation pane.
6. Click Channel and Container Management.

   The Channels page appears.
7. In the Channels section, click the Edit link for Search.

   The Edit Channel page appears for the Search channel.
8. Edit the `searchServer` property with the Search server's name.
9. Click Save.

➤ To Add last-modified to the Search Result Display

1. Modify `searchOnly.jsp` file by adding last-modified to the list of `viewAttributes`.

   For example:

```
<search:setViewAttributes
viewAttributes="hl-url,hl-title,hl-description,score,content-length,hl-clas
sification,last-modified"/>
```

2. Modify `results.jsp` file to display the last-modified date for document results using the SOIF `getValue` tag.

   For example:

```
<% if (formbean.getDescription().equals("full")) { %>
    <FONT color=<%=tFontColor%> face=<%=tFontFace%>><search:getValue
soifAttribute="description" escape="false"/></FONT><BR>
    <FONT color=#707070 face=<%=tFontFace%>><search:getURL
escape="true"/><BR>
    <search:getValue soifAttribute="content-length" id="sz"/>
    <search:getValue soifAttribute="last-modified"/><BR>
<% } %>
```

3. Run the `touch` command.

   For example, type **`touch *.jsp`**.
4. Reload the Desktop to verify the change.

➤ **To Remove content-length from Search Results**

1. (Optional) Modify `searchOnly.jsp` file by removing content-length from the list of viewAttributes.

   The line to modify is the following:

```
<search:setViewAttributes viewAttributes=
"hl-url,hl-title,hl-description,score,content-length,classification
hl-classification"/>
```

   Remove content-length from this line.

2. Modify `results.jsp` file by removing the line that displays the content-length.

   The line to modify is the following:

```
<% } else if (formbean.getDescription().equals("full")) { %>
    <FONT size=-1 color=<%=tFontColor%>
face=<%=tFontFace%>><search:getValue soifAttribute="hl-description"
escape="false"/></FONT><BR>
    <FONT size=-1 color=#707070 face=<%=tFontFace%>><search:getValue
soifAttribute="hl-url" escape="false"/><BR>
    <search:getValue soifAttribute="content-length" id="sz"/>
```

3. Remove `<search:getValue soifAttribute="content-length" id="sz"/>` from this file.

4. Run the `touch` command.

   For example, type **`touch *.jsp`**.

5. Reload the Desktop to verify the change.

➤ **To Display the Total Number of Documents in the Search Result Status Message**

In this procedures, Steps 1 and 2 are independent of each other. If desired, run the `touch` command after Step 1 to see the results.

1. Modify `results.jsp` by changing the search status line to add the `<search:getTotalDocuments/>` tag.

   For example:

```
<NOBR><B>Document matches <search:getFirstHit/> - <search:getToHit/> (of
<search:getHitCount/>)</B> out of <search:getTotalDocuments/></NOBR><BR>
```

This results in the following display:

```
Document matches 1 - 6 (of 6) out of 37
```

2. Change `browseResults.jsp` by adding the `<search:getTotalDocuments/>` tag to the search status.

   For example:

```
<FONT color="<%=tFontColor%>" face="<%=tFontFace%>"
size="-1"><b>Subcategories <search:getFirstHit/> - <search:getToHit/> (of
<search:getHitCount/>)</B> out of <search:getTotalDocuments/></FONT><br>
```

This results in the following display:

```
Category matches 1 - 2 (of 2) out of 86
```

3. Run the `touch` command.

   For example, type **`touch *.jsp`**.

4. Reload the Desktop to verify the change.

➤ **To Remove author from the Advanced Search Interface**

1. Comment our or remove the author related HTML from the `advancedSearch.jsp` file.

   For example:

```
<!--     -->
<!-- To disclude the "author" row, remark out the following section -->
<!--     -->
<TR>
    <td valign=middle align=right height=40><FONT color=<%=tFontColor%>
face=<%=tFontFace%>><nobr> <LABEL FOR="advAuthor">Author</LABEL>
    <SELECT NAME="authorOp">
    <OPTION VALUE=<%=SearchContext.CONTAIN%>
<%=formbean.authorOpSelection(SearchContext.CONTAIN)%>>does</OPTION>
```

```
    <OPTION VALUE=<%=SearchContext.NOTCONTAIN%>
<%=formbean.authorOpSelection(SearchContext.NOTCONTAIN)%>>does not</OPTION>
    </SELECT>contain </FONT></nobr></TD>
    <td valign=middle align=left height=40><INPUT TYPE="text"
NAME="authorVal" id="advAuthor" VALUE=
"<%=SearchContext.htmlEncode(formbean.getAuthorVal())%>"></TD>
</TR>
```

2. Comment out author- related lines in `advQuery.jsp` file.

```
// h = new HashMap();
    // h.put(SearchContext.OPERAND, "author");
    // h.put(SearchContext.OPERATION, formbean.getAuthorOp());
    // h.put(SearchContext.VALUE, formbean.getAuthorVal());
    // l.add(h);

    h = new HashMap();
    h.put(SearchContext.OPERAND, "title");
```

3. Run the `touch` command.

   For example, type **`touch *.jsp`**.

4. Reload the Desktop to verify the change.

➤ To Add a New Field to Advanced Search

1. Uncomment the keywords section in `advancedSearch.jsp` file.

```
<!--     -->
<!-- To Include the "Keywords" row, unremark the following section -->
<!--
<TR>
<td valign=middle align=right height=40><FONT color=<%=tFontColor%>
face=<%=tFontFace%>><nobr> <LABEL FOR="advKeywords">Keywords</LABEL>
<SELECT NAME="keywordsOp">
<OPTION VALUE=<%=SearchContext.CONTAIN%>
<%=formbean.keywordsOpSelection(SearchContext.CONTAIN)%>>does</OPTION>
<OPTION VALUE=<%=SearchContext.NOTCONTAIN%>
<%=formbean.keywordsOpSelection(SearchContext.NOTCONTAIN)%>>does
not</OPTION>
</SELECT>contain </FONT></nobr></TD>
<td valign=middle align=left height=40><INPUT TYPE="text" NAME="keywordsVal"
id="advKeywords" VALUE=
"<%=SearchContext.htmlEncode(formbean.getKeywordsVal())%>"></TD>
</TR>
-->
```

Remove the `<!--` and `-->` comment marks from this section.

2. Add the keywords to `advQuery.jsp` file.

```
h = new HashMap();
h.put(SearchContext.OPERAND, "keywords");
h.put(SearchContext.OPERATION, formbean.getKeywordsOp());
h.put(SearchContext.VALUE, formbean.getKeywordsVal());
l.add(h);
```

3. Run the `touch` command.

   For example, type **`touch *.jsp`**.

4. Reload the Desktop to verify the change.

Customizing the Discussion Channels

Customizing DiscussionLite Channel

➤ **To Customize the DiscussionLite Channel Link Display Window**

1. Change directory to `/etc/opt/SUNWps/desktop/default/DiscussionLite` directory and edit the following JSP files.

   - `display.jsp` - In this file, comment out or delete the following line:

```
<a
href="<%=desktopPathInfo%>?last=false&Discussions_dmode=vl&did=<%=Encoder.u
rlEncode(url)%>"><B><search:getValue soifAttribute="title" escape="true"
truncate="28"/></B></a>
```

   - `discussionLiteContent.jsp` - In this file, comment out or delete the following lines. :

```
Map pathInfo = new HashMap();
pathInfo.put("action", "content");
pathInfo.put("provider", "JSPDynamicSingleContainer");
pathInfo.put("JSPDynamicSingleContainer.selectedChannel", "Discussions");
pathInfo.put("last", "false");
pageContext.setAttribute("pathInfo", pathInfo);
<dtpc:getDesktopURL id="desktopPathInfo" pathinfo="$pathInfo"/>
```

2. Replace all occurrences of `desktopPathInfo` with `dt`.

   The `desktopPathInfo` ensures that links are always displayed in the Discussions channel in a JSPDynamicSingleContainer. Remove this if you want links to be displayed in the Discussions Channels on the same tab. For example, replace the following line:

```
<td align=center><font size="-1"><a target="ps_main" href=
"<%=desktopPathInfo%>?Discussions_dmode=cmt">New Discussion</a></font>
```

with the following:

```
<td align=center><font size="-1"><a target="ps_main" href=
"dt?Discussions_dmode=cmt">New Discussion</a></font>
```

This procedure will work only if Discussions and DiscussionLite are displayed on the same tab as in the sample portal. DiscussionLite links will be displayed in the Discussions channel on the right side on the collaboration tab.

➤ **To Display DiscussionLite on the Front tab**

1. Modify MyFrontPageTabContainer and add DiscussionLite to the available and selected lists. For example:

   In the administration console:

   a. Log in and select Services (from the View pull-down menu) for your organization.

   b. Select Portal Desktop.

   c. Select Channel and Container Management link and MyFrontPageTabContainer from the list of containers.

      The page to edit the MyFrontPageTabContainer is displayed.

   d. Select DiscussionLite from the Existing Channels list and select the Add arrow pointing towards Available and Visible list.

   e. Select Save and log out.

   From the command line, edit `dp-org.xml` file to include DiscussionLite in the available and selected list. For example, add the text shown below in bold:

```
<Container name="MyFrontPageTabPanelContainer"
provider="PredefinedFrontPageTabPanelContainerProvider">
    <Properties> ...</Properties>
    <Available>
         <Reference value="DiscussionLite"/>
    </Available>
    <Selected>
         <Reference value="DiscussionLite"/>
    </Selected>
    ...
</Container>
```

2. Modify the DiscussionLite channel display profile `isEditable` property and set it to `true`.
3. Log in and verify.

Customizing Discussions Channel

➤ **To Display Additional Fields in the List View of Discussions**

1. Change directories to `/etc/opt/SUNWps/desktop/default/Discussions`.
2. Modify `query.jsp` file and add `xxx` field to `viewAttributes`.

   For example, add content-length as follows:

```
<search:setViewAttributes viewAttributes=
"url,title,description,rd-rating,author,last-modified,rd-last-changed,rd-re
ference-id,
rd-num-rating,rd-sum-rating,rd-peak-rating,rd-reference-url,content-lengt
h"/>
```

3. Add the new fields in `fullDiscussionDisplay.jsp` file wherever appropriate.

   For example:

```
<search:getValue soifAttribute="content-length"/>
```

➤ **To Modify the Sort Order in List All Discussions Page**

By default, discussions are sorted by the last-modified date/time. That is, discussions are displayed in a descending order with the latest or most recent discussion shown first.

To modify the sort order in list All Discussions page, modify the `viewOrder` property in `fullDiscussion.jsp` file. For example, you can reset the value below to `author` or `rd-last-changed`:

```
<jx:set var="viewOrder" value="-last-modified"/>
```

➤ To Modify viewHits in View Discussion Page

```
<jx:set var="hitNumber" value="500"/>
```

By default, the `viewHits` property is set to 500. Make this -1 if you want all the comments to be displayed or reduce this number to improve performance. For example, if the value is 200, only the first 200 comments will be displayed in this case (includes comments and sub-comments.) The `hitNumber` for view discussion page can be reset in the `viewDiscussion.jsp` file.

➤ To Inherit Classification and readACL

If you have classified only the parent discussion manually or modified the access control for the parent discussion, you may want to inherit those values in discussion replies as follows:

1. Change directories to `/etc/opt/SUNWps/desktop/default/Discussions`.
2. Edit `feedbackProcess.jsp` file and modify the values of `inheritClassification` and `inheritReadACL`.

   By default, these are set to `false`. Reset them to `true` if you want comments to inherit the parent's classification field and `readACL` field. Note that comments are automatically protected in this case.
3. Save the file.

➤ To Control Access to Discussions

This can be accomplished by one of the following two ways:

1. Modify the `dbname` property in the display profile for Discussions and DiscussionLite channel for each role to point to a different database.

   In this case users in one role cannot view discussions created by users in a different role.
2. Or modify the `ReadACL` of the parent discussion as it gets submitted in the database and set `inheritReadACL` to `true`.

   That is, you must search for the discussion first or search for `rd-reference-id <contains> ROOT` and modify the `readACL` field for discussions.

Chapter 15

Customizing the Desktop End-User Online Help

This chapter provides information on customizing the Sun Java System Portal Server Desktop online help.

This chapter contains the following sections:

- Overview of the Desktop End-User Online Help
- Location of the Desktop End-User Online Help HTML files
- Modifying the Desktop End-User Online Help HTML files

Overview of the Desktop End-User Online Help

The Desktop end-user online help is a collection of HTML files that is referenced in the display profile. Each provider, including the various container providers, has a display profile entry for a corresponding help file. In the display profile, each provider definition has the default value for the help file. If a channel that uses the provider has a different help file, the `helpURL` property can be defined in the channel definition also, overriding the provider's value.

The display profile entries for the online help are defined in the provider properties. The entries define the string name, `helpURL`, and its value.

The `helpURL` property is a conditional property. Multiple values can be associated with the `helpURL` property and the display profile API returns the proper value depending on the client type and locale. If your portal server is configured to serve multiple clients (such as HTML, WML) in multiple locales (such as english, french), the `helpURL` property will allow you to set up multiple help files based on the type of client and type of locale you are serving.

For example, Code Example 15-1 specifies different help files for different locales and different client type.

Code Example 15-1 HelpURL Property in Display Profile Definition

```
<ConditionalProperties condition="client" value="WML">
    <ConditionalProperties condition="locale" value="en">
        <String name="helpURL" value="en/wml/help.wml"/>
    </ConditionalProperties>
    <ConditionalProperties condition="locale" value="fr">
        <String name="helpURL" value="fr/wml/help.wml"/>
    </ConditionalProperties>
</ConditionalProperties>
<ConditionalProperties condition="client" value="HTML">
    <ConditionalProperties condition="locale" value="en">
        <String name="helpURL" value="en/html/help.html"/>
    </ConditionalProperties>
    <ConditionalProperties condition="locale" value="fr">
        <String name="helpURL" value="fr/html/help.html"/>
    </ConditionalProperties>
</ConditionalProperties>
```

The value of the `helpURL` property can be either a relative path or an absolute path. The location in Code Example 15-1 is relative to the Desktop static content root.

The static content root is the install directory of static content. By default the static content root is:

portal-server-install-root`/SUNWps/web-apps/https-`*psserver*`/portal`

NOTE References to the locations and contents of files in the web-apps directory are for information only. They do not represent the definition of an interface that you can depend on for any future release.

The relative path will be generated as:

static_content_root/doc_root/locale/helpURL_value

The doc root is defined in the display profile also. For example, if the doc root is `docs`, and the user's locale is `en_US`, with the `helpURL` value, the final value for the help location will be:

portal-server-install-root`/SUNWps/web-apps/https-`*psserver*`/portal/docs/en_US/desktop/userinfo.htm`

The following is an example of an absolute URL that defines a help file location. Use a similar format for using an absolute URL to define the location of an online help file.

```
<String name="helpURL" value="http://sesta.com/docs/desktop/userinfo.htm"/>
```

Location of the Desktop End-User Online Help HTML Files

The source location of the Desktop end-user online help files for the sample Desktop is:

portal-server-install-root/SUNWps/web-src/docs/locale/desktop

The following is a list of help files included with the sample portal. In the table, the first column lists the names of the help files. The second column contains a brief description of the help file.

addressbook.html	Describes the address book channel
bkmark.htm	Describes the bookmark channel.
calendar.html	Describes the calendar channel.
content.htm	Describes the Content link on the Desktop.
discussions.htm	Describes the discussions channel.
fdesktop.htm	Provides an overview of using the Desktop (for frames-based Desktop).
glossary.htm	Provides a glossary for the online help.
help.htm	Describes the Help link on the Desktop.
home.htm	Describes the Home link on the Desktop.
imchan.htm	Describes the instant messaging channel.
ix.htm	Provides an index for the online help.
jspchann.htm	Describes the JavaServer Pages™ channel.
layout.htm	Describes the Layout link on the Desktop.
login.htm	Describes the membership login channel.

`logout.htm`	Describes the Logout link on the Desktop.
`mailhelp.html`	Describes the mail channel.
`mlchck.htm`	Describes the mail check channel.
`myapps.htm`	Describes the My Applications channel.
`notes.htm`	Describes the notes channel.
`options.htm`	Describes the Options link on the Desktop.
`ParSample.html`	Provides an example of how the Portal `PAR` command-line utility is used to import and export a PAR file.
`rsschann.htm`	Describes the RSS channel.
`search.htm`	Describes the search channel.
`sections.htm`	Section help file (describes frames-based Desktop sections).
`subscriptions.htm`	Describes the subscriptions channel.
`tabs.htm`	Provides an overview of using a tabbed Desktop interface.
`theme.htm`	Describes the Theme link on the Desktop.
`topics.htm`	Provides a table of contents for the online help.
`urlscrpr.htm`	Describes the sample URL scraper channel.
`usechann.htm`	Provides an overview of using the Desktop channels.
`usedesk.htm`	Provides an overview of using the Desktop.
`userinfo.htm`	Describes the user information channel.
`webchann.htm`	Describes the simple web service channel.
`xmlchann.htm`	Describes the XML channel.
`yahoo.htm`	Describes the Yahoo channel.

Modifying the Desktop End-User Online Help HTML files

You can customize the end-user online help by editing the existing HTML online help files or by creating new HTML files.

Editing An Existing Help File

You can edit an existing help file to customize the content to meet specific requirements of your organization. For example, you can remove or change the `SunONE.jpg` image or meta text that is currently displayed on the sample help files, or replace the help file completely with a file of the same name.

After modifying the file, run the *portal-server-install-root*`/SUNWps/bin/deploy redeploy -deploy_admin_password` *password* command to deploy the file into the web-app location.

This method of modifying the online help files is useful if you use the sample providers that are shipped with the Portal Server software.

For example, if you use the UserInfoProvider that ships with the Sun Java System Portal Server product, the display profile for that provider already defines the `helpURL` value as `desktop/userinfo.htm`. By editing the help file `userinfo.htm`, no changes to the display profile are necessary.

See"Location of the Desktop End-User Online Help HTML Files" on page 193 for the online help file names and descriptions, and where they are installed on your system.

Creating a New Help File

You can create a new help file by creating a new HTML file. This method of customizing the online help is useful if, for example, you add a new provider to the Desktop. When you create a new help file, you must modify the display profile to contain the new `helpURL` value. You can either manually edit the display profile or use the Identity Server software administration console.

➤ **To Create a New Online Help File and to Define the `helpURL` Value Manually**

1. Create an HTML file for the provider you want to document.

2. Place your file in the appropriate directory.

   You can place your custom help files under the web server document root, in the directory specified as root by the display profile:

   portal-server-install-root/SUNWps/web-src/docs/locale/desktop

   Or, you can deploy them in a custom web application archive. See the web server documentation for information on how to deploy a web application archive.

3. Run the *portal-server-install-root*/SUNWps/bin/deploy redeploy -deploy_admin_password *password* command to deploy the file.

4. Define the helpURL value for that file in the display profile.

   To define the helpURL value for a new online help file, use the format described in the section "Overview of the Desktop End-User Online Help" on page 191.

5. Use the dpadmin command to load the display profile into LDAP.

6. Verify that the new help file is displayed correctly.

To create a new online help file and to define the helpURL value in the Identity Server software administration console, do the following:

1. Create an HTML file for the provider you want to document.

2. Place your file in the appropriate directory.

   You can place your custom help files under the web server document root, in the directory specified as root by the display profile:

   portal-server-install-root/SUNWps/web-src/docs/locale/desktop

   Or, you can deploy them in a custom web application archive. See the web server documentation for information on how to deploy a web application archive.

3. Run the *portal-server-install-root*/SUNWps/bin/deploy redeploy -deploy_admin_password *password* command to deploy the file.

4. Login to the Identity Server software administration console as the administrator.

5. Choose Services from the View pull-down menu for your organization.

6. Choose the arrow next to Portal Desktop.

7. Choose the Channel and Container Management link.

8. In the channels area, choose Edit Properties for the channel you want to edit.
9. Enter the `helpURL` Value.
10. Choose Save.
11. Verify that the new help file is displayed correctly.

Appendix A

Desktop Template Files

This appendix describes the Desktop template files. It contains the following sections:

- Overview of Desktop Templates
- Desktop Templates in the default Directory
- Desktop Templates in the sampleportal Directory

Overview of Desktop Templates

Sun Java System Portal Server uses two types of files for displaying Desktop channels and pages: JavaServer Pages™ (JSP™) and template files. JavaServer Pages are the preferred way of displaying the Desktop. This appendix describes only the Desktop templates. See Appendix C, "JavaServer Pages Reference" for information about JavaServer Pages.

The desktop default JSP and template files are installed in the `/etc/opt/SUNWps/desktop/default` directory and if you choose to install sample portal, the sample portal part of the JSP and template files are installed the `/etc/opt/SUNWps/desktop/sampleportal` directory. For more information on these two directories, see Chapter 1, "Introduction to Customizing the Desktop."

The Desktop Type attribute in the Desktop attributes page of the Sun Java System Identity Server software administration console specifies from what subdirectory to retrieve either the JSP or template files for the Desktop. The default for this attribute is `sampleportal`, meaning the `sampleportal` subdirectory.

For more information on the desktop type attribute, see "Changing the Desktop Type" on page 86.

Desktop Templates in the default Directory

The templates described in this appendix are in the following provider subdirectories, beneath the `/etc/opt/SUNWps/desktop/default` directory:

- AddressBookProvider
- AppProvider
- BookmarkProvider
- CalendarProvider
- error
- LoginProvider
- LotusNotesAddressBookProvider, LotusNotesCalendarProvider, and LotusNotesMailProvider
- MailCheckProvider
- MailProvider
- MSExchangeAddressBookProvider, MSExchangeCalendarProvider, and MSExchangeMailProvider
- TemplateTabContainerProvider and TemplateTableContainerProvider
- UserInfoProvider
- default directory template files

AddressBookProvider

Table A-1 lists the templates in the `html` subdirectory of the AddressBookProvider and includes a brief description of the template file. In this two columned table, the left column contains the template file name and the right column includes a brief description of the corresponding template file.

Table A-1 AddressBookProvider Template Files

`display-clientURL.template`	Used for displaying the “Launch Address Book” link
`display-entries.template`	Used for formatting the table headers
`display-entry.template`	Used for formatting the display of the address book entry
`display-error.template`	Used for displaying error messages
`display-summary.template`	Used for formatting number of total and unread messages

Table A-1 AddressBookProvider Template Files *(Continued)*

`display.template`	Used for overall channel formatting
`edit-checkbox.template`	Used for creating edit page checkboxes
`edit-end.template`	Used for creating end of the edit page
`edit-link.template`	Used for creating application helper edit link
`edit-password.template`	Used for creating edit page password boxes
`edit-select.template`	Used for creating edit page select boxes
`edit-selectoption.template`	Used for creating edit page select box options
`edit-start.template`	Used for creating the start of the edit page
`edit-string.template`	Used for creating edit page text boxes
`edit.template`	
`ma-edit-link.template`	
`ma-edit.template`	

AppProvider

Table A-2 lists the templates in the `AppProvider` subdirectory and includes a brief description of the template file. In this two columned table, the left column contains the template file name and the right column includes a brief description of the corresponding template file.

Table A-2 AppProvider Template Files

`display.template`	Contains the JavaScript code that launches the windows that the HTML applications show up in.

BookmarkProvider

Table A-3 on page 202 lists the templates in the `html` subdirectory of BookmarkProvider and includes a brief description of the template file. In this two columned table, the left column contains the template file name and the right column includes a brief description of the corresponding template file.

Table A-3 BookmarkProvider Template Files

`display.template`	Contains JavaScript code for the Bookmark provider to open new windows with the URLs typed in and do the correct URL `http://` prepending. Also contains a small bit of formatting for the URL entry box.
`edit.template`	Contains the formatting for the Edit page for the Bookmark provider.
`editUrlWrapper.template`	Used by the edit page of Bookmark Provider to draw the part where the bookmarks are removed.
`editWindowOption.template`	Contains the markup for the radio buttons used to select the window option.
`urlWrappter.template`	Contains markup for each URL shown in `display.template`.

CalendarProvider

Table A-4 lists the templates in the `html` subdirectory of the CalendarProvider and includes a brief description of the template file. In this two columned table, the left column contains the template file name and the right column includes a brief description of the corresponding template file.

Table A-4 CalendarProvider Template Files

`display-clientURL.template`	Used for displaying the Launch Calendar link.
`display-dayView-emptyEventList.template`	Used for displaying the message “No events scheduled for today.”
`display-dayView-emptyTaskList.template`	Used for displaying the message “No tasks are pending for today.”
`display-dayView-event.template`	Used for formatting events
`display-dayView-eventAllDay.template`	Used for formatting an all day event
`display-dayView-otherTasks.template`	Used for formatting other tasks
`display-dayView-overdueTasks.template`	Used for formatting overdue tasks
`display-dayView-task.template`	Used for formatting a “normal” task

Table A-4 CalendarProvider Template Files *(Continued)*

`display-dayView.template`	Used for formatting the layout of all tasks and events
`display-error.template`	Used for displaying error messages
`display-monthView-dayOfWeek.template`	Used for creating week layout within the month
`display-monthView-emptyEventList.template`	Used for formatting when there are no events
`display-monthView-emptyTaskList.template`	Used for formatting when there are no tasks
`display-monthView-event.template`	Used for formatting an event
`display-monthView-eventAllDay.template`	Used for formatting an all day event
`display-monthView-task.template`	Used for formatting a task
`display-monthView-weekView.template`	Used for formatting the week view with a month
`display-monthView.template`	Used for generating the entire month layout
`display-summary-events.template`	Used to show number of events
`display-summary-tasks.template`	Used to show number of tasks
`display-summary.template`	Used for formatting number of total and unread messages
`display-weekView-currentDayHeader.template`	Used for formatting header for week view
`display-weekView-emptyEventList.template`	Used for formatting an empty event list
`display-weekView-emptyTaskList.template`	Used for formatting an empty task list
`display-weekView-event.template`	Used for formatting an event
`display-weekView-eventAllDay.template`	Used for formatting an all day event
`display-weekView-task.template`	Used for formatting a task

Table A-4 CalendarProvider Template Files *(Continued)*

`display-weekView.template`	Used for the overall week view
`display.template`	Used for overall channel formatting
`edit-checkbox.template`	Used for creating edit page checkboxes
`edit-config-options.template`	
`edit-end.template`	Used for creating end of the edit page
`edit-link.template`	Used for creating application helper edit link
`edit-password.template`	Used for creating edit page password boxes
`edit-select.template`	Used for creating edit page select boxes
`edit-selectoption.template`	Used for creating edit page select box options
`edit-separate.template`	
`edit-start.template`	Used for creating the start of the edit page
`edit-string.template`	Used for creating edit page text boxes
`edit.template`	
`url.template`	Used for creating hyperlinks

error

Table A-5 lists the templates in the `error` subdirectory and includes a brief description of the template file. In this two columned table, the left column contains the template file name and the right column includes a brief description of the corresponding template file.

Table A-5 error Template Files

`banner.template`	The banner across the top of the Desktop pages.
`banner_nocontext.template`	
`error.template`	
`error_nocontext.template`	Displayed when no context is available.
`noneditablechannel.template`	Template that will be used when there is an error in desktop when an edit page for a channel which is not editable is accessed. Displayed only when the user attempts to edit a channel which cannot be edited.

Table A-5 error Template Files *(Continued)*

noprivilege.template	Template that will be used when there is an error in desktop when a user with no privilege to access the desktop is trying to access the desktop. Displayed when a user who doesn't have the privilege to see the desktop attempts to access the desktop.
unknownchannel.template	Template that will be used when there is an error in desktop when an undefined channel is being accessed. Displayed when the user is trying to access a channel which is not defined in the system.

LoginProvider

Table A-6 lists the templates in the LoginProvider subdirectory and includes a brief description of the template file. In this two columned table, the left column contains the template file name and the right column includes a brief description of the corresponding template file.

Table A-6 LoginProvider Template Files

display.template	Contains the JavaScript code that launches the login window.
display_AuthLDAP.template	Contains the JavaScript code that launches the LDAP login window.
display_AuthUnix.template	Contains the JavaScript code that launches the UNIX login window.
libertyLogin.template	
persistentCookie.template	Partial HTML template for remembering the user's name and password.

LotusNotesAddressBookProvider

Table A-7 lists the templates in the LotusNotesAddressBookProvider subdirectory and includes a brief description of the template file. In this two columned table, the left column contains the template file name and the right column includes a brief description of the corresponding template file.

Table A-7 LotusNotesAddressBookProvider Template Files

display-clientURL.template	Used for displaying the "Launch Address Book" link

Table A-7 LotusNotesAddressBookProvider Template Files *(Continued)*

`display-entries.template`	Used for formatting the table headers
`display-entry.template`	Used for formatting the display of the address book entry
`display-error.template`	Used for displaying error messages
`display-summary.template`	Used for formatting number of total and unread messages
`display.template`	Used for overall channel formatting
`edit-checkbox.template`	Used for creating edit page checkboxes
`edit-end.template`	Used for creating end of the edit page
`edit-link.template`	Used for creating application helper edit link
`edit-password.template`	Used for creating edit page password boxes
`edit-select.template`	Used for creating edit page select boxes
`edit-selectoption.template`	Used for creating edit page select box options
`edit-start.template`	Used for creating the start of the edit page
`edit-string.template`	Used for creating edit page text boxes

LotusNotesCalendarProvider

Table A-8 lists the templates in the `html` subdirectory of the LotusNotesCalendarProvider and includes a brief description of the template file. In this two columned table, the left column contains the template file name and the right column includes a brief description of the corresponding template file.

Table A-8 LotusNotesCalendarProvider Template Files

`display-clientURL.template`	Used for displaying the Launch Calendar link.
`display-dayView-emptyEventList.template`	Used for displaying the message “No events scheduled for today.”
`display-dayView-emptyTaskList.template`	Used for displaying the message “No tasks are pending for today.”
`display-dayView-event.template`	Used for formatting events
`display-dayView-eventAllDay.template`	Used for formatting an all day event

Table A-8 LotusNotesCalendarProvider Template Files *(Continued)*

display-dayView-otherTasks.template	Used for formatting other tasks
display-dayView-overdueTasks.template	Used for formatting overdue tasks
display-dayView-task.template	Used for formatting a “normal” task
display-dayView.template	Used for formatting the layout of all tasks and events
display-error.template	Used for displaying error messages
display-monthView-dayOfWeek.template	Used for creating week layout within the month
display-monthView-emptyEventList.template	Used for formatting when there are no events
display-monthView-emptyTaskList.template	Used for formatting when there are no tasks
display-monthView-event.template	Used for formatting an event
display-monthView-eventAllDay.template	Used for formatting an all day event
display-monthView-task.template	Used for formatting a task
display-monthView-weekView.template	Used for formatting the week view with a month
display-monthView.template	Used for generating the entire month layout
display-summary-events.template	Used to show number of events
display-summary-tasks.template	Used to show number of tasks
display-summary.template	Used for formatting number of total and unread messages
display-weekView-currentDayHeader.template	Used for formatting header for week view
display-weekView-emptyEventList.template	Used for formatting an empty event list
display-weekView-emptyTaskList.template	Used for formatting an empty task list

Table A-8 LotusNotesCalendarProvider Template Files *(Continued)*

`display-weekView-event.template`	Used for formatting an event
`display-weekView-eventAllDay.template`	Used for formatting an all day event
`display-weekView-task.template`	Used for formatting a task
`display-weekView.template`	Used for the overall week view
`display.template`	Used for overall channel formatting
`edit-checkbox.template`	Used for creating edit page checkboxes
`edit-end.template`	Used for creating end of the edit page
`edit-link.template`	Used for creating application helper edit link
`edit-password.template`	Used for creating edit page password boxes
`edit-select.template`	Used for creating edit page select boxes
`edit-selectoption.template`	Used for creating edit page select box options
`edit-start.template`	Used for creating the start of the edit page
`edit-string.template`	Used for creating edit page text boxes
`url.template`	Used for creating hyperlinks

LotusNotesMailProvider

Table A-9 lists the templates in the `html` subdirectory of the LotusNotesMailProvider and includes a brief description of the template file. In this two columned table, the left column contains the template file name and the right column includes a brief description of the corresponding template file.

Table A-9 LotusNotesMailProvider Template Files

`display-clientURL.template`	Used for displaying the Launch Mail link
`display-error.template`	Used for displaying error messages
`display-headers-message.template`	Used for formatting individual message information
`display-headers.template`	Used for formatting message table headings
`display-summary.template`	Used for formatting number of total and unread messages

Table A-9 LotusNotesMailProvider Template Files *(Continued)*

`display.template`	Used for overall channel formatting
`edit-checkbox.template`	Used for creating edit page checkboxes
`edit-end.template`	Used for creating end of the edit page
`edit-link.template`	Used for creating application helper edit link
`edit-password.template`	Used for creating edit page password boxes
`edit-select.template`	Used for creating edit page select boxes
`edit-selectoption.template`	Used for creating edit page select box options
`edit-start.template`	Used for creating the start of the edit page
`edit-string.template`	Used for creating edit page text boxes
`url.template`	Used for creating hyperlinks

MailCheckProvider

Table A-10 lists the templates in the `MailCheckProvider` subdirectory and includes a brief description of the template file. In this two column table, the left column contains the template file name and the right column includes a brief description of the corresponding template file.

Table A-10 MailCheckProvider Template Files

`display.template`	Contains MailCheck provider layout content.
`edit.template`	Contains the formatting for the Edit page for the MailCheck provider.

MailProvider

Table A-11 lists the templates in the `html` subdirectory of the MailProvider and includes a brief description of the template file. In this two column table, the left column contains the template file name and the right column includes a brief description of the corresponding template file.

Table A-11 MailProvider Template Files

`display-clientURL.template`	Used for displaying the Launch Mail link
`display-error.template`	Used for displaying error messages

Table A-11 MailProvider Template Files *(Continued)*

`display-headers-message.template`	Used for formatting individual message information
`display-headers.template`	Used for formatting message table headings
`display-summary.template`	Used for formatting number of total and unread messages
`display.template`	Used for overall channel formatting
`edit-checkbox.template`	Used for creating edit page checkboxes
`edit-end.template`	Used for creating end of the edit page
`edit-link.template`	Used for creating application helper edit link
`edit-password.template`	Used for creating edit page password boxes
`edit-select.template`	Used for creating edit page select boxes
`edit-selectoption.template`	Used for creating edit page select box options
`edit-start.template`	Used for creating the start of the edit page
`edit-string.template`	Used for creating edit page text boxes
`url.template`	Used for creating hyperlinks

MSExchangeAddressBookProvider

Table A-12 lists the templates in the `html` subdirectory of MSExchangeAddressBookProvider and includes a brief description of the template file. In this two column table, the left column contains the template file name and the right column includes a brief description of the corresponding template file.

Table A-12 MSExchangeAddressBookProvider Template Files

`display-clientURL.template`	Used for displaying the “Launch Address Book” link
`display-entries.template`	Used for formatting the table headers
`display-entry.template`	Used for formatting the display of the address book entry
`display-error.template`	Used for displaying error messages
`display-summary.template`	Used for formatting number of total and unread messages
`display.template`	Used for overall channel formatting
`edit-checkbox.template`	Used for creating edit page checkboxes
`edit-end.template`	Used for creating end of the edit page

Table A-12 MSExchangeAddressBookProvider Template Files *(Continued)*

`edit-link.template`	Used for creating application helper edit link
`edit-password.template`	Used for creating edit page password boxes
`edit-select.template`	Used for creating edit page select boxes
`edit-selectoption.template`	Used for creating edit page select box options
`edit-start.template`	Used for creating the start of the edit page
`edit-string.template`	Used for creating edit page text boxes

MSExchangeCalendarProvider

Table A-13 lists the templates in the `html` subdirectory of the MSExchangeCalendarProvider and includes a brief description of the template file. In this two column table, the left column contains the template file name and the right column includes a brief description of the corresponding template file.

Table A-13 MSExchangeCalendarProvider Template Files

`display-clientURL.template`	Used for displaying the Launch Calendar link.
`display-dayView-emptyEventList.template`	Used for displaying the message “No events scheduled for today.”
`display-dayView-emptyTaskList.template`	Used for displaying the message “No tasks are pending for today.”
`display-dayView-event.template`	Used for formatting events
`display-dayView-eventAllDay.template`	Used for formatting an all day event
`display-dayView-otherTasks.template`	Used for formatting other tasks
`display-dayView-overdueTasks.template`	Used for formatting overdue tasks
`display-dayView-task.template`	Used for formatting a “normal” task
`display-dayView.template`	Used for formatting the layout of all tasks and events
`display-error.template`	Used for displaying error messages
`display-monthView-dayOfWeek.template`	Used for creating week layout within the month

Table A-13 MSExchangeCalendarProvider Template Files *(Continued)*

display-monthView-emptyEventList.template	Used for formatting when there are no events
display-monthView-emptyTaskList.template	Used for formatting when there are no tasks
display-monthView-event.template	Used for formatting an event
display-monthView-eventAllDay.template	Used for formatting an all day event
display-monthView-task.template	Used for formatting a task
display-monthView-weekView.template	Used for formatting the week view with a month
display-monthView.template	Used for generating the entire month layout
display-summary-events.template	Used to show number of events
display-summary-tasks.template	Used to show number of tasks
display-summary.template	Used for formatting number of total and unread messages
display-weekView-currentDayHeader.template	Used for formatting header for week view
display-weekView-emptyEventList.template	Used for formatting an empty event list
display-weekView-emptyTaskList.template	Used for formatting an empty task list
display-weekView-event.template	Used for formatting an event
display-weekView-eventAllDay.template	Used for formatting an all day event
display-weekView-task.template	Used for formatting a task
display-weekView.template	Used for the overall week view
display.template	Used for overall channel formatting
edit-checkbox.template	Used for creating edit page checkboxes

Table A-13 MSExchangeCalendarProvider Template Files *(Continued)*

`edit-end.template`	Used for creating end of the edit page
`edit-link.template`	Used for creating application helper edit link
`edit-password.template`	Used for creating edit page password boxes
`edit-select.template`	Used for creating edit page select boxes
`edit-selectoption.template`	Used for creating edit page select box options
`edit-start.template`	Used for creating the start of the edit page
`edit-string.template`	Used for creating edit page text boxes
`url.template`	Used for creating hyperlinks

MSExchangeMailProvider

Table A-14 lists the templates in the `html` subdirectory of the MSExchangeMailProvider and includes a brief description of the template file. In this two column table, the left column contains the template file name and the right column includes a brief description of the corresponding template file.

Table A-14 MSExchangeMailProvider Template Files

`display-clientURL.template`	Used for displaying the Launch Mail link
`display-error.template`	Used for displaying error messages
`display-headers-message.template`	Used for formatting individual message information
`display-headers.template`	Used for formatting message table headings
`display-summary.template`	Used for formatting number of total and unread messages
`display.template`	Used for overall channel formatting
`edit-checkbox.template`	Used for creating edit page checkboxes
`edit-end.template`	Used for creating end of the edit page
`edit-link.template`	Used for creating application helper edit link
`edit-password.template`	Used for creating edit page password boxes
`edit-select.template`	Used for creating edit page select boxes
`edit-selectoption.template`	Used for creating edit page select box options
`edit-start.template`	Used for creating the start of the edit page

Table A-14 MSExchangeMailProvider Template Files *(Continued)*

`edit-string.template`	Used for creating edit page text boxes
`url.template`	Used for creating hyperlinks

TemplateTabContainerProvider

Table A-15 lists the templates in the `TemplateTabContainerProvider` subdirectory and includes a brief description of the template file. In this two column table, the left column contains the template file name and the right column includes a brief description of the corresponding template file.

Table A-15 TemplateTabContainerProvider Template Files

`banner.template`	The banner across the top of the Desktop pages, including the Edit, Layout, and Content pages.
`display.template`	
`editForm.template`	Edit page template for creating and removing tabs.
`inlineError.template`	HTML to show an error message.
`makeNewTab.template`	HTML used in editForm.template for creating a new tab.
`menubar.template`	HTML for the menubar across the bottom of the Desktop pages, including the Edit, Layout, and Content pages, and containing Home, Help and Log Out links.
`noCache.template`	HTML META headers to prevent browser caching of the pages.
`removeRenameTab.template`	HTML used in editForm.template for removing and renaming existing tab(s).
`selectedTab.template`	HTML used to show the currently selected tab on the Desktop.
`tab.template`	HTML used to show the unselected tab(s) on the Desktop.
`tabs.template`	Template of the tab provider on the Desktop with tabs on the left.
`tabs_r.template`	Template of the tab provider on the Desktop with tabs on the right.

TemplateTableContainerProvider

Table A-16 lists the templates in the TemplateTableContainerProvider directory and includes a brief description of the template file. In this two column table, the left column contains the template file name and the right column includes a brief description of the corresponding template file.

Table A-16 TemplateTableContainer Template Files

arrangeProvider.js	JavaScript code used in the Desktop Layout page.
banner.template	The banner across the top of the Desktop pages, including the Edit, Layout, and Content pages.
bareProviderWrapper.template	Template for each provider wrapper with no titlebar.
contentBarInContent.template	Content bar for template displayed when a user selects Content on any other page's Content bar.
contentBarInLayout.template	Content bar for template displayed when user selects Layout on any other page's Content bar.
contentLayout.template	Content bar for template displayed on the Desktop before the user selects Content or Layout.
contentTemplate.template	The HTML template for the Content (Channels) page that displays when a user selects Content.
launchPopup.js	JavaScript code to launch a popup window.
layout1Template.template	left / thin, right / wide.
layout2Template.template	left / wide, right / thin.
layout3Template.template	left / thin, center / wide, right / thin.
layout4Template.template	left / thin, center / thin, right / thin.
layoutFullBottom.template	Partial HTML template for Layout pages when the user has a full width channel available at the bottom of the layout.
layoutFullTop.template	Partial HTML template for Layout pages when the user has a full width channel available at the top of the layout
maximizedTemplate.template	Template of a provider when it's in its maximized state.
menubar.template	HTML for the menubar across the bottom of the Desktop pages, including the Edit, Layout, and Content pages and containing Home, Help and Log Out links.
minimized.template	Template of a provider when it's in its minimized state - just the title / button bar showing, no content area showing.

Table A-16 TemplateTableContainer Template Files *(Continued)*

`optionsTemplate.template`	Template for the Options page of the Desktop.
`performColumnSubstitution.js`	JavaScript code used on the Layout page.
`performSubstitution.js`	JavaScript code used on the Layout page.
`popupMenubar.template`	Menubar to be used in popup windows.
`popupTemplate.template`	Used to show provider / channel content in detached windows. Similar use to providerWrapper.template, but not in a table structure.
`providerWrapper.template`	Template that all providers and channels use for layout on Desktop. Defines the look of the border of the providers and channels on the screen.
`removeProvider.js`	JavaScript code used to remove a channel from the Desktop.
`selectAll.js`	JavaScript code used on the Layout page.
`switchColumns.js`	JavaScript code used on the Layout page.
`userTemplate.template`	The base Desktop layout structure document. Very little in this file, as most of the content of the Desktop is swapped in during processing in the servlets.

UserInfoProvider

Table A-17 lists the templates in the `html` subdirectory of UserInfoProvider and includes a brief description of the template file. In this two column table, the left column contains the template file name and the right column includes a brief description of the corresponding template file.

Table A-17 UserInfoProvider Template Files

`content.template`	Content page for the UserInfo provider.
`edit.template`	Edit page for the UserInfo provider.
`netmailSettings.template`	Partial template inserted into `edit.template` for Mail information.
`passwordHandler-Membership.template`	Partial template inserted into `edit.template` if Membership Authentication is used, to allow the user to change their password.

default

Table A-18 on page 217 lists the templates in the `default` directory and includes a brief description of the template file. In this two column table, the left column contains the template file name and the right column includes a brief description of the corresponding template file.

Table A-18 Template Files in the default Directory

`AtachCommand.template`	Handles reattaching a detached channel.
`banner.template`	The banner across the top of the Desktop pages
`contentLayout.template`	Content bar for template displayed on the Desktop before the user selects Content or Layout.
`detachCommand.template`	Handles detaching a channel.
`detachEditCommand.template`	Handles link to the Edit page for the detached channel.
`detachRemoveCommand.template`	Handles closing or removing a detached channel.
`editCommand.template`	Handles link to the Edit page for this channel.
`helpHref.template`	Generates the help URL for each of the channels. Displays the help contents in a new window.
`inlineError.template`	
`MaximizeCommand.template`	Allows the channel to be displayed in the maximize mode so that the channel occupies the entire Desktop.
`menubar.template`	HTML for the menubar across the bottom of the Desktop pages.
`minimizeCommand.template`	Allows the channel to be displayed in the minimize mode so that only the title bar of the channel is displayed and no content of the channel is displayed.
`minMaximizeCommand.template`	Handles minimizing and maximizing a channel.
`normalizeCommand.template`	Allows the channel to be displayed in the normal mode so that the channel is displayed in the Desktop, with all other channels in the same table container.
`providerCommands.template`	Commands available in title bar.
`redirect.template`	
`removeCommand.template`	Removes the channel.
`bulletColor.js`	JavaScript code used to select and display bullet color.

Table A-18 Template Files in the default Directory *(Continued)*

`isPageCompletelyLoaded.js`	
`openURLInParent.js`	Javascript to open a URL in the parent window. Used in popup windows.
`pageLoaded.js`	
`toolbarRollovers.js`	JavaScript code use to display selection of Content or Layout by color change.

Desktop Templates in the sampleportal Directory

The templates described in this appendix are in the following provider subdirectories, beneath the `/etc/opt/SUNWps/desktop/sampleportal` directory:

- MyFrontPageTemplatePanelContainer
- PredefinedFrontPageTemplatePanelContainerProvider and PredefinedSamplesTemplatePanelContainerProvider
- SamplesTemplatePanelContainer and ToolsTemplatePanelContainer
- TemplateTabContainerProvider, TemplateTabCustomTableContainerProvider, and TemplateTableContainer
- sampleportal

MyFrontPageTemplatePanelContainer

Table A-19 lists the templates in the `MyFrontPageTemplatePanelContainer` directory and includes a brief description of the template file. In this two column table, the left column contains the template file name and the right column includes a brief description of the corresponding template file.

Table A-19 MyFrontPageTemplatePanelContainer Template Files

`banner.template`	The banner across the top of the Desktop pages.
`menubar.template`	HTML for the menubar across the bottom of the Desktop pages, including the Edit, Layout, and Content pages, and containing Home, Help and Log Out links.

PredefinedFrontPageTemplatePanelContainerProvider

Table A-20 lists the templates in the `PredefinedFrontPageTemplatePanelContainerProvider` directory and includes a brief description of the template file. In this two column table, the left column contains the template file name and the right column includes a brief description of the corresponding template file.

Table A-20 PredefinedFrontPageTemplatePanelContainerProvider Template Files

`banner.template`	The banner across the top of the Desktop pages.
`bareProviderWrapper.template`	Template for each provider wrapper with no titlebar.
`contentBarInContent.template`	Content bar for template displayed when a user selects Content on any other page's Content bar.
`contentBarInLayout.template`	Content bar for template displayed when user selects Layout on any other page's Content bar.
`contentLayout.template`	Content bar for template displayed on the Desktop before the user selects Content or Layout.
`contentTemplate.template`	The HTML template for the Content (Channels) page that displays when a user selects Content.
`layout1Template.template`	left / thin, right / wide.
`layout2Template.template`	left / wide, right / thin.
`layout3Template.template`	left / thin, center / wide, right / thin.
`layout4Template.template`	left / thin, center / thin, right / thin.
`layoutFullBottom.template`	Partial HTML template for Layout pages when the user has a full width channel available at the bottom of the layout.
`layoutFullTop.template`	Partial HTML template for Layout pages when the user has a full width channel available at the top of the layout
`maximizedTemplate.template`	Template of a provider when it's in its maximized state.
`menubar.template`	HTML for the menubar across the bottom of the Desktop pages, including the Edit, Layout, and Content pages and containing Home, Help and Log Out links.
`minimized.template`	Template of a provider when it's in its minimized state - just the title / button bar showing, no content area showing.
`optionsTemplate.template`	Template for the Options page of the Desktop.
`popupMenubar.template`	Menubar to be used in popup windows.

Table A-20 PredefinedFrontPageTemplatePanelContainerProvider Template Files *(Continued)*

popupTemplate.template	Used to show provider / channel content in detached windows. Similar use to providerWrapper.template, but not in a table structure.
providerWrapper.template	Template that all providers and channels use for layout on Desktop. Defines the look of the border of the providers and channels on the screen.
userTemplate.template	The base Desktop layout structure document. Very little in this file, as most of the content of the Desktop is swapped in during processing in the servlets.
arrangeProvider.js	JavaScript code used in the Desktop Layout page.
launchPopup.js	JavaScript code to launch a popup window.
performColumnSubstitution.js	JavaScript code used on the Layout page.
performSubstitution.js	JavaScript code used on the Layout page.
removeProvider.js	JavaScript code used to remove a channel from the Desktop.
selectAll.js	JavaScript code used on the Layout page.
switchColumns.js	JavaScript code used on the Layout page.

PredefinedSamplesTemplatePanelContainerProvider

Table A-21 lists the templates in the PredefinedFrontPageTemplatePanelContainerProvider directory and includes a brief description of the template file. In this two column table, the left column contains the template file name and the right column includes a brief description of the corresponding template file.

Table A-21 PredefinedSamplesTemplatePanelContainerProvider Template Files

banner.template	The banner across the top of the Desktop pages.
bareProviderWrapper.template	Template for each provider wrapper with no titlebar.
contentBarInContent.template	Content bar for template displayed when a user selects Content on any other page's Content bar.
contentBarInLayout.template	Content bar for template displayed when user selects Layout on any other page's Content bar.
contentLayout.template	Content bar for template displayed on the Desktop before the user selects Content or Layout.

Table A-21 PredefinedSamplesTemplatePanelContainerProvider Template Files *(Continued)*

contentTemplate.template	The HTML template for the Content (Channels) page that displays when a user selects Content.
layout1Template.template	left / thin, right / wide.
layout2Template.template	left / wide, right / thin.
layout3Template.template	left / thin, center / wide, right / thin.
layout4Template.template	left / thin, center / thin, right / thin.
layoutFullBottom.template	Partial HTML template for Layout pages when the user has a full width channel available at the bottom of the layout.
layoutFullTop.template	Partial HTML template for Layout pages when the user has a full width channel available at the top of the layout
maximizedTemplate.template	Template of a provider when it's in its maximized state.
menubar.template	HTML for the menubar across the bottom of the Desktop pages, including the Edit, Layout, and Content pages, and containing Home, Help and Log Out links.
minimized.template	Template of a provider when it's in its minimized state - just the title / button bar are displayed and no content area is displayed.
optionsTemplate.template	Template for the Options page of the Desktop.
popupMenubar.template	Menubar to be used in popup windows.
popupTemplate.template	Used to show provider / channel content in detached windows. Similar use to providerWrapper.template, but not in a table structure.
providerWrapper.template	Template that all providers and channels use for layout on Desktop. Defines the look of the border of the providers and channels on the screen.
userTemplate.template	The base Desktop layout structure document. Very little in this file, as most of the content of the Desktop is swapped in during processing in the servlets.
launchPopup.js	JavaScript code to launch a popup window.
performColumnSubstitution.js	JavaScript code used on the Layout page.
performSubstitution.js	JavaScript code used on the Layout page.
removeProvider.js	JavaScript code used to remove a channel from the Desktop.

Table A-21 PredefinedSamplesTemplatePanelContainerProvider Template Files *(Continued)*

`selectAll.js`	JavaScript code used on the Layout page.
`switchColumns.js`	JavaScript code used on the Layout page.

SamplesTemplatePanelContainer

Table A-22 lists the templates in the `SamplesTemplatePanelContainer` directory and includes a brief description of the template file. In this two column table, the left column contains the template file name and the right column includes a brief description of the corresponding template file.

Table A-22 SamplesTemplatePanelContainer Template Files

`banner.template`	The banner across the top of the Desktop pages.
`menubar.template`	HTML for the menubar across the bottom of the Desktop pages, including the Edit, Layout, and Content pages, and containing Home, Help and Log Out links.

TemplateTabContainerProvider

Table A-23 lists the templates in the `TemplateTabContainerProvider` directory and includes a brief description of the template file. In this two column table, the left column contains the template file name and the right column includes a brief description of the corresponding template file.

Table A-23 TemplateTabContainerProvider Template Files

`banner.template`	The banner across the top of the Desktop pages.
`menubar.template`	HTML for the menubar across the bottom of the Desktop pages, including the Edit, Layout, and Content pages, and containing Home, Help and Log Out links.

TemplateTabCustomTableContainerProvider

Table A-24 lists the templates in the `TemplateTabCustomTableContainerProvider` directory and includes a brief description of the template file. In this two column table, the left column contains the template file name and the right column includes a brief description of the corresponding template file.

Table A-24 TemplateTabCustomTableContainerProvider Template Files

`banner.template`	The banner across the top of the Desktop pages.
`menubar.template`	HTML for the menubar across the bottom of the Desktop pages, including the Edit, Layout, and Content pages, and containing Home, Help and Log Out links.

TemplateTableContainer

Table A-25 lists the templates in the `TemplateTableContainer` directory and includes a brief description of the template file. In this two column table, the left column contains the template file name and the right column includes a brief description of the corresponding template file.

Table A-25 TemplateTableContainer Template Files

`banner.template`	The banner across the top of the Desktop pages.
`menubar.template`	HTML for the menubar across the bottom of the Desktop pages, including the Edit, Layout, and Content pages, and containing Home, Help and Log Out links.

ToolsTemplatePanelContainer

Table A-26 lists the templates in the `ToolsTemplatePanelContainer` directory and includes a brief description of the template file. In this two column table, the left column contains the template file name and the right column includes a brief description of the corresponding template file.

Table A-26 ToolsTemplatePanelContainer Template Files

`banner.template`	The banner across the top of the Desktop pages.
`menubar.template`	HTML for the menubar across the bottom of the Desktop pages, including the Edit, Layout, and Content pages, and containing Home, Help and Log Out links.

sampleportal

Table A-27 lists the templates in the sampleportal directory and includes a brief description of the template file. In this two column table, the left column contains the template file name and the right column includes a brief description of the corresponding template file.

Table A-27 Template Files in the sampleportal Directory

AtachCommand.template	Handles reattaching a detached channel.
MaximizeCommand.template	Allows the channel to be displayed in the maximize mode so that the channel occupies the entire Desktop.
detachEditCommand.template	Handles link to the Edit page for the detached channel.
detachCommand.template	Handles detaching a channel.
detachRemoveCommand.template	Handles closing or removing a detached channel.
editCommand.template	Handles link to the Edit page for the detached channel.
helpHref.template	Generates the help URL for each of the channels. Displays the help contents in a new window.
minimizeCommand.template	Allows the channel to be displayed in the minimize mode so that only the title bar of the channel is displayed and no content of the channel is displayed.
normalizeCommand.template	Allows the channel to be displayed in the normal mode so that the channel is displayed in the Desktop, with all other channels in the same table container.
removeCommand.template	Removes the channel.

Miscellaneous Template Information

This section contains miscellaneous information on using templates when customizing the Desktop.

Dynamic Template Reloading

If you make changes to the Desktop templates, note that these templates are dynamically reloaded. The reload interval is by default set to thirty seconds. You can change the reload interval in the `desktopconfig.properties` file at `/etc/opt/SUNWps/desktop` directory.

➤ **To Change the Template Reload Interval**

1. Log in to the Portal Server host and change directories to `/etc/opt/SUNWps/desktop`.
2. Open the `desktopconfig.properties` file and reset the `templateScanInterval` property value.
3. Save and close the file.

Appendix B

Desktop Tag Reference

This appendix serves as a reference to the tags found in the templates discussed in Appendix A. It contains the following sections:

- Overview of the Tags
- Provider-Specific Tags
- Common Tags

Overview of the Tags

Sun Java System Portal Server uses two types of tags—JSP tags and template tags. JSP tags are used in the JavaServer Pages™ (JSPs™)—see Appendix D for more information—and template tags are used in the HTML pages of the Desktop. This appendix outlines the tags used in the HTML templates of the Desktop.

How the Desktop Template Tags Work

A template produces a channel, page, or table. The tags in a template are swapped with for real values at runtime. For example, the `[tag:netmailSettings]` tag swaps in a partial HTML template that has mail server information. The `[tag:fontFace]` tag swaps in the value of the font that the user has chosen on the Theme page. The `[tag:switchColumns]` tag swaps in the `switchColumns.js` template file, a JavaScript™ template, that lets users change how the columns are displayed on their Desktops.

Kinds of Tags Used in the Desktop

The following is a two column table: column one lists the tag and column two provides a brief description of the corresponding tag.

[`url:`*`url`*]	`url` will be encoded at run time with the `ProviderContext.encodeURL()` method.
[`surl:`*`url`*]	`url` will be pre-appended by the static root at run time. For example, in a Web Server instance, `[surl:/desktop/imagesnothing.gif]->/var/opt/SUNWps/https-`*`servername`*`/portal/web-apps/desktop/images/nothing.gif`
[*dturl*]	`dturl` will be replaced by the desktop URL at run time. For example, `[dturl]?action=logout->http://`*`server`*`:`*`port`*`/portlet/dt?action=logout`

Provider-Specific Tags

This section describes the tags specific to each provider. It contains the following:

- AddressBookProvider
- AppProvider
- BookmarkProvider
- CalendarProvider
- LoginProvider
- MailCheckProvider
- MailProvider
- TemplateTabContainerProvider
- TemplateTableContainerProvider
- UserInfoProvider

AddressBookProvider

The tags described below are used by AddressBookProvider and providers who extend this provider (such as LotusNotesAddressBookProvider and MSExchangeAddressBookProvider).

Table B-1 Tags specific to AddressBookProvider

[tag:ab-display-clientURL-appURL]	Used to display the application launch link
[tag:ab-display-entry-list]	Used as a placeholder to put in all of the address book entries
[tag:ab-display-entry-firstname]	Displays first name
[tag:ab-display-entry-lastname]	Displays last name
[tag:ab-display-entry-commonname]	Displays common name
[tag:ab-display-entry-email]	Displays email
[tag:ab-display-entry-email-link]	Displays email as a link
[tag:ab-display-error]	Displays error message
[tag:ab-display-summary-entries]	Displays a summary of the entries
[tag:ab-display-summary]	Displays summary
[tag:ab-display-entries]	Displays entries
[tag:ab-display-clientURL]	

AppProvider

The tags described below are used by AppProvider.

Table B-2 Tags specific to AppProvider

[tag:apps]	Replaced with the application content

BookmarkProvider

The tags described below are used by BookmarkProvider.

Table B-3 Tags specific to BookmarkProvider

[tag:windowOption]	JavaScript variable default for launching Bookmark windows (taken from preferences)
[tag:bookmarks]	List of bookmark links
[tag:resourceCount]	Number of bookmarks
[tag:resourceName]	Name of the bookmark
[tag:resourceURL]	URL of the bookmark
[tag:resourceList]	Checkable list of bookmarks for Edit page in table format
[tag:windowOptions]	Default for checkboxes of how the bookmark should be opened (new window, existing window, and so on)
[tag:index]	Used to reference the URLs in the edit page
[tag:targetName]	Name of Bookmark used on edit page
[tag:targetValue]	URL for bookmark used on edit page
[tag:all_new_checked]	the value for this is either CHECKED or “” based on the window preference specified in edit page
[tag:ownWindow]	localized string from resource bundle displaying the text for window options on the edit page.
[tag:one_new_checked]	the value for this is either CHECKED or “” based on the window preference selected in edit page
[tag:singleWindow]	localized string from resource bundle displaying the text for window options on the edit page.
[tag:same_checked]	the value for this is either CHECKED or “” based on the window preference selected in edit page
[tag:mainWindow]	localized string from resource bundle displaying the text for window options on the edit page.
[tag:link]	URL for bookmark used for constructing the channel content.
[tag:name]	Name of bookmark used in the channel content.

CalendarProvider

The tags described below are used by CalendarProvider and providers who extend this provider (such as LotusNotesCalendarProvider and MSExchangeCalendarProvider).

Table B-4 Tags Specific to CalendarProvider

[tag:calendar-display-client-uri]	Used to display the application launch link
[tag:calendar-display-dayView-event-startHourOfDay0]	start hour for 0 based 24-hour clock
[tag:calendar-display-dayView-event-startHourOfDay1]	start hour for 1 based 24-hour clock
[tag:calendar-display-dayView-event-startHour0]	start hour for 0 based 12-hour clock
[tag:calendar-display-monthView-event-startHour1]	start hour for 0 based 12-hour clock
[tag:calendar-display-dayView-event-endHourOfDay0]	end hour for 0 based 24-hour clock
[tag:calendar-display-dayView-event-endHourOfDay1]	end hour for 1 based 24-hour clock
[tag:calendar-display-dayView-event-endHour0]	end hour for 0 based 12-hour clock
[tag:calendar-display-dayView-event-startHour2]	start hour based on user preference of 12 or 24 hour clock format
[tag:calendar-display-dayView-task-dueHour2]	end hour based on user preference of 12 or 24 hour clock format
[tag:calendar-display-dayView-task-pendHour2]	hour based on user preference of 12 or 24 hour clock format
[tag:calendar-display-dayView-event-startMinute]	minutes of the start time

Table B-4 Tags Specific to CalendarProvider *(Continued)*

[tag:calendar-display-dayView-event-startAmPm]	am or pm identifier
[tag:calendar-display-dayView-event-endAmPm]	
[tag:calendar-display-dayView-task-dueAmPm]	
[tag:calendar-display-dayView-task-pendAmPm]	
[tag:calendar-display-dayView-event-endHour2]	end hour based on user preference of 12 or 24 hour clock format
[tag:calendar-display-dayView-task-dueMinute]	minutes of the due time
[tag:calendar-display-dayView-event-endMinute]	minutes of the end time
[tag:calendar-display-dayView-description-seperator]	description separator specified by the resource bundle 'seperatorDescription' value
[tag:calendar-display-dayView-event-description]	event description
[tag:calendar-display-dayView-event-location]	event location
[tag:calendar-display-dayView-event-summary]	event summary
[tag:calendar-display-dayView-event-allDay]	if the event is an All Day event, then it is identified by the resource bundle 'CalendarProvider-allDayEvent' value
[tag:calendar-display-dayView-task-summary]	task summary
[tag:calendar-display-dayView-task-description]	task description
[tag:calendar-display-dayView-overdueTask-description]	
[tag:calendar-display-dayView-task-dueHour1]	hour of task for 12-hour clock
[tag:calendar-display-dayView-task-dueMonth]	month the task was due

Table B-4 Tags Specific to CalendarProvider *(Continued)*

[tag:calendar-display-dayView-task-dueDay]	day of the month the task was due
[tag:calendar-display-dayView-task-dueYear]	year the task was due
[tag:calendar-display-dayView-overdueTask-summary]	overdue task summary
[tag:calendar-display-dayView-task-pendMinute]	minutes of the task
[tag:calendar-display-dayView-task-location]	task location
[tag:calendar-display-dayView-task-complete-start]	start time of completed task
[tag:calendar-display-dayView-dayOfWeek]	today's day of week
[tag:calendar-display-dayView-month]	today's month
[tag:calendar-display-dayView-day]	today's day in month
[tag:calendar-display-dayView-year]	today's year
[tag:calendar-display-dayView-taskList]	task list content
[tag:calendar-display-dayView-eventList]	event list content
[tag:calendar-display-dayView-otherTaskList]	other tasks list content
[tag:calendar-display-dayView-overdueTaskNum]	number of overdue tasks for today
[tag:calendar-display-dayView-overdueTaskList]	overdue task list content
[tag:display-dayView-dueTask-Header]	if tasks exist, then the task header is identified by the resource bundle 'dueTasks' value
[tag:display-dayView-overdueTask-Header]	if overdue tasks exist, then the overdue task header is identified by the resource bundle 'overdueTasks' value
[tag:display-dayView-dueEvent-Header]	if events exist, then the event header is identified by the resource bundle 'dueEvents' value

Table B-4 Tags Specific to CalendarProvider *(Continued)*

[tag:display-dayView-otherTask-Header]	if other tasks exist, then the other tasks header is identified by the resource bundle ‘otherTasks’ value
[tag:calendar-display-event-conflict]	if the event is in conflict, then it is identified by the resource bundle ‘conflict’ value.
[tag:calendar-display-error]	error message
[tag:calendar-display-monthView-dayOfWeek0]	day of week 0
[tag:calendar-display-monthView-dayOfWeek1]	day of week 1
[tag:calendar-display-monthView-dayOfWeek2]	day of week 2
[tag:calendar-display-monthView-dayOfWeek3]	day of week 3
[tag:calendar-display-monthView-dayOfWeek4]	day of week 4
[tag:calendar-display-monthView-dayOfWeek5]	day of week 5
[tag:calendar-display-monthView-dayOfWeek6]	day of week 6
[tag:calendar-display-monthView-event-endHour1]	end hour for 0 based 12-hour clock
[tag:calendar-display-monthView-event-startHour2]	start hour based on user preference of 12 or 24 hour clock format
[tag:calendar-display-monthView-event-startMinute]	minutes of the start time
[tag:calendar-display-monthView-event-startAmPm]	am or pm identifier
[tag:calendar-display-monthView-event-endHour2]	end hour based on user preference of 12 or 24 hour clock format
[tag:calendar-display-monthView-event-endMinute]	minutes of the end time
[tag:calendar-display-monthView-event-endAmPm]	am or pm identifier

Table B-4 Tags Specific to CalendarProvider *(Continued)*

[tag:calendar-display-monthView-event-summary]	event summary
[tag:calendar-display-monthView-event-allDay]	if the event is an All Day event, then it is identified by the resource bundle 'CalendarProvider-allDayEvent' value
[tag:calendar-display-monthView-task-pendHour2]	hour based on user preference of 12 or 24 hour clock format
[tag:calendar-display-monthView-task-pendAmPm]	am or pm identifier
[tag:calendar-display-monthView-task-pendMinute]	minutes of the task
[tag:calendar-display-monthView-task-summary]	task summary
[tag:calendar-display-monthView-day0]	day in month 0
[tag:calendar-display-monthView-day1]	day in month 1
[tag:calendar-display-monthView-day2]	day in month 2
[tag:calendar-display-monthView-day3]	day in month 3
[tag:calendar-display-monthView-day4]	day in month 4
[tag:calendar-display-monthView-day5]	day in month 5
[tag:calendar-display-monthView-day6]	day in month 6
[tag:calendar-display-monthView-eventList0]	events for day in month 0
[tag:calendar-display-monthView-taskList0]	tasks for day in month 0
[tag:calendar-display-monthView-eventList1]	events for day in month 1
[tag:calendar-display-monthView-taskList1]	tasks for day in month 1

Table B-4 Tags Specific to CalendarProvider *(Continued)*

[tag:calendar-display-monthView-eventList2]	events for day in month 2
[tag:calendar-display-monthView-taskList2]	tasks for day in month 2
[tag:calendar-display-monthView-eventList3]	events for day in month 3
[tag:calendar-display-monthView-taskList3]	tasks for day in month 3
[tag:calendar-display-monthView-eventList4]	events for day in month 4
[tag:calendar-display-monthView-taskList4]	tasks for day in month 4
[tag:calendar-display-monthView-eventList5]	events for day in month 5
[tag:calendar-display-monthView-taskList5]	tasks for day in month 5
[tag:calendar-display-monthView-eventList6]	events for day in month 6
[tag:calendar-display-monthView-taskList6]	tasks for day in month 6
[tag:calendar-display-monthView-currentDayOfWeek]	today's day of week
[tag:calendar-display-monthView-currentMonth]	today's month
[tag:calendar-display-monthView-currentDay]	today's day in month
[tag:calendar-display-monthView-currentYear]	today's year
[tag:calendar-display-monthView-dayOfWeek]	day of week content
[tag:calendar-display-monthView-weekView0]	week content for week 0
[tag:calendar-display-monthView-weekView1]	week content for week 1

Table B-4 Tags Specific to CalendarProvider *(Continued)*

[tag:calendar-display-monthView-weekView2]	week content for week 2
[tag:calendar-display-monthView-weekView3]	week content for week 3
[tag:calendar-display-monthView-weekView4]	week content for week 4
[tag:calendar-display-summary-events]	event summary information
[tag:calendar-display-summary-tasks]	task summary information
[tag:calendar-display-summary-events]	event summary
[tag:calendar-display-summary-tasks]	task summary
[tag:calendar-display-weekView-currentDayOfWeek]	today's day of week
[tag:calendar-display-weekView-currentMonth]	today's month
[tag:calendar-display-weekView-currentDay]	today's day in month
[tag:calendar-display-weekView-currentYear]	today's year
[tag:calendar-display-weekView-event-startHour1]	start hour for 0 based 12-hour clock
[tag:calendar-display-weekView-event-endHour1]	end hour for 0 based 12-hour clock
[tag:calendar-display-event-conflict]	if the event is in conflict, then it is identified by the resource bundle 'conflict' value.
[tag:calendar-display-weekView-event-startHour2]	start hour based on user preference of 12 or 24 hour clock format
[tag:calendar-display-weekView-event-startMinute]	minutes of the start time
[tag:calendar-display-weekView-event-startAmPm]	am or pm identifier
[tag:calendar-display-weekView-event-endHour2]	end hour based on user preference of 12 or 24 hour clock format

Table B-4 Tags Specific to CalendarProvider *(Continued)*

[tag:calendar-display-weekView-event-endMinute]	minutes of the end time
[tag:calendar-display-weekView-event-endAmPm]	am or pm identifier
[tag:calendar-display-weekView-event-summary]	event summary
[tag:calendar-display-weekView-event-allDay]	if the event is an All Day event, then it is identified by the resource bundle 'CalendarProvider-allDayEvent' value
[tag:calendar-display-weekView-event-summary]	event summary
[tag:calendar-display-weekView-task-pendHour2]	hour based on user preference of 12 or 24 hour clock format
[tag:calendar-display-weekView-task-pendMinute]	minutes of the end time
[tag:calendar-display-weekView-task-pendAmPm]	am or pm identifier
[tag:calendar-display-weekView-task-summary]	task summary
[tag:calendar-display-weekView-currentDayHeader]	today's date information
[tag:calendar-display-weekView-dayOfWeek0]	day of week 0
[tag:calendar-display-weekView-dayOfWeek1]	day of week 1
[tag:calendar-display-weekView-dayOfWeek2]	day of week 2
[tag:calendar-display-weekView-dayOfWeek3]	day of week 3
[tag:calendar-display-weekView-dayOfWeek4]	day of week 4
[tag:calendar-display-weekView-dayOfWeek5]	day of week 5
[tag:calendar-display-weekView-dayOfWeek6]	day of week 6

Table B-4 Tags Specific to CalendarProvider *(Continued)*

[tag:calendar-display-weekView-day0]	day in week 0
[tag:calendar-display-weekView-day1]	day in week 1
[tag:calendar-display-weekView-day2]	day in week 2
[tag:calendar-display-weekView-day3]	day in week 3
[tag:calendar-display-weekView-day4]	day in week 4
[tag:calendar-display-weekView-day5]	day in week 5
[tag:calendar-display-weekView-day6]	day in week 6
[tag:calendar-display-weekView-eventList0]	events for day in week 0
[tag:calendar-display-weekView-taskList0]	tasks for day in week 0
[tag:calendar-display-weekView-eventList1]	events for day in week 1
[tag:calendar-display-weekView-taskList1]	tasks for day in week 1
[tag:calendar-display-weekView-eventList2]	events for day in week 2
[tag:calendar-display-weekView-taskList2]	tasks for day in week 2
[tag:calendar-display-weekView-eventList3]	events for day in week 3
[tag:calendar-display-weekView-taskList3]	tasks for day in week 3
[tag:calendar-display-weekView-eventList4]	events for day in week 4
[tag:calendar-display-weekView-taskList4]	tasks for day in week 4
[tag:calendar-display-weekView-eventList5]	events for day in week 5
[tag:calendar-display-weekView-taskList5]	tasks for day in week 5
[tag:calendar-display-weekView-eventList6]	events for day in week 6

Table B-4 Tags Specific to CalendarProvider *(Continued)*

[tag:calendar-display-weekView-taskList6]	tasks for day in week 6
[tag:calendar-display-dayView-summary]	summary content
[tag:calendar-display-dayView]	day content
[tag:calendar-display-weekView]	week content
[tag:calendar-display-monthView]	month content
[tag:calendar-display-clientURL]	client application URL

LoginProvider

The tags described below are used by LoginProvider.

Table B-5 Tags Specific to LoginProvider

[tag:persistentCookie]	Inserts the persistent cookie template
[tag:libertyLogin]	Inserts the libertyLogin.Template
[tag:loginHelpUrl]	Help link for login
[tag:preLoginURL]	Inserts the liberty preLogin URL. The value is specified in the channel property preLoginURL which is typically of the form: http://www.siroe.com:80/amserver/preLogin?metaAlias=www.siroe.com&goto=http://www.siroe.com:80/portal/dt

MailCheckProvider

The tags described below are used by MailCheckProvider.

Table B-6 Tags Specific to MailCheckProvider

[tag:mailCheckErrorMessages]	Contains error messages if the user has not filled in correct values for the IMAP Server Name, IMAP User and IMAP Password.

Table B-6 Tags Specific to MailCheckProvider *(Continued)*

[tag:mailCheckContents]	Displays the result of connecting to the given server with the user ID and password. The message could be "Mail Server is alive, 0 unread message(s)."
[tag:IMAPServerName]	Name of the IMAP mail server
[tag:IMAPUserId]	user ID
[tag:IMAPPassword]	Password for the mail server

MailProvider

The tags described below are used by MailProvider and providers who extend this provider (such as LotusNotesMailProvider and MSExchangeMailProvider).

Table B-7 Tags Specific to MailProvider

[tag:mail-display-clientURL-uri]	Used to display application launch link
[tag:mail-display-error]	Display error message
[tag:mail-display-headers-message-subject]	mail subject
[tag:mail-display-headers-message-mailto]	Mail senders
[tag:mail-display-headers-message-name]	Mail sender's name
[tag:mail-display-headers-message-date]	Date message received
[tag:mail-display-headers-message-status]	Message status
[tag:mail-display-headers-subject]	Displays subject
[tag:mail-display-headers-from]	Displays From
[tag:mail-display-headers-date]	Displays Date
[tag:mail-display-headers-status]	Displays status
[tag:mail-display-summary-unread]	displays number of unread messages
[tag:mail-display-summary-total]	displays total number of messages
[tag:mail-display-summary]	Displays the summary info

Table B-7 Tags Specific to MailProvider *(Continued)*

[tag:mail-display-headers]	displays the messages headers
[tag:mail-display-clientURL]	displays the application launch URL

TemplateTabContainerProvider

The tags described below are used by TemplateTabContainerProvider and providers who extend this provider.

Table B-8 Tags Specific to TemplateTabContainerProvider

[tag:borderWidth]	Size of border (CELLPADDING) around provider HTML table
[tag:size]	Table width (WIDTH), fixed at 100%
[tag:thinProviders]	Checkbox list of available thin providers to add to desktop
[tag:wideProviders]	Checkbox list of available wide providers to add to desktop
[tag:fullProviders]	Checkbox list of available full width providers to add to desktop
[tag:leftUserProviderList]	List that shows left column channels
[tag:rightUserProviderList]	List that shows right column channels
[tag:centerUserProviderList]	List that shows center column channels
[tag:leftContent]	Channels for left column
[tag:centerContent]	Channels for center column
[tag:rightContent]	Channels for right column
[tag:leftWidth]	Percentage of total width for left column
[tag:centerWidth]	Percentage of total width for center column
[tag:rightWidth]	Percentage of total width for right column
[tag:fullBottomContent]	The content generated by the full-bottom channel
[tag:fullbottomUserProviderList]	List of Full Width (Bottom) Channels to move
[tag:fulltopUserProviderList]	List of Full Width (Top) Channels to move
[surl:/desktop/images/layout1.gif]	Icon for thin-wide layout

Table B-8 Tags Specific to TemplateTabContainerProvider *(Continued)*

[surl:/desktop/images/layout2.gif]	Icon for wide-thin layout
[surl:/desktop/images/layout3.gif]	Icon for thin-wide-thin layout
[tag:layoutOneChecked]	Makes Radio button for layout one the default.
[tag:layoutTwoChecked]	Makes Radio button for layout two the default.
[tag:layoutThreeChecked]	Makes Radio button for layout three the default.
[surl:/desktop/images/layout4.gif]	Icon for thin-thin-thin layout
[tag:layoutFourChecked]	Makes Radio button for layout four the default.
[tag:parentContainerName]	The top level container name for the TemplateTabContainer

TemplateTableContainerProvider

The tags described below are used by TemplateTableContainerProvider and providers who extend this provider (such as MyFrontPageTemplatePanelContainer, PredefinedFrontPageTemplatePanelContainerProvider, PredefinedSamplesTemplatePanelContainerProvider, SamplesTemplatePanelContainer).

Table B-9 Tags Specific to TemplateTableContainerProvider

[tag:borderWidth]	Size of border (CELLPADDING) around provider HTML table
[tag:size]	Table width (WIDTH), fixed at 100%
[tag:thinProviders]	Checkbox list of available thin providers to add to desktop
[tag:wideProviders]	Checkbox list of available wide providers to add to desktop
[tag:fullProviders]	Checkbox list of available full width providers to add to desktop
[tag:layoutFullTop]	Inserts layoutFullTop.template into this template
[tag:layoutFullBottom]	Inserts layoutFullBottom.template into this template
[tag:leftUserProviderList]	List that shows left column channels
[tag:rightUserProviderList]	List that shows right column channels

Table B-9 Tags Specific to TemplateTableContainerProvider *(Continued)*

[tag:centerUserProviderList]	List that shows center column channels
[tag:fullbottomUserProviderList]	List of Full Width (Bottom) Channels to move
[tag:fulltopUserProviderList]	List of Full Width (Top) Channels to move
[tag:layoutOneChecked]	Makes Radio button for layout one the default.
[tag:layoutTwoChecked]	Makes Radio button for layout two the default.
[tag:layoutThreeChecked]	Makes Radio button for layout three the default.
[tag:layoutFourChecked]	Makes Radio button for layout four the default.
[tag:fullTopContent]	Provider / channel content (HTML) inserted here
[tag:fullBottomContent]	The content generated by the full-bottom channel
[tag:centerContent]	Channels for right column
[tag:rightContent]	Channels for center column
[tag:leftContent]	Channels for left column
[tag:leftWidth]	Percentage of total width for left column
[tag:centerWidth]	Percentage of total width for center column
[tag:rightWidth]	Percentage of total width for right column

UserInfoProvider

The tags described below are used by UserInfoProvider.

Table B-10 Tags Specific to UserInfoProvider

[tag:greeting]	User's greeting
[tag:cn]	User's common or full name
[tag:currentDate]	Date
[tag:timeLeft]	Time left in user's session
[tag:maxIdle]	Maximum idle time
[tag:timezoneList]	List of time zones
[tag:localeList]	List of available locales

Table B-10 Tags Specific to UserInfoProvider *(Continued)*

[tag:netmailSettings]	Inserts netmailSettings.template into this page to get the following information: IMAP server name, SMTP server name, IMAP user ID and IMAP password
[tag:passwordHandler]	Inserts passwordHandler-Membership.template (if available) to change membership password
[tag:iplanet-ps-netmail-imap-server-name]	IMAP server name
[tag:iplanet-ps-netmail-smtp-server-name	SMTP server name
[tag:iplanet-ps-netmail-imap-userid]	IMAP user ID
[tag:iplanet-ps-netmail-imap-password]	IMAP password

Common Tags

Table B-11 Common Tags

[tag:fontFace]	Font chosen
[tag:fontFace1]	Default font for the Desktop (sans-serif)
[tag:iwtDesktop-fontFace1]	Default font for the Desktop (sans-serif)
[tag:desktop-fontFace1]	Default font for the Desktop (sans-serif)
[tag:fontColor]	Color of font chosen
[tag:bgColor]	Provider background color chosen
[tag:borderColor]	Color of the channel border. The border color can be changed by user in the desktop custom theme page.
[tag:titlebarColor]	Color of title bar
[tag:staticContent]	Directory you defined for the deployment URI during installation.
[tag:localeString]	Directory designation for chosen locale
[tag:productName]	Product name
[tag:providerTitle]	Provider / Channel title
[tag:title]	The title of the provider / channel

Table B-11 Common Tags *(Continued)*

[tag:selectedName]	Name of selected tab
[tag:frontContainerName]	Name of the front container
[tag:parentContainerName]	The top level container name for the TemplateTabContainer
[tag:providerName]	The channel name
[tag:channelName]	Channel name as defined in the channel display profile definition.
[tag:theme_channel]	The theme edit channel name at run time, this will be either the presetThemeContainer or the customThemeContainer.
[tag:ErrorMessage]	error content
[tag:providerContent]	Content of the provider
[tag:detachedContent]	Channel content in the detached window
[tag:content]	Content of the channel
[tag:detachedContent]	Channel content in the detached window
[tag:fullTopContent]	Provider / channel content (HTML) inserted here
[tag:name]	display value
[tag:stackTrace]	Produces a stack trace
[tag:MaximizedContent]	Inserts content in the maximize mode
[tag:minimizeText]	A text that is used as an alternate for the minimize icon
[tag:maximizeText]	A text that is used as an alternate for the maximize icon
[tag:minMaximizeText]	Alternated string for the minimize or normalize image in the title bar, this is a localized string, the state of minimize or maximize is determined at run time.
[tag:removeText]	Text for the alt tag for the remove icon
[tag:detachAttachText]	Text for the alt tag for the detach / attach image in the title bar, this is a localized string. The detach or attach mode is determined at run time.
[tag:minMaximizeIcon]	Inserts minimize / maximize icon

Table B-11 Common Tags *(Continued)*

[tag:removeTag]	Alternated string for the remove image in the title bar, this is a localized string
[tag:editTag]	Text for the alt tag for the edit icon
[tag:help_tag]	Text for alt message
[tag:bulletColor]	Inserts bulletColor.js into this template
[tag:toolbarRollover]	Inserts toolbarRollover.js into this template.
[tag:banner]	Inserts banner.template into this template
[tag:menubar]	Inserts menubar.template into template
[tag:contentBarInContent]	Inserts contentBarInContent.template into this template
[tag:contentBarInLayout]	Inserts contentBarInLayout.template into this template
[tag:arrangeProvider]	Inserts arrangeProvider.js template into this template
[tag:performSubstitution]	Inserts performSubstitution.js template
[tag:performColumnSubstitution]	Inserts performColumnSubstitution.js template
[tag:selectAll]	Inserts selectAll.js template
[tag:switchColumns]	Inserts switchColumns.js template
[tag:layoutFullTop]	Inserts layoutFullTop.template into this template
[tag:layoutFullBottom]	Inserts layoutFullBottom.template into this template
[tag:openURLInParent]	Inserts openURLInParent.js template
[tag:popupMenubar]	Inserts popupMenubar.template template
[tag:launchPopup]	Inserts launchPopup.js template
[tag:inlineError]	Replaced with inlineError.template if error has occurred
[tag:removeCommand]	Inserts the removeCommand.template
[tag:detachAtachCommand]	If the channel is detached, insert the detachCommand.template; if the channel is attached, then insert the attachCommand.template.
[tag:editCommand]	Inserts the editCommand.template

Table B-11 Common Tags *(Continued)*

[tag:helpCommand]	Inserts helpHref.template
[tag:minMaximizeCommand]	Inserts the minMaximizeCommand.template
[tag:resourceName]	Used to dynamically build edit pages. Should not be edited.
[tag:header]	
[tag:attName]	
[tag:attSelected]	
[tag:attValue]	
[tag:string]	
[tag:selected]	
[tag:options]	
[tag:process]	
[tag:isAppHandler]	
[tag:mail-display-error]	
[tag:editLink]	Used to display link for the application helper editing / URL to edit
[tag:link]	URL location
[tag:logoutUrl]	the logout URL
[tag:desktop_url]	URL of Desktop to return to
[tag:removeURL]	URL of the channel to be removed
	?action=process&provider=*thecontainername*&*thecontainernam e*.channelAction=remove&*thecontainername*.targetProvider =*providername*
[tag:url]	URL of new location
[tag:editURL]	URL of Edit page for this channel
	?action=edit&provider=*theeditcontainername*&targetprovid er=*providername*&containerName=*thecontainername*
[tag:detachAttachURL]	URL of the channel to be detached
	action=process&provider=*thecontainername*&*thecontainername* .channelAction=attach&*thecontainername*.targetProvider =*providename*

Table B-11 Common Tags *(Continued)*

[tag:maximizeURL]	A URL to show the channel in maximize mode
[tag:minMaximizeURL]	A URL to show the channel in either the minimize or the normal mode, the mode is decided at run time
	URL of channel to minimize or maximize
	?action=process&provider=*thecontainername*&*thecontainernam e*.channelAction=maxmize&*thecontainername*.targetProvide r=*providername*
	?action=process&provider=*thecontainername*&*thecontainernam e*.channelAction=minimize&*thecontainername*.targetProvid er=*providername*
[tag:s_detachImage	Run time path for the detach image
[tag:s_editImage]	Run time path for the edit image
[tag:s_removeImage]	Run time path for the remove image
[tag:s_helpImage]	Run time path for the help image
[tag:s_minimizeImage]	Run time path for the minimized or the maximized image, the state of minimize or maximize is determined at run time.
[tag:s_normalizeImage]	Run time path for the normalized image
[surl:/desktop/css/style.css]	Style sheet used by the Desktop for the banner and tabs templates
[surl:/docs/en/desktop/usedesk.htm]	Help link for Desktop
[surl:/docs/en/desktop/fdesktop.htm]	Help link for frames on the Desktop
[surl:/desktop/images/nothing.gif]	Used to space gifs or channels
[surl:/desktop/images/b_up.gif]	Icon that shows up arrow
[surl:/desktop/images/b_down.gif]	Icon that shows down arrow
[surl:/desktop/images/b_left.gif]	Icon that shows arrow pointing left
[surl:/desktop/images/b_right.gif]	Icon that shows arrow pointing right
[surl:/desktop/images/b_normal.gif]	A URL that points to the normal image
[surl:/desktop/images/b_minimize.gif]	A URL that points to the maximize image

Table B-11 Common Tags *(Continued)*

[surl:/desktop/images/b_maximize.gif]	A URL that points to the maximize image
[surl:/desktop/images/b_attach.gif]	A URL that points to the attach image
[surl:/desktop/images/layout1.gif]	Icon for thin-wide layout
[surl:/desktop/images/layout2.gif]	Icon for wide-thin layout
[surl:/desktop/images/layout3.gif]	Icon for thin-wide-thin layout
[surl:/desktop/images/layout4.gif]	Icon for thin-thin-thin layout
[surl:/images/blueBullet.gif]	Blue button to denote Home, Help, Logout, and so on
[surl:/images/redBullet.gif]	Red button to denote home, help, logout, or whatever has been chosen.
[surl:/images/spacer.gif]	Used to space gifs or channels
[surl:/images/productName.gif]	Logo gif of the product name
[surl:/images/blueBullet.gif]	Blue button to denote Home, Help, Logout, and so on
[tag:help_link]	Help link for Desktop
[tag:help_icon]	Icon for help in channel title bar
[tag:provider_cmds]	Inserts the channel command button links (maximize, detach, and so on)
[tag:serviceTimeout]	Provider Timeout in seconds
[tag:theme_channel]	The theme edit channel name at run time, this will be either the presetThemeContainer or the customThemeContainer.
[tag:detachedContent]	Channel content in the detached window

Appendix C

JavaServer Pages Reference

This appendix describes the various JavaServer Pages™ (JSP™) used by Sun Java System Portal Server software.

This appendix contains these sections:

- Anonymous Desktop JSPs
- Desktop JSPs in the default Directory
- JSPs in the sampleportal Directory

Before using this appendix, you should understand the order in which the Portal Server software looks for JSPs. Some of the JSPs that are used by a particular channel are actually located in the directory named for the provider instead of the channel. In other cases, the JSPs are located in the `/etc/opt/SUNWps/desktop/default` directory. Also, if the Desktop type, for example, is `sampleportal`, the lookup mechanism searches in the `default` directory if the file is not found in the particular `DesktopType` subdirectory.

See "JSP and Template Files Lookup Scenario" on page 31 for more information.

Anonymous Desktop JSPs

Anonymous Desktop JSPs are located under the `/etc/opt/SUNWps/desktop/anonymous/*` directory. This section includes:

- FrameTabContainer JSPs
- JSPDynamicSingleContainer JSPs
- JSPTabContainer JSPs and JSPTableContainer JSPs
- PredefinedFrontPageFramePanelContainerProvider JSPs and PredefinedFrontPageTabPanelContainerProvider JSPs

- PredefinedSamplesFramePanelContainerProvider JSPs and PredefinedSamplesTabPanelContainerProvider JSPs

FrameTabContainer JSPs

Anonymous Desktop JSPs for the FrameTabContainer are located in the `/etc/opt/SUNWps/desktop/anonymous/FrameTabContainer` directory.

The following is a two column table: column one lists the file name; column two provides a brief description.

`frametab.jsp`	This JSP is the anonymous version of the frametab.jsp for the frame tab container with the links for customization deactivated.
`header.jsp`	This JSP is the anonymous version of the `header.jsp` for the frame tab container with the links for customization deactivated.
`menubar.jsp`	This JSP is the anonymous version of the menubar.jsp for frame tab container with the links for customization deactivated.

JSPDynamicSingleContainer JSPs

Anonymous Desktop JSPs for the JSPDynamicSingleContainer are located in the `/etc/opt/SUNWps/desktop/anonymous/JSPDynamicSingleContainer` directory.

JSPTabContainer JSPs

Anonymous Desktop JSPs for the JSPTabContainer are located in the `/etc/opt/SUNWps/desktop/anonymous/JSPTabContainer` directory.

The following is a two column table: column one lists the file name; column two provides a brief description.

`header.jsp`	This JSP is the anonymous version of the `header.jsp` for the tab container with the links for customization deactivated.

menubar.jsp	This JSP is the anonymous version of the menubar.jsp for the tab container with the links for customization deactivated.

JSPTableContainer JSPs

Anonymous Desktop JSPs for the JSPTableContainer are located in the /etc/opt/SUNWps/desktop/anonymous/JSPTableContainer directory.

The following is a two column table: column one lists the file name; column two provides a brief description.

header.jsp	This JSP is the anonymous version of the header.jsp for the table container with the links for customization deactivated.
menubar.jsp	This JSP is the anonymous version of the menubar.jsp for the table container with the links for customization deactivated.
table.jsp	Displays the table container's content view, which is used by the table container inside a frame tab container. (Dynamically includes launchPopup.jsp, tablecolumn.jsp, and tabletopbottom.jsp.)
tabtable.jsp	Displays the table container's content view, which is used by the table container inside a tab container. (Dynamically includes launchPopup.jsp, tablecolumn.jsp, and tabletopbottom.jsp.)
toptable.jsp	Displays the table container's content view, which is used by the table container that is the top most container in the Desktop. (Dynamically includes launchPopup.jsp, tablecolumn.jsp, and tabletopbottom.jsp.)

PredefinedFrontPageFramePanelContainerProvider JSPs

Anonymous Desktop JSPs for the PredefinedFrontPageFramePanelContainerProvider are located in the `/etc/opt/SUNWps/desktop/anonymous/PredefinedFrontPageFramePanelContainerProvider` directory.

The following is a two column table: column one lists the file name; column two provides a brief description.

`header.jsp`	This JSP is the anonymous version of the `header.jsp` for the table container with the links for customization deactivated.
`menubar.jsp`	This JSP is the anonymous version of the `menubar.jsp` for the table container with the links for customization deactivated.
`searchbox.jsp`	Displays the search box that are used in the Desktop header area.
`table.jsp`	Displays the table container's content view, which is used by the table container inside a frame tab container. (Dynamically includes `launchPopup.jsp`, `tablecolumn.jsp`, and `tabletopbottom.jsp`.)
`tabtable.jsp`	Displays the table container's content view, which is used by the table container inside a tab container. (Dynamically includes `launchPopup.jsp`, `tablecolumn.jsp`, and `tabletopbottom.jsp`.)
`toptable.jsp`	Displays the table container's content view, which is used by the table container that is the top most container in the Desktop. (Dynamically includes `launchPopup.jsp`, `tablecolumn.jsp`, and `tabletopbottom.jsp`.)

PredefinedFrontPageTabPanelContainerProvider JSPs

Anonymous Desktop JSPs for the PredefinedFrontPageTabPanelContainerProvider are located in the `/etc/opt/SUNWps/desktop/anonymous/PredefinedFrontPageTabPanelContainerProvider` directory.

The following is a two column table: column one lists the file name; column two provides a brief description.

`header.jsp`	This JSP is the anonymous version of the `header.jsp` for the tab container with the links for customization deactivated.
`menubar.jsp`	This JSP is the anonymous version of the `menubar.jsp` for the tab container with the links for customization deactivated.
`searchbox.jsp`	Displays the search box that are used in the Desktop header area.
`table.jsp`	Displays the table container's content view, which is used by the table container inside a frame tab container. (Dynamically includes `launchPopup.jsp`, `tablecolumn.jsp`, and `tabletopbottom.jsp`.)
`tabtable.jsp`	Displays the table container's content view, which is used by the table container inside a tab container. (Dynamically includes `launchPopup.jsp`, `tablecolumn.jsp`, and `tabletopbottom.jsp`.)
`toptable.jsp`	Displays the table container's content view, which is used by the table container that is the top most container in the Desktop. (Dynamically includes `launchPopup.jsp`, `tablecolumn.jsp`, and `tabletopbottom.jsp`.)

PredefinedSamplesFramePanelContainerProvider JSPs

Anonymous Desktop JSPs for the PredefinedSamplesFramePanelContainerProvider are located in the `/etc/opt/SUNWps/desktop/anonymous/PredefinedSamplesFramePanelContainerProvider` directory.

The following is a two column table: column one lists the file name; column two provides a brief description.

`header.jsp`	This JSP is the anonymous version of the `header.jsp` for the table container with the links for customization deactivated.
`menubar.jsp`	This JSP is the anonymous version of the `menubar.jsp` for the table container with the links for customization deactivated.
`searchbox.jsp`	Displays the search box that are used in the Desktop header area.
`table.jsp`	Displays the table container's content view, which is used by the table container inside a frame tab container. (Dynamically includes `launchPopup.jsp`, `tablecolumn.jsp`, and `tabletopbottom.jsp`.)
`tabtable.jsp`	Displays the table container's content view, which is used by the table container inside a tab container. (Dynamically includes `launchPopup.jsp`, `tablecolumn.jsp`, and `tabletopbottom.jsp`.)
`toptable.jsp`	Displays the table container's content view, which is used by the table container that is the top most container in the Desktop. (Dynamically includes `launchPopup.jsp`, `tablecolumn.jsp`, and `tabletopbottom.jsp`.)

PredefinedSamplesTabPanelContainerProvider JSPs

Anonymous Desktop JSPs for the PredefinedSamplesTabPanelContainerProvider are located in the `/etc/opt/SUNWps/desktop/anonymous/PredefinedSamplesTabPanelContainerProvider` directory.

The following is a two column table: column one lists the file name; column two provides a brief description.

`header.jsp`	This JSP is the anonymous version of the `header.jsp` for the tab container with the links for customization deactivated.
`menubar.jsp`	This JSP is the anonymous version of the `menubar.jsp` for the tab container with the links for customization deactivated.
`searchbox.jsp`	Displays the search box that are used in the Desktop header area.
`table.jsp`	Displays the table container's content view, which is used by the table container inside a frame tab container. (Dynamically includes `launchPopup.jsp`, `tablecolumn.jsp`, and `tabletopbottom.jsp`.)
`tabtable.jsp`	Displays the table container's content view, which is used by the table container inside a tab container. (Dynamically includes `launchPopup.jsp`, `tablecolumn.jsp`, and `tabletopbottom.jsp`.)
`toptable.jsp`	Displays the table container's content view, which is used by the table container that is the top most container in the Desktop. (Dynamically includes `launchPopup.jsp`, `tablecolumn.jsp`, and `tabletopbottom.jsp`.)

Desktop JSPs in the default Directory

This section provides a listing with description of the JSPs in the `/etc/opt/SUNWps/desktop/default` directory. It contains:

- DiscussionLite JSPs, Discussions JSP, and DiscussionsProvider JSPs
- DummyChannel JSPs
- IMProvider JSPs
- JSPContentContainer JSPs, JSPEditContainer JSPs, and JSPLayoutContainer JSPs
- JSPDynamicSingleContainer
- JSPFrameCustomTableContainerProvider JSPs
- JSPProvider JSPs
- JSPSingleContainerProvider JSPs, JSPTabContainerProvider JSPs, JSPTabCustomTableContainerProvider JSPs, and JSPTableContainerProvider JSPs
- Miscellaneous JSPs
- SampleSimpleWebService JSPs and SampleSimpleWebServiceConfigurable JSPs
- Search JSPs and SearchProvider JSPs
- SimpleWebServiceConfigurableProvider JSPs and SimpleWebServiceProvider JSPs
- Subscriptions JSPs and SubscriptionsProvider JSPs
- TabJSPEditContainer JSPs

DiscussionLite JSPs

The DiscussionLite channel JSPs are located in `/etc/opt/SUNWps/desktop/default/DiscussionLite` directory.

The following is a two column table: column one lists the file name; column two provides a brief description.

`discussionLiteContent.jsp`	Content JSP. This JSP gets portal display profile properties and invokes search using `query.jsp` and most of the user interface is in `display.jsp`.
`display.jsp`	Displays results mostly in HTML.
`error.jsp`	Error page.

query.jsp	Sets search parameters and executes search.

Discussions JSP

The Discussions channel JSPs are located in the /etc/opt/SUNWps/desktop/default/Discussions directory.

The following is a two column table: column one lists the file name; column two provides a brief description.

declare.jsp	Declares all portal properties.
discussionContent.jsp	Content page. This JSP routes the request to: • fullDiscussion.jsp if display mode is set to full (or dmode=full) • viewDiscussion.jsp if dmode=vl • feedback.jsp if dmode-cmt • ratingProcess.jsp if dmode=rtg
discussionDoEdit.jsp	Process edit page.
discussionEdit.jsp	Edit page.
error.jsp	Error page.
feedback.jsp	Request is handled by feedback.jsp when the dmode (discussions mode) value is equal to cmt. Routes the request to feedbackForm.jsp.
feedbackDisplay.jsp	Displays the header information above the feedback form.
feedbackForm.jsp	Displays the ‘post reply’ and ‘start a new discussion’ form.
feedbackProcess.jsp	Comment submission is handled by this JSP. The JSP retrieves all the input parameters, builds the SOIF and submits the SOIF to the search database. This JSP consists of scriptlets.

`fullDiscussion.jsp`	Request is handled by `fullDiscussion.jsp` when `dmode` is `full` and routes the request to `fullDiscussionDisplay.jsp`. Controls the list view of discussions.
`fullDiscussionDisplay.jsp`	`fullDiscussionDisplay.jsp` displays the list of results requested by `fullDiscussion.jsp`.
`pageFooter.jsp`	Displays pagination on list discussions page.
`portal.jsp`	General portal Desktop page. Retrieves all portal provider properties.
`query.jsp`	Executes search. Used by all pages to execute a search.
`rating.jsp`	Displays the selection menu for ratings. Included in `viewDiscusisonDisplay.jsp` and `viewDiscussionHeader.jsp`.
`ratingProcess.jsp`	Request is handled by `ratingProcess.jsp` when `dmode=rtg`. Handles rating submission. Consists mostly of scriptlets.
`searchUI.jsp`	Displays the search box on the list discussions page.
`viewDiscussion.jsp`	Request is handled by `viewDiscussion.jsp` when `dmode=vl`. Controls the View A Discussion subtree page.
`viewDiscussionBar.jsp`	Displays the separator bar with the filter, threshold, view menus and the search discussion text field.
`viewDiscussionDisplay.jsp`	Displays the discussion subtree below the separator bar.
`viewDiscussionHeader.jsp`	Displays the detailed view of the discussion. Displayed above the separator bar.
`viewDiscussionNavigation.jsp`	Displays the Navigation links shown above and below the discussion header. Navigation links consist of links for ‘All Discussions’, ‘To parent’, ‘To Discussion, ‘Reference’, ‘Post Reply’.

DiscussionsProvider JSPs

The DiscussionsProvider JSPs are located in `/etc/opt/SUNWps/desktop/default/DiscussionsProvider` directory.

The following is a two column table: column one lists the file name; column two provides a brief description.

`declare.jsp`	Declares all portal properties.
`discussionContent.jsp`	Content page. This JSP routes the request to `full` if if display mode is set to full (or `dmode=full`), or if `vl` is set to `viewDiscussion.jsp`, or `cmt` is set to `feedback.jsp`, or if `rtg` is set to `ratingProcess.jsp`.
`discussionDoEdit.jsp`	Process edit page.
`discussionEdit.jsp`	Edit page.
`error.jsp`	Error page.
`feedback.jsp`	Request is handled by `feedback.jsp` when the `dmode` (discussions mode) value is equal to `cmt`. Routes the request to `feedbackForm.jsp`.
`feedbackDisplay.jsp`	Displays feedback.
`feedbackForm.jsp`	Displays the ‘post reply’ and ‘start a new discussion’ form.
`feedbackProcess.jsp`	Comment submission is handled by this JSP. The JSP retrieves all the input parameters, builds the SOIF and submits the SOIF to the search database. This JSP consists of scriptlets.
`fullDiscussion.jsp`	Sets search parameters, executes search, and displays results.
`fullDiscussionDisplay.jsp`	Displays main discussions with description inline based on `showDesc` property.
`pageFooter.jsp`	Displays pagination on list discussions page.
`portal.jsp`	General portal Desktop page. Retrieves all portal provider properties.
`query.jsp`	Executes search. Used by all pages to execute a search.

rating.jsp	Displays the selection menu for ratings. Included in viewDiscusisonDisplay.jsp and viewDiscussionHeader.jsp.
ratingProcess.jsp	Request is handled by ratingProcess.jsp when dmode=rtg. Handles rating submission. Consists mostly of scriptlets.
searchUI.jsp	Displays the search box on the list discussions page.
viewDiscussion.jsp	Request is handled by viewDiscussion.jsp when dmode=vl. Controls the View A Discussion subtree page.
viewDiscussionBar.jsp	Displays the separator bar with the filter, threshold, view menus and the search discussion text field.
viewDiscussionDisplay.jsp	Displays the discussion subtree below the separator bar.
viewDiscussionHeader.jsp	Displays the detailed view of the discussion. Displayed above the separator bar.
viewDiscussionNavigation.jsp	Displays the Navigation links shown above and below the discussion header. Navigation links consist of links for ‘All Discussions’, ‘To parent’, ‘To Discussion, ‘Reference’, ‘Post Reply’.

DummyChannel JSPs

The DummyChannel JSPs are located in the /etc/opt/SUNWps/desktop/default/DummyChannel directory. These JSPs are used in the situation when there is no valid default channel name specified for the Desktop. For example, when a new organization is created, the default channel name will be set to DummyChannel by default. If the system administrator does not specify a valid default channel name, then the DummyChannel will be displayed with a warning message to remind user that the default channel name needs to be configured.

The following is a two column table: column one lists the file name; column two provides a brief description.

dummy.jsp	Displays the warning message for the user.
header.jsp	The header for the Dummy Channel.

`menubar.jsp`	The menubar for the Dummy Channel.

IMProvider JSPs

The IMProvider JSPs are located in the `/etc/opt/SUNWps/desktop/default/IMProvider` directory.

The following is a two column table: column one lists the file name; column two provides a brief description.

`IMArchiveDisplay.jsp`	Controls searching through archived instant messaging content.
`IMContent.jsp`	Controls the content in the channel.
`IMEdit.jsp`	Controls the content for the edit page of the channel.
`invite.jsp`	Controls the content in the popup window that is displayed when a user is invited to a conference is an instant messaging client that is already running.
`jnlpLaunch.jsp`	Controls the messages that are in the Java Web Start window that is used to start the instant messaging client.
`pluginLaunch.jsp`	Controls the content of the popup window that is used to run the IM client.

JSPContentContainer JSPs

The JSPContentContainer JSPs are located in the `/etc/opt/SUNWps/desktop/default/JSPContentContainer` directory. These JSPs are used for the content view when the Content link is selected in a JSP-based table container.

The following is a two column table: column one lists the file name; column two provides a brief description.

`contentdoedit.jsp`	Processes the result from the Content Edit page.
`contentedit.jsp`	Displays the content Edit page.

JSPDynamicSingleContainer

The JSPDynamicSingleContainer JSPs are located in the `/etc/opt/SUNWps/desktop/default/JSPDynamicSingleContainer` directory. This container is used by the search form in the header on the Desktop front page.

The following is a two column table: column one lists the file name; column two provides a brief description.

`dynamicSingle.jsp`	Used by the DynamicSingleContainer to display the channel specified in the request parameter.

JSPEditContainer JSPs

JSPEditContainer JSPs are located in the `/etc/opt/SUNWps/desktop/default/JSPEditContainer` directory. These JSPs are used when the Edit icon is selected in a channel title bar inside a JSP- based container. Channels that have the `editType` defined as `EDIT_SUBSET` use these JSPs.

The following is a two column table: column one lists the file name; column two provides a brief description.

`doedit.jsp`	Processes the result from the Edit page.
`edit.jsp`	Displays the Edit view of a channel. Also provides a wrapper around the actual Edit view for a given channel.

JSPFrameCustomTableContainerProvider JSPs

JSPFrameCustomTableContainerProvider JSPs are located in the `/etc/opt/SUNWps/desktop/default/JSPFrameCustomTableContainerProvider` directory.

JSPFrameCustomTableContainerProvider JSPs are used when the user creates a new page from Scratch in the sections page.

The following is a two column table: column one lists the file name; column two provides a brief description.

framecustomtable.jsp	Displays the content for the newly created page (table container) from the Sections page.
launchPopup.jsp	This JSP is used to determine the channels that are in detached mode and invoke the detached windows for these channels.
leafWrapper.jsp	Displays the channel title bar and border.
popup.jsp	This JSP is used to draw the content in the detached window for the channel in the detached mode.
popupMenubar.jsp	This JSP is used to draw the menubar in the detached window for a detached channel.
providerCommands.jsp	Displays the channel command buttons and links, such as Remove, Minimize / Maximize, Edit, and Help.
providerWrapper.jsp	This JSP is used to draw the wrapper containing the attach, edit, and help links around the channel displayed in the detached window.
tablecolumn.jsp	Handles the left, center and right columns of a table. (Dynamically included.)
tablemaximized.jsp	Handles the channel in the maximized state on the Desktop front page. This JSP is used to draw the HTML around the channel displayed in the maximized state.
tabletopbottom.jsp	Handles the top and bottom channels of a table. (Dynamically included.)

JSPLayoutContainer JSPs

JSPLayoutContainer JSPs are located in the `/etc/opt/SUNWps/desktop/default/JSPLayoutContainer` directory. These JSPs are used to display the Layout view when the Layout link is selected in a JSP-based table container.

The following is a two column table: column one lists the file name; column two provides a brief description.

`layout1.jsp`	Displays the thin-wide layout view.
`layout2.jsp`	Displays the wide-thin layout view.
`layout3.jsp`	Displays the thin-wide-thin layout view.
`layoutdoedit.jsp`	Processes the result from the Layout Edit page.
`layoutedit.jsp`	Displays the Layout Edit page.
`selectLayout.jsp`	Displays the three layout images and the select radio buttons.

JSPProvider JSPs

JSPProvider JSPs are located in the `/etc/opt/SUNWps/desktop/default/JSPProvider` directory. This directory contains default set of JSPs that are used by the JSP channels.

The following is a two column table: column one lists the file name; column two provides a brief description.

`samplecontent.jsp`	Displays the contents of the JSP channels.
`sampledoedit.jsp`	Invoked when the user completes processing the Edit page of the JSP channels.
`sampleedit.jsp`	Invoked when the user clicks the Edit button of the JSP channels.

JSPSingleContainerProvider JSPs

JSPSingleContainerProvider JSPs are located in the `/etc/opt/SUNWps/desktop/default/JSPSingleContainerProvider` directory.

The following is a two column table: column one lists the file name; column two provides a brief description.

`header.jsp`	Displays the product banner that is used by the Single Container page.

`menubar.jsp`	Displays the menubar that has the Home, Theme, Help, and Logout links.
`single.jsp`	Displays the content for JSPSingleContainerProvider.

JSPTabContainerProvider JSPs

JSPTabContainerProvider JSPs are located in the `/etc/opt/SUNWps/desktop/default/JSPTabContainerProvider` directory. These JSPs are used as the default set of JSPs for a new channel based on JSPTabContainerProvider.

The following is a two column table: column one lists the file name; column two provides a brief description.

`availableTabs.jsp`	Displays the tabs and the URLs associated with each tab for activating them on the front page.
`header.jsp`	Displays the header bar for the Tab Container page. (Dynamically included.)
`makeNewTab.jsp`	Provides the content for the Make New Tab page of the tab container.
`makeTopic.jsp`	Provides the content for each of the tab topics in the Make New Tab page.
`menubar.jsp`	Displays the menubar that has the Home, Tabs, Theme, Help, and Logout links. (Dynamically included.)
`remove.jsp`	Displays the remove, rename, and start links for each of the selected pages of the JSP tab container in the Current Tab Settings page.
`removeRenameTab.jsp`	Displays the remove and rename part of the Edit page for the tab container.
`selectedTab.jsp`	Displays the tab image for the current selected tab in the tab container.
`tab.jsp`	Is the main JSP for the tab container. It draws the content page for the tab container. (Dynamically includes `header.jsp` and `menubar.jsp`.)

tabedit.jsp	Displays the Edit page for the tab container where new pages can be added, removed, or renamed.
tabs.jsp	Displays the available tabs and the links for them to be activated on the Desktop.

JSPTabCustomTableContainerProvider JSPs

JSPTabCustomTableContainerProvider JSPs are located in the /etc/opt/SUNWps/desktop/default/JSPTabCustomTableContainerProvider directory. JSPTabCustomTableContainerProvider JSPs are used when the user creates a new tab from scratch in the tabs page.

The following is a two column table: column one lists the file name; column two provides a brief description.

launchPopup.jsp	Displays the windows that are detached from the table container. (Dynamically included.)
leafWrapper.jsp	Displays the content view for each channel inside the table container.
popup.jsp	This JSP is used to draw the content in the detached window for the channel in the detached mode.
popupMenubar.jsp	This JSP is used to draw the menubar in the detached window for a detached channel.
providerCommands.jsp	Displays the provider command bar for each channel inside the table container.
providerWrapper.jsp	This JSP is used to draw the wrapper containing the attach, edit, and help links around the channel displayed in the detached window.
tabcustomtable.jsp	Displays the table container's content view.
tablecolumn.jsp	Handles the left, center and right columns of a table. (Dynamically included.)
tablemaximized.jsp	Handles the channel in the maximized state on the Desktop front page. This JSP is used to draw the HTML around the channel displayed in the maximized state.

tabletopbottom.jsp	Handles the top and bottom channels of a table. (Dynamically included.)

JSPTableContainerProvider JSPs

JSPTableContainerProvider JSPs are located in the /etc/opt/SUNWps/desktop/default/JSPTableContainerProvider directory. JSPTableContainerProvider JSPs are the default JSPs that are used by the JSPTableContainerProvider channels.

The following is a two column table: column one lists the file name; column two provides a brief description.

header.jsp	Displays the product banner that includes the user reference links for the table container.
launchPopup.jsp	Displays the detached windows that are detached from the table container. (Dynamically included.)
leafWraper.jsp	Displays the content view for each channel inside the table container.
menubar.jsp	Displays the menubar that has the Home, Tabs, Theme, Help, and Logout links.
popup.jsp	This JSP is used to draw the content in the detached window for the channel in the detached mode.
popupMenubar.jsp	This JSP is used to draw the menubar in the detached window for a detached channel.
providerCommands.jsp	Displays the provider command bar for each channel inside the table container.
providerWrapper.jsp	This JSP is used to draw the wrapper containing the attach, edit, and help links around the channel displayed in the detached window.
table.jsp	Displays the table container's content view, which is used by the table container inside a frame tab container. (Dynamically includes launchPopup.jsp, tablecolumn.jsp, and tabletopbottom.jsp.)

`tablecolumn.jsp`	Handles the left, center and right columns of a table. (Dynamically included.)
`tablemaximized.jsp`	Handles the channel in the maximized state on the Desktop front page. This JSP is used to draw the HTML around the channel displayed in the maximized state.
`tabletopbottom.jsp`	Handles the top and bottom channels of a table. (Dynamically included.)
`tabtable.jsp`	Displays the table container's content view, which is used by the table container inside a tab container. (Dynamically includes `launchPopup.jsp`, `tablecolumn.jsp`, and `tabletopbottom.jsp`.)
`toptable.jsp`	Displays the table container's content view, which is used by the table container that is the top most container in the Desktop. (Dynamically includes `launchPopup.jsp`, `tablecolumn.jsp`, and `tabletopbottom.jsp`.)

Miscellaneous JSPs

Miscellaneous JSPs are located in the `/etc/opt/SUNWps/desktop/default` directory. These JSPs are used by more than one channel, and are also used as a default if the named JSP is not found in the provider or channel subdirectory.

The following is a two column table: column one lists the file name; column two provides a brief description.

`defaultHeader.jsp`	Displays the default product banner that includes the user reference links.
`defaultMenubar.jsp`	Displays the default menubar that includes the user reference links.
`launchPopup.jsp`	Displays the detached windows that are detached from the table containers. (Dynamically included.)
`PortletBanner.jsp`	This JSP is used to draw the banner on the edit page of a JSR 168 portlet when the edit button is clicked on the portlet.

PortletEdit.jsp	This JSP is used to draw the edit page of a JSR 168 portlet when the edit button is clicked on the portlet.
PortletHelp.jsp	This JSP is used as a wrapper for the portlet's help content on the help page of a JSR 168 portlet when the help button is clicked on the portlet.
PortletMenubar.jsp	This JSP is used to draw the menubar on the edit page of a JSR 168 portlet when the edit button is clicked on the portlet.
providerCommands.jsp	Displays the minimize, maximize, help, edit, detach, remove links in the channel title bar.
searchbox.jsp	Displays the search box that are used in the desktop header area.
singlePreferenceHeader.jsp	Displays the product banner that is used by the single containers.
singlePreferenceMenubar.jsp	Displays the menubar that is used by the single containers.
tablePreferenceHeader.jsp	Displays the product banner that is used by the table containers.
tablePreferenceMenubar.jsp	Displays the menubar that is used by the table containers.
tabPreferenceHeader.jsp	Displays the product banner that is used by the tab containers.
tabPreferenceMenubar.jsp	Displays the menubar that is used by the tab containers.

SampleSimpleWebService JSPs

SampleSimpleWebService JSPs are located in the `/etc/opt/SUNWps/desktop/default/SampleSimpleWebService` directory.

The following is a two column table: column one lists the file name; column two provides a brief description.

CurrencyExchangeService.wsdl	Displays the WSDL file for the current exchange web service.

webserviceContent.jsp	Displays the Content view of the simple web service channels.
webserviceInputEdit.jsp	Displays the Edit view of the simple web service channels.

SampleSimpleWebServiceConfigurable JSPs

SampleSimpleWebServiceConfigurable JSPs are located in the /etc/opt/SUNWps/desktop/default/SampleSimpleWebServiceConfigurable directory.

The following is a two column table: column one lists the file name; column two provides a brief description.

CurrencyExchangeService.wsdl	Displays the WSDL file for the current exchange web service.
webserviceContent.jsp	Displays the Content view of the simple web service configurable channels.
webserviceWsdlEdit.jsp	Displays the Edit view of the simple web service configurable channels.

Search JSPs

Search JSPs are located in the /etc/opt/SUNWps/desktop/default/Search directory.

The following is a two column table: column one lists the file name; column two provides a brief description.

advQuery.jsp	Converts the advanced query to the list format. This JSP consists mostly of Java™ code, which is exposed so that the advanced search query list can be customized according to the schema changes.

`advancedSearch.jsp`	Displays the interface to perform an advanced search, including the description menu. This JSP uses the SearchRequestBean to store request parameters and display the form values. The bean reduces Java™ scriptlets in the JSP. You use this JSP to make changes to the advanced search interface, removal for search fields, and addition of fields.
`basicSearch.jsp`	Displays the interface to perform a basic search.
`browseHeader.jsp`	Contains the browse related code that shows up in the browse interface. Includes the category tree Home link and the Search in all categories, and Search within a category radio buttons.
`browseOnly.jsp`	Sets and executes the parameters for category browsing using the search tag library. This JSP sets all the parameters required to browse and executes the search and includes the `browseResults.jsp` page.
`browseResults.jsp`	Displays category tree in the browse section. (It consists of many Java scriptlets, so modifying this JSP requires good Java proficiency.)
`browseSearch.jsp`	Sets and executes the parameters for searching and browsing within categories using the Search tag library. The JSP sets the RDMType to `rd-request` and query language to search, and sets other search parameters. It includes the `browseSearchResults.jsp` page to display the category matches.
`browseSearchResults.jsp`	Displays the number of category matches found and the links to matching categories. (It consists of many Java scriptlets, so modifying this JSP requires good Java proficiency.)
`descMenu.jsp`	Contains the description menu, that is, the Full, Brief, and Title menus. This menu is included in the basic, advanced, and browse interfaces.
`error.jsp`	Displays error messages.
`pageFooter.jsp`	Displays the list of pages, Next, and Previous links.

psSearch.jsp	Is the portal server related JSP file. The user profile property values are retrieved from the portal server. The customer can substitute this file if the values can be retrieved from other data stores.
results.jsp	Specifies the number of matches found and displays document results, score, title, description, and so on, for each document. (Consists of some Java scriptlets.)
score.jsp	Computes the scale that displays the document match relevance.
searchContent.jsp	Displays the Content view of the Search channel, and delegates the request to other search JSPs, based on the request type. The basic search, advanced search, or browse interfaces are displayed based on the requested mode. The search results are displayed based on the request type. This JSP includes the advancedSearch.jsp or basicSearch.jsp based on user selection. browseSearch.jsp and searchOnly.jsp are included only if the user has specified a query; otherwise the category tree (no search) is displayed from browseOnly.jsp. The pageFooter.jsp is included to display the pagination bar in the Search channel.
searchDoEdit.jsp	Invoked when the user completes processing the Edit page of the Search channel.
searchEdit.jsp	Invoked when the user clicks the Edit button of the Search channel.
searchMenu.jsp	Contains the HTML ribbon for Basic, Advanced, and Browse links.
searchOnly.jsp	Sets and executes the parameters for search using search the tag library. The advQuery.jsp is included if its an advanced search. This JSP includes the results.jsp to display the document matches.

SearchProvider JSPs

SearchProvider JSPs are located in the `/etc/opt/SUNWps/desktop/default/SearchProvider` directory.

The following is a two column table: column one lists the file name; column two provides a brief description.

`advQuery.jsp`	Converts the advanced query to the list format. This JSP consists mostly of Java code, which is exposed so that the advanced search query list can be customized according to the schema changes.
`advancedSearch.jsp`	Displays the interface to perform an advanced search, including the description menu. This JSP uses the `SearchRequestBean` to store request parameters and display the form values. The bean reduces Java scriptlets in the JSP. You use this JSP to make changes to the advanced search interface, removal for search fields, and addition of fields.
`browseHeader.jsp`	Contains the browse-related code that shows up in the browse interface. Includes the category tree Home link and the Search in all categories, and Search within a category radio buttons.
`basicSearch.jsp`	Displays the interface to perform a basic search.
`browseOnly.jsp`	Sets and executes the parameters for category browsing using the search tag library. This JSP sets all the parameters required to browse and executes the search and includes the `browseResults.jsp` page.
`browseResults.jsp`	Displays category tree in the browse section. (It consists of many Java scriptlets, so modifying this JSP requires good Java proficiency.)

`browseSearch.jsp`	Sets and executes the parameters for searching and browsing within categories using the Search tag library. The JSP sets the RDMType to rd-request and query language to search, and sets other search parameters. It includes the `browseSearchResults.jsp` page to display the category matches.
`browseSearchResults.jsp`	Displays the number of category matches found and the links to matching categories. (It consists of many Java scriptlets, so modifying this JSP requires good Java proficiency.)
`descMenu.jsp`	Contains the description menu, that is, the Full, Brief, and Title menus. This menu is included in the basic, advanced, and browse interfaces.
`error.jsp`	Displays error messages.
`pageFooter.jsp`	Displays the list of pages, Next, and Previous links.
`psSearch.jsp`	Is the portal server related JSP file. The user profile property values are retrieved from the portal server. The customer can substitute this file if the values can be retrieved from other data stores.
`results.jsp`	Specifies the number of matches found and displays document results, score, title, description for each document. (Consists of some Java scriptlets.)
`score.jsp`	Computes the scale that displays the document match relevance.

searchContent.jsp	Displays the Content view of the Search channel, and delegates the request to other search JSPs, based on the request type. The basic search, advanced search, or browse interfaces are displayed based on the requested mode. The search results are displayed based on the request type. This JSP includes the advancedSearch.jsp or basicSearch.jsp based on user selection. browseSearch.jsp and searchOnly.jsp are included only if the user has specified a query; otherwise the category tree (no search) is displayed from browseOnly.jsp. The pageFooter.jsp is included to display the pagination bar in the Search channel.
searchDoEdit.jsp	Invoked when the user completes processing the Edit page of the Search channel.
searchEdit.jsp	Invoked when the user clicks the Edit button of the Search channel.
searchMenu.jsp	Contains the HTML ribbon for Basic, Advanced, and Browse links.
searchOnly.jsp	Sets and executes the parameters for search using search the tag library. The advQuery.jsp is included if its an advanced search. This JSP includes the results.jsp to display the document matches.

SimpleWebServiceConfigurableProvider JSPs

SimpleWebServiceConfigurableProvider JSPs are located in the /etc/opt/SUNWps/desktop/default/SimpleWebServiceConfigurableProvider directory. These JSPs are the default JSPs that are used by the SimpleWebServiceConfigurableProvider channels.

The following is a two column table: column one lists the file name; column two provides a brief description.

webserviceContent.jsp	Displays the Content view of the simple web service configurable channels.

webserviceWsdlEdit.jsp	Displays the Edit view of the simple web service configurable channels.

SimpleWebServiceProvider JSPs

SimpleWebServiceProvider JSPs are located in the /etc/opt/SUNWps/desktop/default/SimpleWebServiceProvider directory. These JSPs are default JSPs that are used by the SimpleWebServiceProvider channels.

The following is a two column table: column one lists the file name; column two provides a brief description.

webserviceContent.jsp	Displays the Content view of the simple web service configurable channels.
webserviceInputEdit.jsp	Displays the Edit view of the simple web service channels.

Subscriptions JSPs

The Subscriptions channel JSPs are located in the /etc/opt/SUNWps/desktop/default/Subscriptions directory.

The following is a two column table: column one lists the file name; column two provides a brief description.

subscontent.jsp	Displays the list of subscriptions per type of subscriptions (such as category subscriptions, discussions subscriptions, saved search subscriptions). For each, the list of user's subscriptions labels and associated document hit link is displayed. The hit link, points the end user to display a detailed view of the information matching their subscription.

`subsdoedit.jsp`	Used to manage (such as delete / update) existing subscriptions, or adding new subscriptions when the user clicks on “subscribe to” links from the search or discussion channel. This JSP segments the subscriptions in to three types: category subscriptions, discussions subscriptions, saved search subscriptions.
`subsedit.jsp`	This JSP is triggered to handle the subscriptions changes made by the end user in the page presented by `subsedit.jsp`. The role of this JSP, is to update the Identity Server subscriptions service attributes holding the subscription information.

SubscriptionsProvider JSPs

The SubscriptionsProvider JSPs are located in the `/etc/opt/SUNWps/desktop/default/SubscriptionsProvider` directory.

The following is a two column table: column one lists the file name; column two provides a brief description.

`subscontent.jsp`	Displays the list of subscriptions per type of subscriptions (such as category subscriptions, discussions subscriptions, saved search subscriptions). For each, the list of user’s subscriptions labels and associated document hit link is displayed. The hit link, points the end user to display a detailed view of the information matching their subscription.
`subsdoedit.jsp`	Used to manage (such as delete / update) existing subscriptions, or adding new subscriptions when the user clicks on “subscribe to” links from the search or discussion channel. This JSP segments the subscriptions in to three types: category subscriptions, discussions subscriptions, saved search subscriptions.

`subsedit.jsp`	This JSP is triggered to handle the subscriptions changes made by the end user in the page presented by `subsedit.jsp`. The role of this JSP, is to update the Identity Server subscriptions service attributes holding the subscription information.

TabJSPEditContainer JSPs

TabJSPEditContainer JSPs are located in the `/etc/opt/SUNWps/desktop/default/TabJSPEditContainer` directory.

The following is a two column table: column one lists the file name; column two provides a brief description.

`contentLayoutBar.jsp`	Displays the Edit page for channels in JSPTabContainer.
`doedit.jsp`	Is the process page for TabJSPEditContainer.
`tabedit.jsp`	Displays the Content and Layout links and the current selected tab on the Edit page.

JSPs in the sampleportal Directory

This section provides a listing with description of the JSPs in the `/etc/opt/SUNWps/desktop/sampleportal/*` directory. It contains

- FrameTabContainer JSPs
- JSPContentContainer JSPs
- JSPCreateChannelContainer JSPs
- JSPCustomThemeContainer JSPs
- JSPDynamicSingleContainer JSPs
- JSPEditContainer JSPs
- JSPFrameCustomTableContainerProvider JSPs
- JSPLayoutContainer JSPs
- JSPPopupContainer JSPs

- JSPPresetThemeContainer JSPs
- JSPSingleContainer JSPs
- JSPTabContainer JSPs
- JSPTabCustomTableContainerProvider JSPs
- JSPTableContainerProvider JSPs
- Miscellaneous JSPs
- PredefinedFrontPageFramePanelContainerProvider and PredefinedFrontPageTabPanelContainerProvider
- PredefinedSamplesFramePanelContainerProvider and PredefinedSamplesTabPanelContainerProvider
- SampleJSP JSPs
- SampleSimpleWebService JSPs

NOTE All the JSPs in sampleportal directory inherit the new Themes.

FrameTabContainer JSPs

FrameTabContainer JSPs are located in the `/etc/opt/SUNWps/desktop/sampleportal/FrameTabContainer` directory.

The following is a two column table: column one lists the file name; column two provides a brief description.

`banner.jsp`	Contains the required JavaScript™ and the body tag for the right frame of the frame tab container.
`frameset.jsp`	Contains the HTML source for frames of the frameset container.
`frametab.jsp`	Is the main JSP for the frame tab container and throws out the requested content for each frame based on the request parameters from the frameset JSP.
`frametabedit.jsp`	Displays the Edit page for the frame tab container where new pages can be added, removed, or renamed.

`frametabmenu.jsp`	Displays the left frame for the frame tab container that has the list of available pages and links to them.
`header.jsp`	Displays the product banner that is used by the frame tab container.
`makeNewTab.jsp`	Provides the content for the Make New Page part of the Sections page in the frame tab container.
`makeTopic.jsp`	Provides the content for each of the page topics in the Make New Page.
`menubar.jsp`	Displays the menubar that has the Home, Tabs, Theme, Help, and Logout links.
`remove.jsp`	Displays the remove, rename, and start links for each of the selected pages of the frame tab container in the Current Tab Settings page.
`removeRenameTab.jsp`	Displays the Start Page, Tab, and Actions part of the Current Tab Settings page for the frame tab container.
`selectedTab.jsp`	Displays the content for the right frame of the frame tab container.

JSPContentContainer JSPs

The JSPContentContainer JSPs are located in the `/etc/opt/SUNWps/desktop/sampleportal/JSPContentContainer` directory. These JSPs are used for the content view when the Content link is selected in a JSP-based table container.

The following is a two column table: column one lists the file name; column two provides a brief description.

`contentedit.jsp`	Displays the content Edit page.

JSPCreateChannelContainer JSPs

The JSPCreateChannelContainer JSPs are located in the `/etc/opt/SUNWps/desktop/sampleportal/JSPCreateChannelContainer` directory.

The following is a two column table: column one lists the file name; column two provides a brief description.

`createchannel.jsp`
`createchannelcontent.jsp`
`createchanneldoedit.jsp`
`createchanneledit.jsp`
`createchannelui.jsp`
`deletechannel.jsp`
`deletechannelui.jsp`
	These JSPs are used for the User Defined Channel page on the Desktop.

JSPCustomThemeContainer JSPs

JSPCustomThemeContainer JSPs are located in the `/etc/opt/SUNWps/desktop/sampleportal/JSPCustomThemeContainer` directory. These JSPs are used when the Custom Theme link is selected in the Themes page.

The following is a two column table: column one lists the file name; column two provides a brief description.

`customthemedoedit.jsp`	Processes the result from the Custom Theme page.
`customthemeedit.jsp`	Displays the Custom Theme Edit page.
`themepreview.jsp`	Displays the preview view of the Custom Theme page.

JSPDynamicSingleContainer JSPs

The JSPDynamicSingleContainer JSPs are located in the `/etc/opt/SUNWps/desktop/sampleportal/JSPDynamicSingleContainer` directory.

The following is a two column table: column one lists the file name; column two provides a brief description.

`dynamicSingle.jsp`	Used by the DynamicSingleContainer to display the channel specified in the request parameter. This JSP uses the Desktop theme.

JSPEditContainer JSPs

The JSPEditContainer JSP is located in the `/etc/opt/SUNWps/desktop/sampleportal/JSPEditContainer` directory. These JSPs are used when the Edit icon is selected in a channel title bar inside a JSP- based container. Channels that have the `editType` defined as `EDIT_SUBSET` use these JSPs. The difference between this JSP and the one from default directory is that this includes the Desktop theme style.

The following is a two column table: column one lists the file name; column two provides a brief description.

`edit.jsp`	Displays the Edit view of a channel. Also provides a wrapper around the actual Edit view for a given channel.

JSPFrameCustomTableContainerProvider JSPs

The JSPFrameCustomTableContainerProvider JSP is located in the `/etc/opt/SUNWps/desktop/sampleportal/JSPFrameCustomTableContainerProvider` directory.

JSPFrameCustomTableContainerProvider JSPs are mainly used by the user created pages in the frame tab container. When the user creates a new page from the Sections page in the frame tab container, a new table container is created dynamically. The following JSPs are used by the user created table containers. The difference between this JSP and the one from default directory is that this includes the Desktop theme style.

The following is a two column table: column one lists the file name; column two provides a brief description.

`framecustomtable.jsp`	Displays the content for the newly created page (table container) from the Sections page.

JSPLayoutContainer JSPs

The JSPLayoutContainer JSPs are located in the `/etc/opt/SUNWps/desktop/sampleportal/JSPLayoutContainer` directory. These JSPs are used to display the Layout view when the Layout link is selected in a JSP-based table container.

The following is a two column table: column one lists the file name; column two provides a brief description.

`layoutedit.jsp`	Displays the Layout Edit page.

JSPPopupContainer JSPs

The JSPPopupContainer JSPs are located in the `/etc/opt/SUNWps/desktop/sampleportal/JSPPopupContainer` directory. These JSPs were used to detach a channel from a JSP-based Desktop. However, this container is no longer used by the sample portal. The detached windows in sample portal are now drawn by the `popup.jsp`, `popupmenubar.jsp`, and `providerwrapper.jsp` in JSPTableContainer. These JSPs are present for backward compatibility only.

The following is a two column table: column one lists the file name; column two provides a brief description.

`popup.jsp`	Displays the contents of the channel inside the detached window.
`popupMenubar.jsp`	Displays the Update, Close, and Logout links inside the popup window.
`providerWrapper.jsp`	Combines the above JSPs.

JSPPresetThemeContainer JSPs

The JSPPresetThemeContainer JSPs are located in the `/etc/opt/SUNWps/desktop/sampleportal/JSPPresetThemeContainer` directory. These JSPs are used when the Preset Themes link is selected in the Themes page.

The following is a two column table: column one lists the file name; column two provides a brief description.

`themedoedit.jsp`	Processes the result from the Preset Theme page.
`themeedit.jsp`	Displays the Preset Theme Edit page.
`themepreview.jsp`	This file exists for backward compatibility and shows that the theme changes apply to a channel only.

JSPSingleContainer JSPs

The JSPSingleContainer JSPs are located in the `/etc/opt/SUNWps/desktop/sampleportal/JSPSingleContainer` directory.

The following is a two column table: column one lists the file name; column two provides a brief description.

`header.jsp`	Displays the product banner that is used by the Single Container page.
`menubar.jsp`	Displays the menubar that has the Home, Tabs, Theme, Help, and Logout links.
`single.jsp`	Displays the content for JSPSingleContainerProvider.

JSPTabContainer JSPs

The JSPTabContainer JSPs are located in the `/etc/opt/SUNWps/desktop/sampleportal/JSPTabContainer` directory.

The following is a two column table: column one lists the file name; column two provides a brief description.

`header.jsp`	Displays the product banner that is used by the Tab Container page. (Dynamically included.)

menubar.jsp	Displays the menubar that has the Home, Tabs, Theme, Help, and Logout links. (Dynamically included.)
selectedTab.jsp	Displays the tab image for the current selected tab in the tab container.
tab.jsp	Is the main JSP for the tab container. It draws the content page for the tab container. (Dynamically includes header.jsp and menubar.jsp.)
tabedit.jsp	Displays the Edit page for the tab container where new pages can be added, removed, or renamed.
tabs.jsp	Generates the available tabs and the links for them to be activated on the Desktop.

NOTE The header.jsp and menubar.jsp have the new Themes look.

JSPTabCustomTableContainerProvider JSPs

The JSPTabCustomTableContainer JSPs is located in the /etc/opt/SUNWps/desktop/sampleportal/JSPTabCustomTableContainerProvider directory.

The following is a two column table: column one lists the file name; column two provides a brief description.

tabcustomtable.jsp	Displays the table container's content view.

JSPTableContainerProvider JSPs

The JSPTableContainer JSPs are located in the /etc/opt/SUNWps/desktop/sampleportal/JSPTableContainerProvider directory.

The following is a two column table: column one lists the file name; column two provides a brief description.

`header.jsp`	Displays the product banner that includes the user reference links for the table container.
`leafWraper.jsp`	Displays the content view for each channel inside the table container.
`menubar.jsp`	Displays the menubar that has the Home, Theme, Help, and Logout links.
`popup.jsp`	This JSP is used to draw the content in the detached window for the channel in the detached mode.
`popupMenubar.jsp`	This JSP is used to draw the menubar in the detached window for a detached channel.
`providerWrapper.jsp`	This JSP is used to draw the wrapper containing the attach, edit, and help links around the channel displayed in the detached window.
`table.jsp`	Displays the table container's content view, which is used by the table container inside a frame tab container. (Dynamically includes `launchPopup.jsp`, `tablecolumn.jsp`, and `tabletopbottom.jsp`.)
`tabtable.jsp`	Displays the table container's content view, which is used by the table container inside a tab container. (Dynamically includes `launchPopup.jsp`, `tablecolumn.jsp`, and `tabletopbottom.jsp`.)
`toptable.jsp`	Displays the table container's content view, which is used by the table container that is the top most container in the Desktop. (Dynamically includes `launchPopup.jsp`, `tablecolumn.jsp`, and `tabletopbottom.jsp`.)

Miscellaneous JSPs

Miscellaneous JSPs are located in the `/etc/opt/SUNWps/desktop/sampleportal` directory. These JSPs are used by more than one channel, and are also used as a default if the named JSP is not found in the provider or channel subdirectory.

The following is a two column table: column one lists the file name; column two provides a brief description.

PortletBanner.jsp	This JSP is used to draw the banner on the edit page of a JSR 168 portlet when the edit button is clicked on the portlet.
PortletMenubar.jsp	This JSP is used to draw the menubar on the edit page of a JSR 168 portlet when the edit button is clicked on the portlet.
defaultHeader.jsp	Displays the default product banner that includes the user reference links.
defaultMenubar.jsp	Displays the default menubar that includes the user reference links.
framePreferenceHeader.jsp	Displays the product banner that is used by the frame tab containers.
framePreferenceMenubar.jsp	Displays the menubar that is used by the frame tab containers.
searchbox.jsp	Displays the search box that are used in the desktop header area.
singlePreferenceHeader.jsp	Displays the product banner that is used by the single containers.
singlePreferenceMenubar.jsp	Displays the menubar that is used by the single containers.
tabPreferenceHeader.jsp	Displays the product banner that is used by the tab containers.
tabPreferenceMenubar.jsp	Displays the menubar that is used by the tab containers.
tablePreferenceHeader.jsp	Displays the product banner that is used by the table containers.
tablePreferenceMenubar.jsp	Displays the menubar that is used by the table containers.

PredefinedFrontPageFramePanelContainerProvider

The PredefinedFrontPageFramePanelContainerProvider JSPs are located in the `/etc/opt/SUNWps/desktop/sampleportal/PredefinedFrontPageFramePanelContainerProvider` directory.

The following is a two column table: column one lists the file name; column two provides a brief description.

`header.jsp`	Displays the product banner that includes the user reference links for the table container.
`launchPopup.jsp`	Displays the detached windows that are detached from the table container. (Dynamically included.)
`leafWrapper.jsp`	Displays the content view for each channel inside the table container.
`menubar.jsp`	Displays the menubar that has the Home, Theme, Help, and Logout links.
`popup.jsp`	This JSP is used to draw the content in the detached window for the channel in the detached mode.
`popupMenubar.jsp`	This JSP is used to draw the menubar in the detached window for a detached channel.
`providerCommands.jsp`	Displays the channel command buttons and links, such as Remove, Minimize / Maximize, Edit, and Help.
`providerWrapper.jsp`	This JSP is used to draw the wrapper containing the attach, edit, and help links around the channel displayed in the detached window.
`table.jsp`	Displays the table container's content view, which is used by the table container inside a frame tab container. (Dynamically includes `launchPopup.jsp`, `tablecolumn.jsp`, and `tabletopbottom.jsp`.)
`tablecolumn.jsp`	Handles the left, center and right columns of a table. (Dynamically included.)

tablemaximized.jsp	Handles the channel in the maximized state on the Desktop front page. This JSP is used to draw the HTML around the channel displayed in the maximized state.
tabletopbottom.jsp	Handles the top and bottom channels of a table. (Dynamically included.)
tabtable.jsp	Displays the table container's content view, which is used by the table container inside a tab container. (Dynamically includes launchPopup.jsp, tablecolumn.jsp, and tabletopbottom.jsp.)
toptable.jsp	Displays the table container's content view, which is used by the table container that is the top most container in the Desktop. (Dynamically includes launchPopup.jsp, tablecolumn.jsp, and tabletopbottom.jsp.)

PredefinedFrontPageTabPanelContainerProvider

The PredefinedFrontPageTabPanelContainerProvider JSPs are located in the /etc/opt/SUNWps/desktop/sampleportal/PredefinedFrontPageTabPanelContainerProvider directory.

The following is a two column table: column one lists the file name; column two provides a brief description.

header.jsp	Displays the product banner that includes the user reference links for the table container.
launchPopup.jsp	Displays the detached windows that are detached from the table container. (Dynamically included.)
leafWrapper.jsp	Displays the content view for each channel inside the table container.
menubar.jsp	Displays the menubar that has the Home, Theme, Help, and Logout links.
popup.jsp	This JSP is used to draw the content in the detached window for the channel in the detached mode.

`popupMenubar.jsp`	This JSP is used to draw the menubar in the detached window for a detached channel.
`providerCommands.jsp`	Displays the channel command buttons and links, such as Remove, Minimize / Maximize, Edit, and Help.
`providerWrapper.jsp`	This JSP is used to draw the wrapper containing the attach, edit, and help links around the channel displayed in the detached window.
`table.jsp`	Displays the table container's content view, which is used by the table container inside a frame tab container. (Dynamically includes `launchPopup.jsp`, `tablecolumn.jsp`, and `tabletopbottom.jsp`.)
`tablecolumn.jsp`	Handles the left, center and right columns of a table. (Dynamically included.)
`tablemaximized.jsp`	Handles the channel in the maximized state on the Desktop front page. This JSP is used to draw the HTML around the channel displayed in the maximized state.
`tabletopbottom.jsp`	Handles the top and bottom channels of a table. (Dynamically included.)
`tabtable.jsp`	Displays the table container's content view, which is used by the table container inside a tab container. (Dynamically includes `launchPopup.jsp`, `tablecolumn.jsp`, and `tabletopbottom.jsp`.)
`toptable.jsp`	Displays the table container's content view, which is used by the table container that is the top most container in the Desktop. (Dynamically includes `launchPopup.jsp`, `tablecolumn.jsp`, and `tabletopbottom.jsp`.)

PredefinedSamplesFramePanelContainerProvider

The PredefinedSamplesFramePanelContainerProvider JSPs are located in the `/etc/opt/SUNWps/desktop/sampleportal/PredefinedSamplesFramePanelContainerProvider` directory.

The following is a two column table: column one lists the file name; column two provides a brief description.

`header.jsp`	Displays the product banner that includes the user reference links for the table container.
`launchPopup.jsp`	Displays the detached windows that are detached from the table container. (Dynamically included.)
`leafWrapper.jsp`	Displays the content view for each channel inside the table container.
`menubar.jsp`	Displays the menubar that has the Home, Theme, Help, and Logout links.
`popup.jsp`	This JSP is used to draw the content in the detached window for the channel in the detached mode.
`popupMenubar.jsp`	This JSP is used to draw the menubar in the detached window for a detached channel.
`providerCommands.jsp`	Displays the channel command buttons and links, such as Remove, Minimize / Maximize, Edit, and Help.
`providerWrapper.jsp`	This JSP is used to draw the wrapper containing the attach, edit, and help links around the channel displayed in the detached window.
`table.jsp`	Displays the table container's content view, which is used by the table container inside a frame tab container. (Dynamically includes `launchPopup.jsp`, `tablecolumn.jsp`, and `tabletopbottom.jsp`.)
`tablecolumn.jsp`	Handles the left, center and right columns of a table. (Dynamically included.)

tablemaximized.jsp	Handles the channel in the maximized state on the Desktop front page. This JSP is used to draw the HTML around the channel displayed in the maximized state.
tabletopbottom.jsp	Handles the top and bottom channels of a table. (Dynamically included.)
tabtable.jsp	Displays the table container's content view, which is used by the table container inside a tab container. (Dynamically includes launchPopup.jsp, tablecolumn.jsp, and tabletopbottom.jsp.)
toptable.jsp	Displays the table container's content view, which is used by the table container that is the top most container in the Desktop. (Dynamically includes launchPopup.jsp, tablecolumn.jsp, and tabletopbottom.jsp.)

PredefinedSamplesTabPanelContainerProvider

The PredefinedSamplesTabPanelContainerProvider JSPs are located in the /etc/opt/SUNWps/desktop/sampleportal/PredefinedSamplesTabPanelContainer Provider directory.

The following is a two column table: column one lists the file name; column two provides a brief description.

header.jsp	Displays the product banner that includes the user reference links for the table container.
launchPopup.jsp	Displays the detached windows that are detached from the table container. (Dynamically included.)
leafWrapper.jsp	Displays the content view for each channel inside the table container.
menubar.jsp	Displays the menubar that has the Home, Theme, Help, and Logout links.
popup.jsp	This JSP is used to draw the content in the detached window for the channel in the detached mode.
popupMenubar.jsp	This JSP is used to draw the menubar in the detached window for a detached channel.

providerCommands.jsp	Displays the channel command buttons and links, such as Remove, Minimize / Maximize, Edit, and Help.
providerWrapper.jsp	This JSP is used to draw the wrapper containing the attach, edit, and help links around the channel displayed in the detached window.
table.jsp	Displays the table container's content view, which is used by the table container inside a frame tab container. (Dynamically includes launchPopup.jsp, tablecolumn.jsp, and tabletopbottom.jsp.)
tablecolumn.jsp	Handles the left, center and right columns of a table. (Dynamically included.)
tablemaximized.jsp	Handles the channel in the maximized state on the Desktop front page. This JSP is used to draw the HTML around the channel displayed in the maximized state.
tabletopbottom.jsp	Handles the top and bottom channels of a table. (Dynamically included.)
tabtable.jsp	Displays the table container's content view, which is used by the table container inside a tab container. (Dynamically includes launchPopup.jsp, tablecolumn.jsp, and tabletopbottom.jsp.)
toptable.jsp	Displays the table container's content view, which is used by the table container that is the top most container in the Desktop. (Dynamically includes launchPopup.jsp, tablecolumn.jsp, and tabletopbottom.jsp.)

SampleJSP JSPs

The SampleJSP JSPs are located in the /etc/opt/SUNWps/desktop/sampleportal/SampleJSP directory. These JSPs are used by the SampleJSP channel.

The following is a two column table: column one lists the file name; column two provides a brief description.

`samplecontent.jsp`	Displays the contents of the Sample JSP channel. This JSP uses of the Desktop theme, and is used as an example of the `channelHighlightColor` attribute.

SampleSimpleWebService JSPs

The SampleSimpleWebService JSP is located in the `/etc/opt/SUNWps/desktop/sampleportal/SampleSimpleWebService` directory.

The following is a two column table: column one lists the file name; column two provides a brief description.

`webserviceContent.jsp`	Displays the Content view of the simple web service channels.

Miscellaneous JSP Information

This section contains miscellaneous information on using JSPs when customizing the Desktop.

Performing Redirects

To perform redirects from the `doedit.jsp` page back to the `edit.jsp` page, use the following URL:

```
response.sendRedirect(p.getProviderContext().getDesktopURL(request).toStrin
g() + "?action=edit&provider=ipsdtJSPEditChannel" + "&targetprovider=" +
p.getName() + "&reedit=true");
```

To return back to the Edit page, use the following, which does not hardcode the editchannel name.

```
String editChannel = request.getParameter("editChannelName");
response.sendRedirect(p.getProviderContext().getDesktopURL(request).toStrin
g() + "?action=edit&provider=<%=editChannel%>&targetprovider=" +
p.getName() + "&redit=true");
```

JSP vs. Theme Color

The colors of a theme can be changed in two places: in the JSP or template file that uses the theme tag, and in the display profile using the GlobalThemes attribute.

If you change the color value in the JSP or template file, the Desktop uses this value and not what is in the GlobalThemes attribute.

For example, if you manually change the following background color in a JSP file

```
BGCOLOR="<dttheme:getAttribute name="borderColor"/>"
```

to

```
BGCOLOR="#FFFFFF"
```

then the Desktop shows white for border color, and the theme color is not used. There is no way the theme or the Desktop can detect this.

Recompiling JSPs

Every time you make a modification to a JSP file, you need to recompile. You do this by running the `touch` command on the modified container's top-level JSP file.

For example, type **`touch tab.jsp`**.

NOTE A typical desktop will include content from several JSPs. However, if you make a modification to a non-toplevel JSP that has been included in a top level JSP, the included JSP will not be recomplied. The end result is your desktop changes will never be reflected.

If the top-level JSP is touched, the JSP engine recompiles all the relevant JSPs. If you cannot find the top level JSP, run the `touch` command on all JSPs in the directory, for example,

```
touch *.jsp
```

This modifies all JSPs, including the top-level JSP.

See "JavaServer Page Caching Information" on page 298 for information on how to find the top-level (parent) JSP.

JavaServer Page Caching Information

When the system compiles JavaServer Pages™ (JSP™), the result is only one Java™ class per parent JSP. There is no compiled class for the static included JSPs. Thus, when you change an included JSP, you need to run the touch command on the parent JSP to recompile the parent JSP with the changed JSP. The Portal Server software JSP engine checks the last modification time on the compiled class and the JSP file to see if the JSP needs to be recompiled. In this way, the change takes effect immediately.

To find a JSP's parent JSP, search in the JSP for the string `<%@ include file="filename.jsp" %>`. Some JSPs are dynamically included by using the `<jsp:include page="header.jsp" flush="true"/>` syntax instead of `<%@ include file="header.jsp" %>`. This syntax compiles `header.jsp` and generates a separate Java class.

The path to cached JSPs is constructed in such a way so that the compiled JSPs do not conflict with each other in multi-server instances, when multiple Desktop types contain the same JSPs and for multiple clientTypes and locales. So when JSPs are dynamically included, the touch command does not need to be run for the parent JSP.

Debugging JSPs

The JSP classes are created at:

```
/var/opt/SUNWps/https-psservername/portal/tmp/_jsps
```

You can find compilation and runtime errors in the Desktop debug log at:

```
/var/opt/SUNWam/debug/desktop.debug
```

Also, all JSPProvider based Desktop channels have a property called `showExceptions`. This property, by default, is set to `false`; setting it to `true` causes the JSP exception to show up as the content of the channel.

Appendix D

JavaServer Pages Tag Library Reference

This appendix describes the Desktop custom tag library and serves as a reference to the tags available within the library.

This appendix contains the following sections:

- Tag Library Overview
- Using Tags in JSPs
- Tag Overview
- Tag Reference

See the following supplemental documentation for more information on JavaServer Pages™ (JSP™) Custom Tags and Apache's JSP Standard Tag Library (JSTL):

`http://developer.java.sun.com/developer/technicalArticles/xml/WebAppDev3/`

`http://jakarta.apache.org/taglibs/doc/standard-doc/`

Tag Library Overview

In JavaSever Pages technology, actions are elements that can create and access programming language objects and affect the output stream. JSP technology supports reusable modules called custom actions. You invoke a custom action by using a custom tag in a JSP. A tag library is a collection of custom tags. The Desktop custom tag library contains tags that you use to perform Desktop operations for JSPs.

Overview

Before tag libraries, JSPs were difficult to maintain because you were forced to use JavaBeans™ components and scriptlets as the main mechanism for performing tasks. Custom actions, that is, a tag library, alleviate this problem by bringing the benefits of another level of componentization to JSP. A tag library encapsulates recurring tasks so that they can be reused across more than one application.

The Sun Java System Portal Server software Desktop tag library consists of six parts:

- Core tags that can be used on any provider or container that implement the PAPI interface.
- Tags that can be used to operate on a provider or container that support the ProviderContext and ContainerProviderContext interfaces.
- Tags that operate on specific container building-block providers (SingleContainer, TableContainer, TabContainer, and so on).
- JSP Standard tag libraries from Apache.
- Tags that support the Search function.
- Tags that provide theme support in the Desktop.

The Desktop tag library has the following Tag Library Descriptors (TLDs) in the `/etc/opt/SUNWps/desktop/default/tld` directory. The tag library is exposed, for convenience, through using multiple TLDs, so that tags are in their appropriate functional area.

`desktop.tld`	Contains core Desktop tags that bring forward the functionality available from the ProviderContext interface.
`desktopContainerProviderContext.tld`	Contains tags that bring forward the functionality available from the ContainerProviderContext interface.
`desktopProviderContext.tld`	Contains tags to operate on providers that extend the ProviderContext
`desktopSingle.tld`	Contains tags to operate on single container channels. Single container channels are based on JSPSingleContainerProvider or a subclass thereof.

desktopTable.tld	Contains tags to operate on table container channels. Table container channels are based on JSPTableContainerProvider or a subclass thereof.
desktopTab.tld	Contains tags to operate on tab container channels. Tab container channels are based on JSPTabContainerProvider or a subclass thereof.
desktopTheme.tld	Contains tags for theme support in the Desktop.
im.tld	Contains tags for IMProvider class.
jr.tld	Contains tags that accept rtexprvalues for their attributes. This is a JSP Standard tag library from Apache.
jx.tld	Contains tags that accept attribute values specified using the “expression languages” that JSPTL introduces, which currently is only simplest possible expression language (SPEL). This is a JSP Standard tag library from Apache.
search.tld	Contains tags for search support in the Desktop.

The many of the Java™ classes that support these tags reside in a JAR file desktoptl.jar in the *Web-Container-Instance*/portal/web-apps/WEB-INF/lib directory. The classes for the jx.tld and jr.tld tags reside in the jsptl.jar file.

Desktop Tag Library Hierarchy

The Desktop tag library can be viewed as a wrapper of PAPI, ProviderContext, ContainerProviderContext, and specific container building-block providers in the Portal Server software. Thus, a hierarchy is implied.

For example, the ContainerProviderContext (interface) extends ProviderContext (interface). When you use a tag in desktopContainerProviderContext.tld, it also make sense to use it in desktopProviderContext.tld. Similarly, when you use a tag in desktopProviderContext.tld, it also makes sense to use provider tags in desktop.tld because ProviderAdapter implements Provider. By putting the tag in the level that provides the most use, you will not have to make duplicate tags.

At the bottom of this chain are the specific containers (single, table, and tab). Because all these containers extend JSPContainerProviderAdapter, they can use tags in their respective TLDs, as well as tags in `desktopContainerProviderContext.tld`, `desktopProviderContext.tld`, and `desktop.tld`.

Using Tags in JSPs

In your JSP you can use HTML, Java, JavaScript™, and tags to bring in repetitive Java functions. Tags encapsulate core functionality common to many JSP applications.

In Code Example D-1, a JSP for a single channel, the tag libraries are defined at the top. The library's symbol is then used with the tag name to bring in the appropriate information or function.

Code Example D-1 Listing of `single.jsp`

```
<%--
    Copyright 2001 Sun Microsystems, Inc.  All rights reserved.
    PROPRIETARY/CONFIDENTIAL.  Use of this product is subject to license terms.
--%>
<%-- single.jsp  --%>

<%@ taglib uri="/tld/desktop.tld" prefix="dt" %>
<%@ taglib uri="/tld/desktopSingle.tld" prefix="dtsingle" %>

<%@ page session=”false” %>

<dt:obtainContainer container="$JSPProvider">
    <dtsingle:singleContainerProvider>
    <dtsingle:obtainSelectedChannel>
<%@ include file="header.jsp" %>
<img src="<dt:scontent/>/desktop/images/nothing.gif" height="8" width="0" border="0"
alt=""><br>

<!-- provider content goes here -->
        <dt:getContent/>
    </dtsingle:obtainSelectedChannel>
<%@ include file="menubar.jsp" %>
<%@ include file="footer.html" %>
    </dtsingle:singleContainerProvider>
</dt:obtainContainer>
```

In Code Example B-1, the following lines define the tag libraries that are used in the single.jsp template file.

```
<%@ taglib uri="/tld/desktop.tld" prefix="dt" %>
<%@ taglib uri="/tld/desktopSingle.tld" prefix="dtsingle" %>
```

NOTE You need to include a taglib directive in the page before any custom tag is used.

The `dt:obtainContainer container="$JSPProvider"` tag gets a return value by reference (see “Attributes and Return Values” on page 436 for more information).

Three tags are used out of the `desktop.tld` library. They are `obtainContainer`, `scontent`, and `getContent`. See Code Example D-2 for a partial listing of `desktop.tld` including these three tags. The class and property information about each tag is defined here.

Code Example D-2 Partial Listing of `desktop.tld`

```
<?xml version=”1.0” encoding=”ISO-8859-1”?>

<!DOCTYPE taglib PUBLIC “-//Sun Microsystems, Inc.//DTD JSP Tag Library 1.1//EN”
“http://java.sun.com/j2ee/dtds/web-jsptaglibrary_1_1.dtd”>

<taglib>
    <tlibversion>0.1</tlibversion>
    <jspversion>1.1</jspversion>
    <shortname>desktop</shortname>

    <tag>
            <name>obtainChannel</name>
           <tagclass>com.sun.portal.desktop.taglib.provider.ObtainChannelTag</tagclass>
            <bodycontent>JSP</bodycontent>
            <attribute>
                <name>channel</name>
                <required>true</required>
                <rtexprvalue>false</rtexprvalue>
            </attribute>
    </tag>
    <tag>
            <name>obtainContainer</name>
           <tagclass>com.sun.portal.desktop.taglib.container.ObtainContainerTag</tagclass>
            <bodycontent>JSP</bodycontent>
```

Code Example D-2 Partial Listing of `desktop.tld` *(Continued)*

```
            <attribute>
                <name>container</name>
                <required>true</required>
                <rtexprvalue>false</rtexprvalue>
            </attribute>
    </tag>

------------clip

    <tag>
             <name>getContent</name>
            <tagclass>com.sun.portal.desktop.taglib.provider.GetContentTag</tagclass>
             <bodycontent>empty</bodycontent>
    </tag>
------------clip

    <!-- Utility Tag, can be used anywhere -->
    <tag>
             <name>scontent</name>
             <tagclass>com.sun.portal.desktop.taglib.util.SContentTag</tagclass>
             <bodycontent>empty</bodycontent>
    </tag>
</taglib>
```

In Code Example D-2 on page 305, the bodycontent property has a value of JSP or empty. The JSP value means that you can add more content including more Java code or JSP tags between the tag open and tag close for this particular tag. The empty value means that a tag can contain no extra content and is complete in itself.

In Code Example D-2 on page 305, the required property has a value of `true` meaning that a value for that attribute is required. If the required property has a value of `false`, a value for the attribute is optional. The `rtexprvalue` or run time expression value means that an attribute value can be evaluated at run time or can have a hardcoded value. By default, all of the core, provider, and container tags have this value as `false`.

Two tags are used out of the `desktopSingle.tld` library. They are `singleContainerProvider` and `obtainSelectedChannel`. See Code Example D-3 on page 307 for the listing of `desktopSingle.tld` with these two tags. The class and property information about each tag is defined here.

Code Example D-3 Listing of `desktopSingle.tld`

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<!DOCTYPE taglib PUBLIC "-//Sun Microsystems, Inc.//DTD JSP Tag Library 1.1//EN"
"http://java.sun.com/j2ee/dtds/web-jsptaglibrary_1_1.dtd">

<taglib>
    <tlibversion>0.1</tlibversion>
    <jspversion>1.1</jspversion>
    <shortname>desktopSingle</shortname>

    <!-- Validator Tag -->

    <tag>
            <name>singleContainerProvider</name>

<tagclass>com.sun.portal.desktop.taglib.container.single.SingleContainerProviderTag</tagcla
ss>
            <bodycontent>JSP</bodycontent>
    </tag>

    <tag>
            <name>obtainSelectedChannel</name>

<tagclass>com.sun.portal.desktop.taglib.container.single.ObtainSelectedChannelTag</tagclass
>
            <bodycontent>JSP</bodycontent>
    </tag>

</taglib>
```

The first tag in `desktopSingle.tld` is a validator tag. The validator tag ensures that the JSP using the tag library is a valid type, looking for single container information, not tab information.

Using the Desktop Tag Library in Your Application

To use the Desktop tag library within your web application, you need to set up the context properly by using the Context Setup tags (tags with the prefix `obtain`). See Table D-1 on page 312 for the Context Setup tags.

Because it is possible to have provider(s) within a container, to access a channel within the container you would need the following (partial JSP with `obtainContainer` and `obtainChannelFromContainer` tags):

Code Example D-4 Partial JSP

```
<desktop:obtainContainer container="$JSPProvider">
   <!-- make sure we can use the obtainChannelsFromContainer  ->
   <desktopcpc:containerProviderContext>
     <desktopcpc:obtainChannelFromContainer channel="myChannel">
       <!-- tags to operate on channel "myChannel" within the container stored as attribute
"JSPProvider" in page context-->

     </desktopcpc:obtainChannelFromContainer>
   </desktopcpc:containerProviderContext>
</desktop:obtainContainer>
```

The Desktop tag library also provides an `obtainParentContainer` tag, which gives you access to the parent container anytime (adding the `obtainParentContainer` tag to the partial JSP):

Code Example D-5 Partial JSP with `obtainParentContainer` Tag

```
<desktop:obtainContainer container="$JSPProvider">
  <!-- make sure we can use the obtainChannelsFromContainer  ->
  <desktopcpc:containerProviderContext>
     <desktopcpc:obtainChannelFromContainer channel="myChannel">

      <!-- tags to operate on channel "myChannel" within the container stored as attribute
"JSPProvider" in page context -->

      <desktopcpc:obtainParentContainer>
        <!-- tags to operate on the container stored as attribute "JSPProvider" in page
context -->
      </desktopcpc:obtainParentContainer>

      <!-- more tags to operate on channel "myChannel" within the container stored as
attribute "JSPProvider" in page context -->

    </desktopcpc:obtainChannelFromContainer>
  </desktopcpc:containerProviderContext>
</desktop:obtainContainer>
```

NOTE The current design does not allow multiple levels of containment hierarchy in a single JSP, thus the tag library does not support deeper nesting of the obtain tags.

Example Tab Page with Selected Channels

The following example shows how the `obtainContainer`, `obtainChannelsFromContainer`, and `obtainParentContainer` tags can be used in a JSP to produce a tab page with selected channels.

```
<desktop:obtainContainer container="$JSPProvider">
<!-- get the selected channel list and store it in attribute selectedChannel
-->
    <desktop:getSelectedChannels id="selectedChannel"/>
        <!-- make sure we can use the obtainChannelsFromContainer  ->
        <desktopcpc:containerProviderContext>
            <!-- loop through each channel using Jakarta forEach tag.  With
each iteration, the next -->
            <!-- channel's value (name of the channel) will be stored in
attribute "channel" -->
            <jx:forEach var="channel" items="$selectedChannel">
            <desktopcpc:obtainChannelFromContainer channel="myChannel">
                <!-- print out the title of this channel -->
                <desktop:getTitle/>
                <br>

                <desktopcpc:obtainParentContainer>
                    <!-- make sure the container is a tab container be we
use a tab container tag -->
                    <desktoptab:tabContainerProvider>
                        <!-- store the selected tab name in attribute
selectedTabName -->
                        <desktoptab:getSelectedTabName
id="selectedTabName"/>
                        <!-- declare selectedTabName as String using Jakarta
tag, so the value can be used in the jsp -->
                        <jx:declare id="selectedTabName"
type="java.lang.String"/>
                        <desktop:getName/> container has the tab
<%=selectedTabName%> selected.
                    </desktoptab:tabContainerProvider>
                </desktopcpc:obtainParentContainer>
            </desktopcpc:obtainChannelFromContainer>
        </jx:forEach>
    </desktopcpc:containerProviderContext>
</desktop:obtainContainer>
```

The desktop:getSelectedChannels id="selectedChannel"/ tag provides an example of an empty bodycontent property. The jx:forEach var="channel" items="$selectedChannel" tag is a function from one of the JSP Standard tag libraries included with the Portal Server software.

Tag Overview

This section describes the tag attributes and exceptions.

Attributes and Return Values

You can pass two kinds of attributes to the Desktop library tags:

- String - The simplest way to pass in an attribute is to specify it as a string, for example:

  ```
  <dt:sometag attribute1="myAttribute"/>
  ```

  The string myAttribute becomes the value of attribute1. If a tag expects an integer attribute, use the following:

  ```
  <dt:sometag attribute1="12345"/>
  ```

  The tag uses the corresponding Java classes (java.lang.Integer) to translate the string to an integer value. The same applies to all the primitive Java types (boolean, int, and so on).

- Reference - Sometimes you cannot pass in an attribute as a string. For example, it is impossible to specify a provider object as a string. In this case, the attribute needs to be passed in as a reference defined in the pageContext. You do so by concatenating the character $ with the name of the object stored in the pageContext, for example:

  ```
  <dt:sometag attribute1="$myAttribute"/>
  ```

  In this example, the value of attribute1 is whatever object is stored in the pageContext. The method pageContext.findAttribute( ) is used, not getAttribute( ). So it is possible to define some value in the request object and pass this object in as a reference.

NOTE This mechanism is the main form of communication between tags in the Desktop tag library as well as other tags in other tag libraries (for example, jsptl).

Tags with attributes `id` and `scope` (even if they are optional) are expected to return a value as a result of using the tags. Values can be “returned” in two ways:

- Print to the `JspWriter` stream - If the attribute id is not given when using a tag, the tag prints the value to the `JspWriter` stream. This results in the value showing up in the HTML code generated by the JSPs. In this case, the attribute scope is ignored.
- Store the value in `pageContext` - If you specify the attribute `id` when using a tag, the tag stores the value to the `pageContext` as an attribute with a name specified by the `id` attribute. This operation overwrites whatever original value the attribute has. In addition to the `id`, you can also define the attribute `scope`. It can have one of the following values: `page`, `request`, `session`, or `application`. These values correspond to the `PAGE_SCOPE`, `REQUEST_SCOPE`, `SESSION_SCOPE`, and `APPLICATION_SCOPE` defined in `javax.servlet.jsp.PageContext`. This specifies the scope in which to save the value. If no or unrecognized scope is given, the default is to save it in page scope.

Tag Library Exceptions

The tag library provides five types of exceptions:

1. Invalid Tag Sequence - Occurs when obtain tags are nested in an invalid sequence. See “Using the Desktop Tag Library in Your Application” on page 307 for the sequencing of the obtain tags. The name of the first tag that violates the sequence is provided.
2. Invalid Provider Type - Occurs when a validation test is not passed. That is, the TLD does not support the current provider in the context. The name of the validator tag that failed is provided.
3. Invalid Parameter - Occurs when an attribute passed in with the tag is not a legal parameter for the tag. The name of the attribute that causes the problem is provided.
4. Undefined Parameter - Occurs when an attribute is passed in as reference, but the attribute is not defined in the page context. The name of the attribute is provided.
5. Empty Context - Occurs when there is no container or provider in the context to operate on. Most likely, the appropriate obtain tag is missing so the context is not set up properly.

Tag Reference

Essentially, the Desktop tag library is a wrapper of the PAPI, the ProviderContext, and ContainerProviderContext interfaces, and the specific table containers in Portal Server software. The tags in the Desktop tag library fall into three basic groups: context setup tags, validator tags, and normal tags.

Context Setup Tags

These tags, which start with the prefix `obtain`, set up the context (storing the container or provider in question into the `pageContext`). Whatever tag operation that happens within these tags is done on the provider that is set in the context.

Table D-1 describes the context setup tags in the tag libraries. The table has four columns: the first column lists the tag name, the second describes what the tag does, the third column gives what TLD file the tag is in, and the fourth lists that tag's attributes with brief comments.

All the context setup tags contain a value of JSP for the bodycontent tag.

Table D-1 Context Setup Tags

Tag Name	Description	in TLD File	Attributes/Descriptions
`obtainChannel`	Gets channel object.	`desktop.tld`	`channel` (required) - the name of the channel
`obtainContainer`	Gets container object.	`desktop.tld`	`container` (required) - the name of the container
`obtainParentContainer`	Gets the parent container name.	`desktopContainerProviderContext.tld`	none
`obtainChannelFromContainer`	Gets the channel name.	`desktopContainerProviderContext.tld`	`channel` (required) - the name of the channel
`obtainSelectedChannel`	Gets the selected channel name.	`desktopSingle.tld`	none
`obtainTab`	Gets the tab object.	`desktopTab.tld`	`tab` (required) - the name of the tab
`obtainTabByName`	Gets the tab name.	`desktopTab.tld`	`name` (required) - the name of the tab

Validator Tags

These tags validate that the provider in context can legally use the tags in the TLD. Each TLD, with the exception of `desktop.tld`, has a validator tag defined (usually with the name of the TLD file). If the provider in context cannot use the tags defined in the TLD, an exception is thrown that is displayed on the screen and processing stops.

Users should surround tags that belong to a specific TLD with the respective validator tag. However, it is not possible to enforce that in a JSP environment. To make it easier for users to "guess" which TLD they can use or to debug the JSPs, `getProviderClassName()` and `getContainerClassName()` tags are provided in the `desktop.tld`. They return the class name of the container or provider in the context.

The following is a two column table: column one lists the tag name; column two provides a brief description.

`providerContext`	Validates that the provider in the context can legally use the tags in `desktopproviderContext.tld`.
`containerProviderContext`	Validates that the provider in the context can legally use the tags in `desktopcontainerProviderContext.tld`.
`singleContainerProvider`	Validates that the provider in the context can legally use the tags in `desktopSingle.tld`.
`obtainSelectedChannelFromRequest`	Gets the channel name from request. This tag is in `desktopSingle.tld`.
`tableContainerProvider`	Validates that the provider in the context can legally use the tags in `desktopTable.tld`.
`tabContainerProvider`	Validates that the provider in the context can legally use the tags in `desktopTab.tld`.

Normal Tags

These tags serve as wrappers of PAPI, ProviderContext, ContainerProviderContext, and the specific containers in the Portal Server software.

For the remaining part of this chapter, unless specified, “tag” refers to the normal tag in the Desktop tag library.

Table D-2 through Table D-7 on page 324 describe the normal tags in the tag libraries. Each table has three columns: the first column lists the tag name, the second a description of the tag’s function, and the third column lists that tag’s attributes with brief comments.

The tables are:

- Table D-2 - `desktop.tld`
- Table D-3 on page 317 - `desktopProviderContext.tld`
- Table D-4 on page 322 - `desktopContainerProviderContext.tld`
- Table D-5 on page 322 - `desktopTab.tld`
- Table D-6 on page 323 - `desktopTable.tld`
- Table D-7 on page 324 - `desktopTheme.tld`
- Search Tags - `search.tld`

NOTE The `desktopSingle.tld` tag library has no normal tags.

All of the normal tags in the `desktop.tld` file have a bodycontent of empty.

Table D-2 desktop.tld Normal Tags with Attributes

Tag Name	Description	Attributes/Descriptions
`getProviderClassName`	Returns the class name of the provider that backs the channel.	`id` (optional) `scope` (optional)
`getContent`	Returns a string buffer with the contents of the provider’s object’s default view. This method is called by the clients of the provider object to request the provider’s default view. This method may return null if the provider does not implement a default view. In this case, the provider should return `false` from its `isPresentable()` method.	none
`getTitle`	Returns a string with the title of the channel.	`id` (optional) `scope` (optional) `silentException` (optional)

Table D-2 desktop.tld Normal Tags with Attributes *(Continued)*

Tag Name	Description	Attributes/Descriptions
`getDescription`	Returns a string with the description for the channel.	`id` (optional) `scope` (optional) `silentException` (optional)
`getEdit`	Returns a string buffer with the provider's Edit page.	`id` (optional) `scope` (optional)
`getHelp`	Returns the help URL for this provider. The returned help URL can be either fully qualified URL string (`http://server:port/portal/docs/en/desktop/usedesk.htm`) or a relative path (`desktop/usedesk.htm`). When it is a relative path, the Desktop software resolves it to the full URL.	`id` (optional) `scope` (optional) `silentException` (optional)
`getName`	Returns a string with the name of the provider, which must match the name of the provider the channel was initialized with.	`id` (optional) `scope` (optional)
`getRefreshTime`	Returns a long with the refresh time for this provider in seconds. Use this value to determine if you should fetch a fresh default view for the provider. If the return value from this method is X, you may choose not to fetch fresh content (and use a cached copy instead) if less than X seconds has elapsed since the last time the content was refreshed. If provider content is expected to change infrequently, this method can return some value so that the provider's content is not fetched every time the front page is drawn, thereby saving significant processing time.	`id` (optional) `scope` (optional)
`getWidth`	Returns an integer with the suggested width for the channel to the container of the channel as to how much screen real estate it requires. The values correspond to thick, thin, full top and full bottom.	`id` (optional) `scope` (optional) `silentException` (optional)
getEditType	Returns an integer that defines edit type either EDIT_SUBSET or EDIT_COMPLETE.	`id` (optional) `scope` (optional)
`isEditable`	Returns a Boolean that gives the editable status of the channel. Returns `true` if the channel is editable; otherwise `false`.	`id` (optional) `scope` (optional)

Table D-2 desktop.tld Normal Tags with Attributes *(Continued)*

Tag Name	Description	Attributes/Descriptions
`isPresentable`	Returns a Boolean that gives the presentable status for a channel. Returns `true` if the channel is presentable. Searches for the key HTML with the value true on the client data for the session's client type and returns `true`. If there is no such key, the method returns `true` if the session's client type is named HTML. In both cases, the content-type for the session's client type must equal text/html in order for the method to return `true`.	`id` (optional) `scope` (optional)
`processEdit`	Performs the provider's Edit page processing. Processes a form for this provider. This method is called to process form data associated with the provider. Typically, this method is called to process the Edit page generated from the `getEdit()` method. Usually, the client calling this method on a provider object is the desktop servlet. Form data that is passed into this method in the request has been decoded into Unicode.	`id` (optional) `scope` (optional)
`getContainerClassName`	Returns the class name of the container that backs the container.	`id` (optional) `scope` (optional)
`getSelectedChannels`	Returns a list of selected channel names. The list returned is a Collection of Strings. Each of the Strings is the name of a channel that has been selected.	`id` (required) `scope` (optional)
`getAvailableChannels`	Returns a list of available channel names. The list returned is a Collection of Strings. Each of the Strings is the name of a channel that is available**.**	`id` (required) `scope` (optional)
`scontent`	Returns the URL of the directory that has the static content (for example, images and style sheet). This utility tag can be used anywhere.	`none`

All of the normal tags in the `desktopProviderContext.tld` file have a bodycontent of empty.

Table D-3 desktopProviderContext.tld Normal Tags with Attributes

Tag Name	Description	Attributes/Descriptions
`getStringProperty`	Returns a string with the property value. This is an overloaded method that can return alternately the default or localized version of the property value.	`key` (required) - the name of the property `localized` (optional) `id` (optional) `scope` (optional) `pflist` (optional)
`getBooleanProperty`	Returns a Boolean that gives the value of the property.	`key` (required) - the name of the property `id` (optional) `scope` (optional) `pflist` (optional)
`getCollectionProperty`	Returns a Java Map with the collection property. Here, a collection refers to a multi-value property. Depending on the context, it is either the analogue of Java Maps or Lists. For Lists, the returned Java Map object contains key-value pairs where the key equals the value. This is an overloaded method that can return alternately the default or localized version of the collection property.	`key` (required) - the name of the property `id` (optional) `scope` (optional) `pflist` (optional)
`getIntegerProperty`	Returns an integer with the integer property. This method returns a default value if the property does not exist.	`key` (required) - the name of the property `id` (optional) `scope` (optional) `pflist` (optional)
`getProperty`	Returns a Java Object with a property. The value returned from this method is a Java `Object` of type String, Integer, Boolean, or Map. If the property does not exist, then the default value is returned.	`key` (required) - the name of the property `id` (optional) `scope` (optional)
`setStringProperty`	Sets a string property.	`key` (required) - the name of the property `value` (required) - the value of the property to be set `pflist` (optional)

Table D-3 desktopProviderContext.tld Normal Tags with Attributes *(Continued)*

Tag Name	Description	Attributes/Descriptions
`setBooleanProperty`	Sets a Boolean property.	`key` (required) - the name of the property `value` (required) - the value of the property to be set `pflist` (optional)
`setCollectionProperty`	Sets a collection property.	`key` (required) - the name of the property `value` (required) - the value of the property to be set `pflist` (optional)
`setIntegerProperty`	Sets an integer property.	`key` (required) - the name of the property `value` (required) - the value of the property to be set `pflist` (optional)
`getLocalePropertiesFilters`		`id` (required) `scope` (optional)
`getClientPropertiesFilters`		`id` (required) `scope` (optional)
`getClientAndLocalePropertiesFilters`		`id` (required) `scope` (optional)
`getClassName`	Returns a string with the class name for the provider class that this object is providing an environment for. The class name returned must implement the provider interface. This method is used to construct the provider object. It is used by container channels.	`id` (optional) `scope` (optional)
`getTemplate`	Returns a string buffer with the desktop template. The actual template buffer returned is based on the Desktop type, locale, channel, client type, and the template name.	`file` (required) - name of template to return `table` (optional) - hashtable - tag table used for tag swapping `id` (optional) `scope` (optional)

Table D-3 desktopProviderContext.tld Normal Tags with Attributes *(Continued)*

Tag Name	Description	Attributes/Descriptions
`getDesktopURL`	Returns a string with the Desktop URL. The Desktop URL is the absolute URL used to access the Desktop application. For example: `http://`*server:port*`/portal/dt`. The request object parameter is included to facilitate implementations. It may be used to build the Desktop URL by supplying the server, port, and protocol of the request. It is not required that the request object be utilized to generate the Desktop URL.	`querymap` (optional) `querystring` (optional) `pathinfo` (optional) `escape` (optional) `id` (optional) `scope` (optional)
`getDesktopType`	Returns a string with the Desktop type. The Desktop type, also known as template type, is a string that is one of several indexes used to lookup Desktop templates and JSP files. The Desktop type is typically used to group Desktop customization files to provide different themes.	`id` (optional) `scope` (optional)
`getLocaleString`	Returns a string representation of the locale.	`id` (optional) `scope` (optional)
`getLocale`	Returns Java Locale object representation of the locale.	`id` (optional) `scope` (optional)
`getLogoutURL`	Returns a string with the logout URL. The result of making a connection to the logout URL is typically the termination of the user's session. What actually happens is dependent on the application receiving the URL connection. Providers may use this value to generate links that allow the user to end their session.	`id` (optional) `scope` (optional)
`getStringAttribute`	Returns a string with the value of the string attribute or null if the attribute is not found. Attributes are settings that are not channel-specific. An example of an attribute might be the user's first and last name. Channel-specific settings are called properties. Properties can be retrieved by calling the `get*Property()` methods. Whether a particular value is considered a property or an attribute depends on the underlying implementation of `ProviderContext`.	`key` (required) - the name of the attribute `id` (optional) `scope` (optional)
`setStringAttribute`	Sets a string attribute. Attributes are settings that are not channel-specific. An example of an attribute might be the user's first and last name. Channel-specific settings are called properties. Properties can be set by calling the `set*Property()` methods. Whether a particular value is considered a property or an attribute depends on the underlying implementation of `ProviderContext`.	`key` (required)- the name of the attribute `value` (required) - the value of the attribute

Table D-3 desktopProviderContext.tld Normal Tags with Attributes *(Continued)*

Tag Name	Description	Attributes/Descriptions
`getClientTypeProperty`		`key` (required)
		`clientType` (optional)
		`id` (optional)
		`scope` (optional)
`getClientType`	Returns a string with the client type. There is no requirement as to how the client type is determined. It may be hardcoded, derived from the session, or otherwise.	`id` (optional)
		`scope` (optional)
`getDefaultClientType`	Returns a string with the default client type.	`id` (optional)
		`scope` (optional)
`getCharset`	Returns a string with the character set. The character set is used for decoding input and encoding output.	`id` (optional)
		`scope` (optional)
`getClientPath`	Returns a string with the client path. The client path is one of several components used to lookup Desktop templates and JSPs.This allows the lookup to be client-specific.	`id` (optional)
		`scope` (optional)
`getContentType`	Returns a string with the content type. This value is used to determine if a provider is able to produce content for the client's device.	`id` (optional)
		`scope` (optional)
`getSessionID`	Returns a string with the unique session identifier. The format of the return value is implementation specific. The only guarantee is that it is unique (each user session has a unique session ID).	`id` (optional)
		`scope` (optional)
`getUserID`	Returns a string with the user identifier. The format of the return value is implementation specific. There is no guarantee that this value is unique (there may be multiple user sessions for a given user identifier).	`id` (optional)
		`scope` (optional)
`setClientProperty`	Sets a client property.	`name` (required) - the name of the property
		`value` (required) - the value of the property to be set
`getClientProperty`	Returns a string with the client property.	`name` (required)
		`id` (optional)
		`scope` (optional)
`isLogMessageEnabled`	Returns a Boolean; `true` if the log level is set to message or higher; otherwise `false`.	`id` (optional)
		`scope` (optional)

Table D-3 desktopProviderContext.tld Normal Tags with Attributes *(Continued)*

Tag Name	Description	Attributes/Descriptions
isLogWarningEnabled	Returns a Boolean; true if the log level is set to warning or higher; otherwise false.	id (optional) scope (optional)
logError	Logs a message (any Java Object) if the logging level is error. The location to store logging messages is implementation dependent.	value (required) - message to log throwable (optional)
logMessage	Logs a message (any Java Object) if the logging level is message or higher. The location to store logging messages is implementation dependent.	value (required) - message to log throwable (optional)
logWarning	Logs a message (any Java Object) if the logging level is warning or higher. The location to store logging messages is implementation dependent.	value (required) - message to log throwable (optional)
getDefaultChannelName	Returns a string with the default channel name.	id (optional) scope (optional)
getTopChannelName	Returns the top channel name for the current request.	id (optional) scope (optional)
getStaticContentPath	Gets the URI prefix to web server static content.	id (optional) scope (optional)
getProviderVersion	Get the version of the provider schema for the current channel.	id (optional) scope (optional)
encodeURLParameter	URL encodes a unicode string.	id (optional) scope (optional) key (required)
decodeURLParameter	Decodes the URL encoded Unicode string. This tag just returns back the original string passed in.	id (optional) scope (optional) key (required)
encodeURL	Encodes a URL. Rewrites the URL to include the session id.	url (required) id (optional) scope (optional)
escape	Escapes a String using an encoder class that encodes a specific type of markup. This tag is used to allow provider code to encode content in a device-unaware manner.	id (optional) scope (optional) unescaped (required)

The normal tag in the desktopContainerProviderContext.tld file has a bodycontent of empty.

Table D-4 desktopContainerProviderContext.tld Normal Tags with Attributes

Tag Name	Description	Attributes/Descriptions
getContent	Returns a string buffer with the content of the named channel. This method is provided for convenience. It gets the provider object for the named channel and calls Provider.getContent().	channel (required) - the name of the channel id (optional) scope (optional)

All of the normal tags in the desktopTab.tld file have a bodycontent of empty.

Table D-5 desktopTab.tld Normal Tags with Attributes

Tag Name	Description	Attributes/Descriptions
getAvailableTabs	Returns a list of available tabs. The list returned is a Collection of Strings. Each of the Strings is the name of an Unmodifiable Tab that is available.	id (optional) scope (optional)
getSelectedTabs	Returns the list of selected tabs. The list returned is a Collection of Strings. Each of the Strings is the name of an Unmodifiable Tab that is selected.	id (optional) scope (optional)
getSelectedTab	Returns the selected tab, the current selected Unmodifiable Tab in the user's session.	id (optional) scope (optional)
getMakeTab	Returns the make tab, the tab spec to be used for "Make My Own tab" creation by the user.	id (optional) scope (optional)
getStartTabNam	Returns a string with the start tab Name, the name of the tab to be displayed when the user logs in.	id (optional) scope (optional)
getSelectedTabName	Returns a string with the selected tab Name, the current selected tab in the user's session.	id (optional) scope (optional)
getTabURL	Returns the Tab URL. This method gets the tab URL used to switch the selected tab on the user's desktop.	id (optional) scope (optional)
getName	Returns a string with the name of the tab.	id (optional) scope (optional)
getDesc	Returns a string with the description of the tab.	id (optional) scope (optional)

Table D-5 desktopTab.tld Normal Tags with Attributes *(Continued)*

Tag Name	Description	Attributes/Descriptions
getDisplayName	Returns a string with the display name of the tab.	id (optional) scope (optional)
getEncodedName	Returns a string with the HTML encoded name of the tab.	id (optional) scope (optional)
isPredefined	Determines whether the tab is predefined or not and returns a boolean that gives the predefined status of a tab.	id (optional) scope (optional)
isRemovable	Returns a Boolean that gives the removable status of the tab. Returns true if the tab is removable; otherwise false.	id (optional) scope (optional)
isRenamable	Returns a Boolean that gives the renamable status of the tab. Returns true if the tab is renamable; otherwise false.	id (optional) scope (optional)

All of the normal tags in the desktopTable.tld file have a bodycontent of empty.

Table D-6 desktopTable.tld Normal Tags with Attributes

Tag Name	Description	Attributes/Descriptions
getColumns	Returns a list of channel names that belong in that particular column.	column (required) id (required)
getColumnWidth	Returns the width of a column (a percentage with respect to the entire Desktop). Valid columns are left, center, and right.	column (required) id (optional)
getHasFrame	Returns a Boolean that gives the frame status of the channel. Returns true if the channel has a frame; otherwise false.	id (optional) scope (optional)
getIsMinimized	Returns a Boolean that gives the minimized status of the channel. Returns true if the channel is minimized; otherwise false.	id (optional) scope (optional)
getIsMovable	Determines whether the channel is movable or not and returns a Boolean that gives the movable status of a channel.	id (optional) scope (optional)

Table D-6 desktopTable.tld Normal Tags with Attributes *(Continued)*

Tag Name	Description	Attributes/Descriptions
`getProviderCommand`	Gets the HTML code needed to display the provider commands (minimize channel, help screen, edit channel, and so forth). The commands are put in a Map. The keys for the Map are `minMaximizedCommand`, `helpCommand`, `editCommand`, `detachAttachCommand`, and `removeCommand`.	`id` (optional) `scope` (optional)
`getIsDetached`	Returns a Boolean that gives the detached status of the channel. Returns `true` if the channel is detached; otherwise `false`.	`id` (optional) `scope` (optional)
`getDetached`	Returns the detached channels list. The list returned is a Collection of Strings. Each of the Strings is the name of a channel. The channels returned are not necessary channels that have been detached from the desktop. The tag `getIsDetached` should be used to verify that a channel has been detached.	`id` (required) `scope` (optional)
`getWindowName`	Returns a string with the window name for the detached window when a channel is detached.	`id` (optional) `scope` (optional)
`isPopup`	Determines whether the page is being drawn for a popup channel. Returns true if the table container action is a popup and returns false otherwise.	id (optional) scope (optional)
`getPopupWindowWidth`	Returns an integer with the popup window width for the detached window when a channel is detached.	`id` (optional) `scope` (optional)
`getPopupWindowHeight`	Returns an integer with the popup window height for the detached window when a channel is detached.	`id` (optional) `scope` (optional)
`getContents`	Returns a string buffer with the contents of all non-minimized and selected channels of the table container. The contents are put in a Map with the channel name as the key.	`id` (required) `scope` (optional)

All of the normal tags in the `desktopTheme.tld` file have a bodycontent of empty.

Table D-7 desktopTheme.tld Normal Tags with Attributes

Tag Name	Description	Attributes/Descriptions
`getGlobalThemes`	Returns the list of globally defined themes. The list returned is a Collection of Strings. Each of the Strings is the name of a globally defined theme.	`id` (required) `scope` (optional)

Table D-7 desktopTheme.tld Normal Tags with Attributes *(Continued)*

Tag Name	Description	Attributes/Descriptions
`getSelectedName`	Returns the name of the selected theme. The name returned can be the name of one of the globally defined themes or `CustomTheme`. If `CustomTheme` is returned, this means the user has defined and is using the custom defined theme.	`id` (optional) `scope` (optional)
`setSelectedName`	Sets a theme for the current user. The theme to be set can be one of the globally defined themes or a `CustomTheme`.	`value` (required) - the name of the theme set
`getAttribute`	Returns the value of a theme attribute. Get a specific attribute value of a specific theme.	`name` (required) - the name of the attribute. Possible values are: bgColor, borderColor, titleBarColor, fontColor, borderWidth and fontFace `theme` (optional) - the name of the theme. If this is not specified, the currently selected theme is used. `requestOverride` (optional) true or false - Whether to use value in the request to override the theme value. If this is not specified, false is assumed. This is useful in the preview case. `id` (optional) `scope` (optional) `default` - the default value. If `getAttribute()` of a theme returns null, and a default value is defined in the tag, then return the default value.
`setCustomAttribute`	Sets a customized value in the CustomTheme	`name` (required) - the name of the attribute. Possible values are: bgColor, borderColor, titleBarColor, fontColor, borderWidth and fontFace `value` (required) - the value to be set

Search Tags

The Search tag library contains tag wrappers for the SearchContext Java API. SearchContext is an extension of the Search API with convenient methods for advanced search and search result status. The Search tag library can be divided into various categories depending upon where the tags should be used.

Table Table D-8 on page 327 through Table D-12 on page 331 describe the normal tags in the Search tag library. Each table has three columns: the first column lists the tag name, the second a description of the tag's function, and the third column lists that tag's attributes with brief comments. The tables are:

- Table D-8 on page 327 - searchContext tag
- Table D-9 on page 327 - tags used before a search
- Table D-10 on page 330 - executeSearch tag
- Table D-11 on page 330 - tags related to Search results
- Table D-12 on page 331 - miscellaneous tags

The `search.tld` does not have any context setup or validator tags. All of the normal tags in the `search.tld` file have a bodycontent of empty except `searchContext` and `SOIF`.

A number of attributes have a value of true for `rtexprvalue`, which means the value for this attribute can be obtained at run time or it can be hardcoded. These attributes are listed as dynamic attributes.

Table D-8 `searchContext` Tag

Tag Name	Description	Attributes/Descriptions

`searchContext`	Main outer tag that encloses all other tags.	`bodycontent` - JSP
		`rdmServer` (optional) - `search server` - `http://.../portal/search` or `https://../portal/search`
		`rdmType` (optional) - `rd-request` (default), `taxonomy-request`, `schema-request`, `server-request` or `status-request`
		`ql` (optional) - query language
		`query` (optional) - dynamic attribute

The tags in Table D-9 are normally used before executing a search. Before a successful search can be executed these tags must set a value:

- `setRDMServer`
- `setRDMType`
- `setQuery or setCriteria`

Table D-9 Pre-Search Tags in `search.tld`

Tag Name	Description	Attributes/Descriptions
`setRDMServer`	Sets the RDMServer variable. The server URL should be set explicitly. Format: http:// or `https://`*hostname*`:`*port*`/portal/search` Value has to be set.	`rdmServer` (required) - server URL; can be an expression or a hardcoded string value.

Table D-9 Pre-Search Tags in `search.tld` *(Continued)*

Tag Name	Description	Attributes/Descriptions
`setRDMType`	Sets the RDM Request type. RDMType - Can be one of: rd-request: The default request type. Resource descriptions (documents). taxonomy-request: Taxonomy. schema-request: The schema. server-request: Server information. status-request: Server status information. Set value explicitly; the default set by the system may not return the expected results.	`rdmType` (required) - string
`setViewAttributes`	Sets the SOIF attributes that are returned for the search. This is an optional tag. It assumes the default values if not set explicitly.	`viewAttributes` (required) - string: null (all) or comma delimited list of attributes.
`setSessionID`	The Search Server needs to validate the user's identity for document level security. This tag is a wrapper for `setSessionID(String)` method in `SearchContext`. The JSP gets the portal access Token string and passes it along to the search server using this tag.	`sessionID` (required) - string
`setViewHits`	Sets the maximum number of hits returned.	`viewHits` (required) - integer - dynamic attribute
`setViewOrder`	Sets the sorting order for results.	`viewOrder` (required) - string: null or comma delimited list of attributes each proceeded with + for ascending order or - for descending order.
`setCategory`	Sets the category id. It is mainly required for browsing and category searches. The category is appended to the query string based on the RDMType and query language in the `executeSearch` tag.	`category` (required) - string: current category level; if not set, the root of the taxonomy tree is used and the search is executed for all categories.
`setFirstHit`	`setFirstHit` takes a string input. Sets the starting hit for search results. In other words, start from 1, start from 11 and so forth. It corresponds to the `setFirstHit()` method in the `Search` API. Results are returned from this hit. This is an optional tag. Alternately you can use `setPage` and `setViewHits`.	`fromHit` (required) - integer

Table D-9 Pre-Search Tags in `search.tld` *(Continued)*

Tag Name	Description	Attributes/Descriptions
`setQuery`	Query string. Either this value has to be set or else `setCriteria` has to be set.	`query` (required) - string - dynamic attribute
`setDatabase`	Has a string attribute. Default value is '', which means use berkeley db. Database should be specified if berkeley db should not be used.	`database` (required) - `string` - dynamic attribute
`setPage`	Sets the current page value. The page value is used to calculate the `firstHit` and `totalPages`.	`page` (required) - integer - dynamic attribute
`setSearchAll`	Set `searchAllCategories` or not. If the search is within a particular category then set it to `false`; else set it to `true`. Default is `true`.	`searchAll` (required) - Boolean
`setQueryLanguage`	Sets the query language. ql - Can be one of: search: The default Search query language. Searches documents or the taxonomy. taxonomy-basic: Used for requesting branches or parts of the taxonomy. schema-basic: Queries the Compass schema. url: Retrieves RDs by URL (`scope=url`).	`ql` (required) - string
`setCriteria`	Sets the query string in a list format. Useful in advanced search. The list should have an operand, operator and a value. The list of valid operator's are defined in the `SearchContext` API. The operand can be a schema field. The user can bypass this tag and just use the `setQuery` tag directly by converting a complex query into the syntax that the search engine requires. The `setCriteria` tag is a wrapper for the `setScope(list)` method in the `searchContext` API. This method basically parses the list and converts it into a string. For example, author CONTAINS xyz. This produces a query. Either this value has to be set or else `setQuery` has to be set.	`criteria` (required) - string - dynamic attribute

The executeSearch tag in Table D-10 on page 330 executes the search.

Table D-10 `executeSearch` Tag

Tag Name	Description	Attributes/Descriptions
`executeSearch`	Tag for the execute method in `SearchContext` API. Executes search after doing some validation of the search parameters and query string.	none

The tags in Table D-11 are related to search results and are used after a search is executed. They provide various counts and help display the search results.

Table D-11 Post Search Tags in `search.tld`

Tag Name	Description	Attributes/Descriptions
`SOIF`	Wrapper for `SOIF` API.	`bodycontent` - JSP `input` (required) -SOIF document
`getSoifResult`	Returns SOIF type object.	`id` (optional) `scope` (optional)
`getValue`	Returns a string value of the attribute or returns a string value of a multivalue attribute with index. Wrapper for `SOIF` API. This tag must be used within the `SOIF` tag.	`soifAttribute` (required) - string - attribute name `escape` (optional) `id` (optional) `scope` (optional) `truncate` (optional) - specifies the number of characters to display in the title
`getURL`	Returns a string with the SOIF URL. Wrapper for `SOIF` API. This tag must be used within the `SOIF` tag.	`escape` (optional) `id` (optional)
`getHasNextPage`	Returns `true` if there are more hits for the next page by considering the values of `viewHits` and the current page.	`id` (optional) `scope` (optional)
`getHasPreviousPage`	Returns `true` if there is a previous page. Value based on `viewHits` and current page value.	`id` (optional) `scope` (optional)
`getNoHits`	Returns `true` if no matching hits were found. This is a convenience tag.	`id` (optional) `scope` (optional)

Table D-11 Post Search Tags in `search.tld` *(Continued)*

Tag Name	Description	Attributes/Descriptions
`getHitCount`	Returns the total number of results that matched the query.	`id` (optional) `scope` (optional)
`getToHit`	Returns the last hit being displayed on a page. The value is based on `firstHit` and `viewHits`.	`id` (optional) `scope` (optional)
`getPage`	Returns the current page. If the value is not set, calculates the page based on `viewHits` and `firstHit`.	`id` (optional) `scope` (optional)
`getTotalDocuments`	Returns the total number of documents in the database.	`id` (optional) `scope` (optional)
`getTotalPages`	Returns the total number of pages of hits that are available. The value is calculated from `viewHits` and `hitCount`.	`id` (optional) `scope` (optional)
`getResultCount`	Returns the number of results returned by the search. Returns -1 for an error.	`id` (optional) `scope` (optional)

The tags in Table D-12 are used for debugging.

Table D-12 Miscellaneous Tags in `search.tld`

Tag Name	Description	Attributes
`getSearchString`	Returns a string with a the search query being executed. Good for debugging.	`id` (optional) `scope` (optional)
`getCategory`	Returns a string with the current category name or else set to root.	`id` (optional) `scope` (optional)
`getFirstHit`	Returns the starting hit being displayed.	`id` (optional) `scope` (optional)
`getSessionID`	Returns a string with a user's session id. It has no value unless previously set.	`id` (optional) `scope` (optional)
`getQuery`	Returns the query string; returns an empty string if not set.	`id` (optional) `scope` (optional)
`getViewHits`	Returns `viewHits`, an integer that defines the maximum number of hits returned (range 0-100). The default value is 8 if not set or set outside of range.	`id` (optional) `scope` (optional)

Search Exceptions

When you are developing JSPs or doing Search administration tasks, you may see the following exceptions:

Taglib Use Errors:

- Invalid Tag Sequence: Must be within a searchContext Tag
- Variable Undefined in Page Context
- Invalid nesting of context
- Error in creating search context

Search Request Errors:

- Search server is not defined
- View Hits cannot be 0
- RDM Type must be defined
- Query Language must be defined

TIP When debugging, if you receive unusual exceptions, check that the search server is set and its format is correct and has no typographical errors.

Example of a Search JSP

Code Example D-6 Sample Search JSP

```
<%--
  Copyright 2002 Sun Microsystems, Inc.  All rights reserved.
  PROPRIETARY/CONFIDENTIAL.  Use of this product is subject to license terms.
--%>

<!-- @(#)searchContent.jsp -->

<%@page language= "java" import="com.sun.portal.search.providers.*, java.lang.*,
java.util.*, java.text.*" %>
<%@page errorPage="error.jsp" %>

<%@ taglib uri="/tld/search.tld" prefix="search" %>
<%@ taglib uri="/tld/desktop.tld" prefix="dt" %>
<%@ taglib uri="/tld/desktopProviderContext.tld" prefix="dtpc" %>

<B>Test 1: </B><BR>
```

Code Example D-6 Sample Search JSP *(Continued)*

```
<dt:obtainChannel channel="$JSPProvider">
<dtpc:providerContext>

    <search:searchContext>
    <%--<search:setQueryLanguage ql= "search"/> --%>
    <search:setQuery query= "java"/>
    <%--<search:setRDMType rdmType= "rd-request"/>--%>
    <search:setRDMServer rdmServer= "http://milonga1.sesta.com:80/portal/search"/>
    <search:setViewAttributes viewAttributes=
"url,title,description,score,content-length,classification"/>
    <search:setViewHits viewHits= "10"/>
    <search:setPage page= "1"/>

    <dtpc:getSessionID id = "accessToken"/>
    <search:setSessionID sessionID="$accessToken"/>

    <search:executeSearch/>

    <UL>
    <LI><search:getSearchString/>
    <LI>Query = <search:getQuery/>
    <LI>resultCount = <search:getResultCount/>
    <LI>hitCount = <search:getHitCount/>
    <LI>total Documents = <search:getTotalDocuments/>
    <LI>total Pages = <search:getTotalPages/>
    <LI>hasnoHits= <search:getNoHits/></UL>

    </search:searchContext>
</dtpc:providerContext>
</dt:obtainChannel>

<B>Test 2: </B><BR>
<search:searchContext query="*" rdmType="rd-request" ql="search"
rdmServer="http://milonga1.sesta.com:80/portal/search">
    <search:setViewHits viewHits= "10"/>
    <search:executeSearch/>
    <UL><LI>Query = <search:getQuery/>
    <LI>resultCount = <search:getResultCount/></UL>
</search:searchContext>

<B>Test 3: </B><BR>
<search:searchContext query="*" rdmType="rd-request" ql="search"
rdmServer="http://milonga1.sesta.com:80/portal/search">
    <search:setFirstHit firstHit= "21"/>
    <search:setViewHits viewHits= "10"/>
    <search:executeSearch/>
    <UL><LI>Query = <search:getQuery/>
    <LI>Page = <search:getPage/>
    <LI>resultCount = <search:getResultCount/></UL>
</search:searchContext>
```

Instant Messaging Tags

The IMProvider content page uses a custom tag library defined in a file called `im.tld` which is installed into the `/etc/opt/SUNWps/desktop/default/tld` directory. The `im.tld` file defines the following tags:

Table D-13 Tags in `im.tld`

Tag Name	Description	Attributes
`getContactGroups`	Returns list of contact groups that the user has defined. The list is returned as a Collection of Strings where each String is the display name for a contact group.	`id` (optional) `scope` (optional)
`getContactGroup`	Returns the list of contacts in the named contact group. The list is returned as a Collection of internal objects that can be passed to the `obtainContact` tag.	`group` (required) - The name of the contact group. `id` (optional) `scope` (optional)
`getUsername`	Returns the instant messaging username for the user.	`id` (optional) `scope` (optional)
`getToken`	Returns the login token for the user (either an Identity Server software SSOToken or the user's password.)	`id` (optional) `scope` (optional)
`obtainContact`	A context setup tag that is used to obtain the presence information for the indicated contact. The remaining tags can be used inside this tag.	`contact` (required) - The internal identifier of the contact, typically obtained from the `getContactGroup` tag.
`getContactPresence`	Returns the current presence status for the contact. The status can be: AWAY, BUSY, CLOSED, FORWARDED, IDLE, OPEN, OTHER. These are from the PresenceSession class in the instant messaging API.	`id` (optional) `scope` (optional)
`getContactName`	Return the common name for the contact.	`id` (optional) `scope` (optional)
`getContactUsername`	Returns the instant messaging user name for the contact.	`id` (optional) `scope` (optional)
`getDateTime`	Returns the update time.	`format` (required) `id` (optional) `scope` - (optional)
`isSecureMode`	Returns boolean indicating whether the channel is being accessed via the Secure Remote Access Gateway component and the Netlet is loaded.	`id` (optional) `scope` (optional)

Table D-13 Tags in `im.tld` *(Continued)*

Tag Name	Description	Attributes
`getCodebase`	Returns the codebase to use to download the applet. This takes into account whether the channel is being accessed via the Secure Remote Access Gateway component and the Netlet is loaded.	`id` (optional) `scope` (optional)
`getIMServer`	Returns the `-server` argument to pass to the IM client. This takes into account whether the channel is being accessed via the Secure Remote Access Gateway component and the Netlet is loaded.	`id` (optional) `scope` (optional)

The entire interface to the Instant Messaging server APIs is in the `getContactGroup` tag. This tag will fetch all of the presence information and cache it in the request. The remaining tags will simply fetch the information out of the cache.

Glossary

Refer to the Java Enterprise System Glossary (http://docs/sun.com/doc/816-6873) for a complete list of terms that are used in this documentation set.

Index

A

Additional product documents 16
AddressBookProvider / channel
  Display Profile properties 61, 62
  tags 229
  template files 200, 201
Administrator customizations 19
Anonymous Desktop
  accessing 132–134
  adding the Login channel 136–139
  changing the theme 172
  configuring to always use anonymous log in 134
  customization 129–140
  disabling authlessanonymous log in 131
  disabling log in 130
  enabling authlessanonymous log in 130
  enabling log in 130
  JSPs 251–257
  modifying the banner and meny bar 135
  user session 129
APIs
  Identity Server software 19
  search service 19, 326
  SearchContext 326
Application Programming Interface (*see* APIs)
AppProvider / channel
  Display Profile properties 62, 63
  tags 229
  template file 201
Attributes
  DesktopType 29, 86
  dynamic attributes 326
  global 47
  JSP tag library 310
Authentication
  configuring anonymous authentication 129
  configuring LDAP authentication for UserInfoProvider 142
  configuring to always use anonymous log in 134
  customization 141
  disabling authentication-less log In 131
  enabling authentication-less login 130
  LDAP 142
  modules,implementing 19
  screen customization 20
  using UNIX with LoginProvider 141
Authlessanonymous
  accessing the Desktop through the Identity Server host name 132
  disabling log in 131
  enabling log in 130
  modifying the default Desktop for the user 140
Available list
  categorizing 147
  for container 49

B

Banner
- changing the header image 151
- modifying anonymous Desktop banner 135

Bold monospaced font usage 14

BookmarkProvider / channel
- Display Profile properties 63
- tags 229
- template files 201, 202

Boolean property 45

Border,channel 121

Building-block providers
- container providers
  - JSPSingleContainerProvider 48
  - JSPTabContainerProvider 48
  - JSPTableContainerProvider 48
  - list of 25
- introduction 37
- leaf provider
  - Display Profile properties 54
  - JSPProvider 55
  - list of 25
  - URLScraperProvider 55
  - XMLProvider 56

Bullet graphics 170

Business-to-business portal 11

Business-to-consumer portal 11

Business-to-employee portal 11

Buttons,channel 119

C

Caching
- container 116
- JavaServer Pages (JSPs) 298
- provider 116

CalendarProvider / channel
- Display Profile properties 64
- tags 231–240
- template files 202–204

Channel 170
- accessing directly 180
- changing the border width 121
- changing the layout for a table container 120
- controlling and configuring container caching 116
- customizing refresh times 115
- customizing the border 122
- customizing window preference 116
- Display Profile
  - definition 23, 24
  - object 23, 24
  - properties 23, 45, 82
- dynamic content 20
- removing a button 117–119
- removing a title bar 120
- widths 145

Collection property 46

Colors,Desktop
- changing 155
- changing the default scheme 157

Column
- modifying the layout 148
- modifying the width 149

Conditional properties 46

Container
- channel 25
- contained 103
- creating 39
- deploying JSPs and template files 39
- Display Profile
  - available and selected list 49
  - object 25
  - properties 45, 48–54
- dynamic content 20
- how-to stretch a tab 110
- modifying the default Desktop for authlessanonymous user 140
- removing a button from all channels 117

Content page,Desktop
- changing the header and footer 152
- header and footer files 153

Content providers
- AddressBookProvider 61, 200
- AppProvider 62, 201
- BookmarkProvider 63, 201
- CalendarProvider 63, 202
- definition 37

Display Profile properties 61–78
IMProvider 64
LoginProvider 65, 205
LotusNotesAddressBookProvider 66, 205
LotusNotesCalendarProvider 67, 206
LotusNotesMailProvider 68, 208
MailCheckProvider 69, 209
MailProvider 70, 209
MSExchangeAddressBookProvider 71, 210
MSExchangeCalendarProvider 72, 211
MSExchangeMailProvider 73, 213
NotesProvider 74
SimpleWebServiceConfigurableProvider 76
SimpleWebServiceProvider 75
UserInfoProvider 77
Content,dynamic 20
Content / layout bar 170
Context setup tags 312
Customization
anonymous Desktop 129
authentication 141
container 105
Desktop
areas of 20
by administrator 19
by developer 18
by end user 18
channels 115
introduction 17
tabs 105
Desktop layout 145
end-user online help 191
GlobalThemes 159
instant messaging 123
search channel 20
service provider 175

D

Data,integrated 11
Debug file,Desktop 41
`default` directory 257–280
JSP files 257–280
template files 217, 218
Desktop
adding new layouts 148
administrator customizations 19
base templates 20
branding 151
changing column layout 148
changing content layout 147
changing the colors 155
content page customization 152
creating 37
customization introduction 17
debugging 41
deriving more layouts 145
deriving the 84
determine 84
developer customizations 18
end user customizations 18
end-user online help customization 191
Frame-based 97–101
GlobalThemes (*see* GlobalThemes)
how-to access 41
JavaServer Pages (*see* JSPs,Desktop)
JSP-based 88–96
layout page customization 152
modifications to layout 145–150
samples 36, 89, 93, 97
tag library
hierarchy 303
JavaServer Pages 301–335
template 227–250
template files
in `default` directory 200–218
in `sampleportal` directory 218–224
introduction 31
overview 199
template files (*see* Templates)
DesktopType attribute
changing 86
purpose 29
Developer customizations 18, 19
DiscussionProvider / channel
channel customization 187
customization overview 177
discussion threads 178
DiscussionLite channel
customization 187
customize link display window 187

displaying on the front tab 188
introduction 179
JSPs 258
discussions 178
Discussions channel
accessing directly 181
controlling access to discussions 190
customization 189
displaying additional fields in the list view 189
inheriting classification and readACL 190
introduction 179
JSPs 259
modifying the Sort Order 189
modifying `viewHits` 190
Display Profile properties 59, 60
displaying DiscussionLite on the front tab 188
full view 178
JavaServer Pages
files 261
introduction 179
lite view 178
rating discussions 178
searching discussions 178
tag libraries
introduction 179
Display Profile
AddressBookProvider / channel properties 61
AppProvider / channel properties 62
BookmarkProvider / channel properties 63
CalendarProvider / channel properties 63
channel
properties 26, 82
channel properties 45
container
available and selected list 49
properties 45, 48, 49
definition
channel 23
GlobalThemes 160
introduction 21
provider 26
sample portal 43
DiscussionProvider / channel properties 59
document
format 24
merging 21, 23
priority 45
sample merge process 22
storing 21
structure 21
Document Type Definition (*see* DTD,Display Profile)
editing 38
global properties 44, 46
GlobalThemes properties 167
IMProvider / channel properties 65
introduction 21
JSProvider / channel properties 55
loading 40
LoginProvider / channel properties 66
logo image properties 151
LotusNotesAddressBookProvider / channel properties 66
LotusNotesCalendarProvider / channel properties 67
LotusNotesMailProvider / channel properties 68
MailCheckProvider / channel properties 69
MailProvider / channel properties 70
MSExchangeAddressBookProvider / channel properties 71
MSExchangeCalendarProvider / channel properties 72
MSExchangeMailProvider / channel properties 73
NotesProvider / channel properties 74
objects
channel object 24
container object 25
final values 23
grouping 24
modifying guidelines 27
property object 23
provider object 23, 26
properties
AddressBookProvider / channel 61
AppProvider / channel 62
BookmarkProvider / channel 63
CalendarProvider / channel 64
channel 45
container 45, 48, 49
content providers 61–78
DiscussionProvider / channel 59
global 44, 46
GlobalThemes 167

IMProvider/channel 64
JSPProvider/channel 55
leaf providers 78
LoginProvider/channel 66
LotusNotesAddressBookProvider/channel 66
LotusNotesCalendarProvider/channel 67
LotusNotesMailProvider/channel 68
MailCheckProvider/channel 69
MailProvider/channel 70
MSExchangeAddressBookProvider/channel 71
MSExchangeCalendarProvider/channel 72
MSExchangeMailProvider/channel 73
NotesProvider/channel 74
overview 43
provider 44
SearchProvider/channel 58
service providers 57–60
SimpleWebServiceConfigurableProvider/channel 76
SimpleWebServiceProvider/channel 75
tab container 53
table container 50
types 45
URLScraperProvider/channel 55
UserInfoProvider/channel 77
XMLProvider/channel 56
provider
definition 26
properties 26, 44
SearchProvider/channel properties 58
SimpleWebServiceConfigurableProvider/channel properties 76
SimpleWebServiceProvider/channel properties 75
tab container properties 53
table container properties 50
URLScraperProvider/channel properties 55
UserInfoProvider/channel properties 77
XML files 43
XMLProvider/channel properties 56
Document Type Definition (*see* DTD,Display Profile)
Document,WSDL 75
Documents,Display Profile
merging 21, 23
priority 45
sample merge process 22
storing 21
structure 21, 24
`dpadmin` command
usage guidelines 28
`dp-anon.xml` 44, 88
`dp-org.xml` 43, 87, 88
`dp-providers.xml` 43, 87
DTD,Display Profile 21, 22
DummyChannel JSPs 262

E

End user customizations 18
Errors
JSP compilation and runtime 39
template files 204
Example
Containers for the sample portal Desktop 84
Desktop based on FrameTabContainer 97
Desktop based on JSPTabContainer 89
Desktop based on JSPTableContainer 93
Display Profile document merge process 22
NotesProvider/channel text file 74
tag usage in JSPs 309
Exception,JSP 39

F

File
Desktop color files 156
Desktop debug 41
Desktop online Help 193
Desktop template 199
header and footer files for Theme, Content, and Layout pages 153
Font
changing families and sizes 171
Footer 170
FrameTabContainer
architecture 100
based contained containers 103

default actions 98
default Display Profile settings 99
default layout 97
Display Profile properties 99, 100
introduction 84
JSP files 101
JSPs 281
JSPs for anonymous Desktop 252
sample Desktop 97

G

Global properties
attribute
`docroot` 48
`GlobalThemes` 47
`helpURL` 48
`locales` 48
`userDefinedChannels` 48
`UserTheme` 48
introduction 44
sample,Display Profile 46
GlobalThemes
adding and customizing 170
changing font sizes and font families 171
changing the anonymous Desktop theme 172
changing the color 297
changing the page header and footer 152
customization 159–174
customization overview 159
Desktop page 54
Display Profile definition 160
Display Profile properties 167, 167–169
glossary of terms 170
header and footer files 153
logo image 151
overview 159
Glossary
GlobalThemes terms 170

H

Header,description of 170
Help,Desktop online
customization overview 191
HTML file
customization 195–197
listing 193
Highlight color for channel content 170
How-to
access the anonymous Desktop through the Identity Server host name 132
access the anonymous Desktop through the Portal Server host name 133
add a Channel to a User-defined tab 114
add a role-based tab 112
add a tab to JSPTabContainer 105–109
add a theme to sample portal 170
add new layouts 148
add the Login channel to the anonymous Desktop 136–139
change the anonymous Desktop theme 172
change the background color of tabs 111
change the channel border width 121
change the channel layout for a table container 120
change the colors of a theme 297
change the default color scheme for an organization 157
change the Desktop colors 155
change the Desktop type 86
change the font families and font sizes 171
change the header and footer in the Theme, Content, and Layout pages 152
change the Tab Image 111
change the tab image for JSP-based tab containers 111
change the text 171
closing the Instant Messaging invite window 126
configure anonymous authentication 129, 129–132
configure to always use the anonymous log in 134
create a new Desktop 37
create a tab within a tab 110
customize container caching 115

customize JSPs for restructuring content layout 147
customize the channel border 122
customize the display of Instant Messaging contacts 126
customize the GlobalThemes 171
customize window preference 116
debug the Desktop 41
derive more Desktop layouts 145
disable the user from editing instant messaging server information 123
load the Display Profile 40
make the tab the start tab 112
modify the anonymous banner and menu bar 135
modify the default Desktop for the authlessanonymous user 140
perform JSP redirects 296
remove a button from a single channel 118
remove a button from all channels in a container 117
remove the title bar from a channel 120
stretch a tab across an entire container 110
use tags in JSPs 304
use the tag library in applications 307
write a new content channel 147
HTML files, Desktop online help 193

I

Image
changing the banner header logo 151
changing the tab image for JSP-based tab containers 111
IMProvider
JSPs 263
IMProvider/channel
Display Profile properties 65
tag library 334
`index.html` file,Portal Server 133
Instant Messaging
closing the invite window 126
customizations 123–127
customizing the display of contacts 126
disabling the user from editing the server information 123
Integer property 46
Integrated data 11
Introduction,Desktop customization 17
Italicized font usage 14

J

Jakarta tag library 177
JavaServer Pages (see JSPs,Desktop)
JSPContentContainer
JSPs 263, 282
JSPCreateChannelContainer
JSPs 282
JSPCustomThemeContainer
JSPs 283
JSPDynamicSingleContainer
JSPs 264, 283
JSPs for anonymous Desktop 252
JSPEditContainer
JSP 284
JSPs 264
JSPFrameCustomTableContainerProvider
customization 121, 122
description 103
JSP 284
JSPs 264
JSPLayoutContainer
JS 285
JSPs 265
JSPPopupContainer
JSPs 285
JSPPresetThemeContainer
JSPs 285
JSPProvider
JSPs 266
JSPProvider/channel
Display Profile properties 55
provider class 30
JSPs
tag library
attributes and return values 310

JSPs,Dekstop
JSPDynamicSingleContainer 264
JSPs,Desktop
anonymous Desktop 251
caching 298
changing the theme color 297
compilation errors 39
customizing for restructuring content layout 147
debugging 298
default location 29
deploying 33, 39
DiscussionLite channel 258
Discussions channel 259
DiscussionsProvider 261
DummyChannel 262
exception 39
for sample portal 29
FrameTabContainer for anonymous Desktop 252
IMProvider 263
in `default` directory 257–280
in `sampleportal` directory 280–296
installed location 180
introduction 29
introduction to SearchProvider 176
JSPContentContainer 263, 282
JSPCreateChannelContainer 282
JSPCustomThemeContainer 283
JSPDynamicSingleContainer 283
JSPDynamicSingleContainer for anonymous Desktop 252
JSPEditContainer 264, 284
JSPFrameCustomTableContainerProvider 264, 284
JSPLayoutContainer 265, 285
JSPPopupContainer 285
JSPPresetThemeContainer 285
JSPProvider 266
JSPProvider class 30
JSPSingleContainer 286
JSPSingleContainerProvider 266
JSPTabContainer 281, 286
JSPTabContainer for anonymous Desktop 252
JSPTabContainerProvider 267
JSPTabCustomTableContainerProvider 268, 287
JSPTableContainer for anonymous Desktop 253
JSPTableContainerProvider 269, 287
lookup scenario 31–33
miscellaneous
in `default` directory 270
in `sampleportal` directory 288
modifying 180
performing redirects 296
PredefinedFrontPageFramePanelContainerProvider 290
PredefinedFrontPageFramePanelContainerProvider for anonymous Desktop 254
PredefinedFrontPageTabPanelContainerProvider 291
PredefinedFrontPageTabPanelContainerProvider for anonymous Desktop 255
PredefinedSamplesFramePanelContainerProvider 293
PredefinedSamplesFramePanelContainerProvider for anonymous Desktop 256
PredefinedSamplesTabPanelContainerProvider 294
PredefinedSamplesTabPanelContainerProvider for anonymous Desktop 257
recompile 297
reference 251
runtime errors 39
sample JSP channel 295
SampleSimpleWebService 271, 296
SampleSimpleWebServiceConfigurable 272
Search 272
SearchProvider 275
SimpleWebServiceConfigurableProvider 277
SimpleWebServiceProvider 278
subscriptions channel 278
SubscriptionsProvider 279
TabJSPEditContainer 280
tag library
context setup tags 312
exceptions 311
normal tags 314
overview 301
validator tags 313
using tags 304–307
JSPSingleContainer 48
JSPs 286
JSPSingleContainerProvider 48, 147
JSPs 266
JSPTabContainer
adding a role-based tab 112

architecture 92
based contained containers 103
changing the tab image 111
default actions 90
default configuration 89
default Display Profile settings 90
Display Profile properties 90, 91
how-to add a tab 105–109
introduction 84
JSP files 92
JSPs 286
JSPs for anonymous Desktop 252
sample Desktop 89
usage 48
JSPTabContainerProvider 48, 84
JSPs 267
JSPTabCustomTableContainer
JSP 287
JSPTabCustomTableContainerProvider
customization 121, 122
description 103
JSPs 268
JSPTableContainer
architecture 96
default actions 94
default Display Profile settings 95
default layout 93
Display Profile properties 95
introduction 84
JSP files 96
JSPs 287
JSPs for anonymous Desktop 253
sample Desktop 93
usage 48
JSPTableContainerProvider 48, 84, 121, 122
JSPs 269
modifying the Content link 147

K

Knowledge management 11

L

Layout
adding new layouts 148
changing column layout 148
changing content layout 147
changing the page header and footer 152
Layout page,Desktop
changing the header and footer 152
header and footer files 153
Leaf provider / channel
Display Profile properties 54, 78
introduction 25
Link 170
Link separator 170
List,available and selected
categorizing 147
description 49
LoginProvider / channel
adding the channel to the anonymous Desktop 136, 136–139
Display Profile properties 66
`federationEnabled` 66
tags 240
template files 205
using UNIX authentication 141
Logo image
description 170
Display Profile properties 151
Look and feel customization,Desktop 20
LotusNotesAddressBookProvider / channel
Display Profile properties 66, 67
tags 229
template files 205, 206
LotusNotesCalendarProvider / channel
Display Profile properties 67, 68
tags 231–240
template files 206–208
LotusNotesMailProvider / channel
Display Profile properties 68, 69
tags 241
template files 208

M

MailCheckProvider / channel
  Display Profile properties 69
  tags 240
  template files 209
MailProvider / channel
  Display Profile properties 70, 71
  tags 241
  template files 209, 210
Menu bar
  modifying the anonymous Desktop menu bar 135
Miscellaneous JSPs
  in `default` directory 270
  in `sampleportal` directory 288
Monospaced font usage 14
MSExchangeAddressBookProvider / channel
  Display Profile properties 71, 72
  tags 229
  template files 210
MSExchangeCalendarProvider / channel
  Display Profile properties 72, 73
    `loadSubscribedCalendars` 72
  tags 231–240
  template files 211–213
MSExchangeMailProvider / channel
  Display Profile properties 73, 74
  tags 241
  template files 213
MyFrontPageFramePanelContainer 103
MyFrontPageTabPanelContainer 103
MyFrontPageTemplatePanelContainer
  parent 103
  tags 243
  template files 218

N

Normal tags 313
NotesProvider / channel
  Display Profile properties 74
  sample file 74
  storing 74
  text format 74

O

`obj.conf` file,Identity Server 132
Online help,Desktop
  customization 191–197
  file customization 195
  file location 193
  overview 191

P

Page piping 170
Pages,Desktop 152
PAPI (Provider Application Programming Interface) 19
PredefinedFrontPageFramePanelContainerProvider
  description 104
  JSPs 290
  JSPs for anonymous Desktop 254
PredefinedFrontPageTabPanelContainerProvider
  description 104
  JSPs 291
  JSPs for anonymous Desktop 255
PredefinedFrontPageTemplatePanelContainerProvider
  description 104
  tags 243
  template files 219–220
PredefinedNewTabPanelContainerProvider 109
PredefinedSamplesFramePanelContainerProvider
  description 104
  JSPs 293
  JSPs for anonymous Desktop 256
PredefinedSamplesTabPanelContainerProvider
  description 104
  JSPs 294
  JSPs for anonymous Desktop 257
PredefinedSamplesTemplatePanelContainerProvider

description 104
tags 243
template files 220, 222
PredefinedTabPanelContainer 84
PredefinedTemplatePanelContainer 84
PredefinedToolsTemplatePanelContainerProvider
description 104
Product name 170
Property,Display Profile
AddressBookProvider/channel 61
AppProvider/channel 62
BookmarkProvider/channel 63
CalendarProvider/channel 64
channel 23, 45
channel button 119
container 45, 49
content providers 61–78
DiscussionProvider/channel 59
Global 46
global 44, 46
GlobalThemes 167
IMProvider/channel 65
introduction 26
JSPProvider/channel 55
leaf providers 78
LoginProvider/channel 66
LotusNotesAddressBookProvider/channel 66
LotusNotesCalendarProvider/channel 67
LotusNotesMailProvider/channel 68
MailCheckProvider/channel 69
MailProvider/channel 70
MSExchangeAddressBookProvider/channel 71
MSExchangeCalendarProvider/channel 72
MSExchangeMailProvider/channel 73
NotesProvider/channel 74
object 23
provider 23, 44
SearchProvider/channel 58
SimpleWebServiceConfigurableProvider/channel 76
SimpleWebServiceProvider/channel 75
table container 50
type
Boolean 45
Collection 46
ConditionalProperty 46
Integer 46
introduction 45
Reference 46
String 46
URLScraperProvider/channel 56
UserInfoProvider/channel 77
XMLProvider/channel 57
Provider
creating and extending 42
definition 21
Display Profile
definition 26
object 23, 26
properties 23, 44
dynamic content 20
Provider Application Programming Interface (*see* PAPI)
Provider,building-block
container
JSPSingleContainerProvider 48
JSPTabContainerProvider 48
JSPTableContainerProvider 48
introduction 37
leaf
Display Profile properties 54–57
JSPProvider 55
URLScraperProvider 55
XMLProvider 56
Provider,content
AddressBookProvider 61
AppProvider 62
BookmarkProvider 63
CalendarProvider 63
Display Profile properties 61–78
IMProvider 64
introduction 37
LoginProvider 65
LotusNotesAddressBookProvider 66
LotusNotesCalendarProvider 67
LotusNotesMailProvider 68
MailCheckProvider 69
MailProvider 70
MSExchangeAddressBookProvider 71
MSExchangeCalendarProvider 72
MSExchangeMailProvider 73
NotesProvider 74
SimpleWebServiceConfigurableProvider 76

SimpleWebServiceProvider 75
UserInfoProvider 77

R

Reference property 46
Refresh times,channel 115
`refreshTime` property,channel 115
Resource bundle,creating 41
Resources,static 20

S

Sample portal
adding a theme 170
Desktop
containers 84
default channel 85
introduction 36
viewing 85
Display Profile definitions 43
editing the files 85
elements 83
installation directories 87
introduction 36
overview 83
restoring default settings 87
templates 20
usage guidelines 85
`sampleportal` directory
JSP files 280–296
template files 224
SamplesFramePanelContainer 103
SampleSimpleWebService
JSP 296
JSPs 271
SampleSimpleWebServiceConfigurable
JSPs 272
SamplesTabPanelContainer 103
SamplesTemplatePanelContainer
tags 243
template files 222
SearchProvider/channel
accessing channel directly 181
adding a new field 185
advanced search 58
basic search 58
browse 58
channel customization 20
customization
adding last-modified 182
displaying the total number of documents 183
modifying the default search server 181
overview 175
removing author 184
removing content-length 183
design 176
Display Profile properties 58, 59
exceptions 332
JavaServer Pages
introduction 176
layout 176
JSP tags 326
JSPs 275–277
log files 180
search JSPs 272–274
service APIs 19
tag libraries 176
SearchTabPanelContainer 103
Secondary channel title bar 170
Selected list
categorizing 147
for container 49
Selected tab 170
Service provider
customization 175–190
customization overview 175
customization tips 179
debugging 179
DiscussionProvider (*see* DiscussionProvider/channel)
Display Profile properties 57–60
SearchProvider (*see* SearchProvider/channel)
SubscriptionsProvider 279
SubscriptionsProvider (*see* SubscriptionsProvider/channel)
SimpleWebServiceConfigurableProvider/channel

Display Profile properties 76
JSPs 277
SimpleWebServiceProvider / channel
Display Profile properties 75, 76
JSPs 278
Single Sign-on,enabling 19
Solaris administrative procedures 12
Square brackets usage 14
Static resources 20
String property 46
Structure,Display Profile document 21
SubscriptionsProvider / channel 60
channel JSPs 278
JSPs 279

T

Tab Container
display profile properties 53
Tab notch 170
TabJSPEditContainer
JSPs 280
Table background 170
Table container
changing the channel layout 120
display profile properties 51–53
Tabs,Desktop
adding a channel to a user-defined tab 114
adding a role-based tab 112
changing the back-ground color 111
changing the tab image 111
making a tab the start tab 112
strectching a tab 110
Tag libraries
AddressBookProvider 229
AppProvider 229
BookmarkProvider 229
CalendarProvider 231
Desktop template
common 245–250
overview 227
types 228
IMProvider tags 334
introduction 29
introduction to SearchProvider 176
Jakarta 177
JavaServer Pages (JSPs)
attributes and return values 310
context setup tags 312
exceptions 311
hierarchy 303
normal tags 314
Overview 301
validator tags 313
LoginProvider 240
LotusNotesAddressBookProvider 229
LotusNotesCalendarProvider 231
LotusNotesMailProvider 241
MailCheckProvider 240
MailProvider 241
MSExchangeAddressBookProvider 229
MSExchangeCalendarProvider 231
MSExchangeMailProvider 241
MyFrontPageTemplatePanelContainer 243
PredefinedFrontPageTemplatePanelContainerProvider 243
PredefinedSamplesTemplatePanelContainerProvider 243
SamplesTemplatePanelContainer 243
search tags 326
Tag Library Descriptors (*see* TLDs)
TemplateTabContainerProvider 242
TemplateTableContainerProvider 243
UserInfoProvider 244
using in application 307
TemplateEditContainerProvider
description 103
Templates
base Desktop 20
deploying 33, 39
in `default` directory 200–218
in `sampleportal` directory 218–224
installed location 29
introduction 29, 30
lookup scenario 31–33
sample portal 20
tag library overview 227
TemplateTabContainer
based contained container 103
introduction 84

TemplateTabContainerProvider / channel
Desktop information 84
introduction 49
tags 242
template files 214, 222
TemplateTabCustomTableContainerProvider
description 104
template files 222
TemplateTableContainer
description 84
template files 223
TemplateTableContainerProvider / channel
introduction 49
reference to 84
tags 243
template files 215–216
Themes (*see* GlobalThemes)
Title,HTML 151
TLDs
`desktop.tld` 302
`desktopContainerProviderContext.tld` 302
`desktopProviderContext.tld` 302
`desktopSingle.tld` 302
`desktopTab.tld` 303
`desktopTable.tld` 303
`desktopTheme.tld` 303
`im.tld` 303
`jr.tld` 303
`jx.tld` 303
`search.tld` 303
ToolsTemplatePanelContainer
template files 223
Top container,customizing 20

U

Unselected tab 170
URLScraperProvider / channel
Display Profile properties 56
UserInfoProvider / channel
configuring LDAP authentication 142
Display Profile properties 77, 78
tags 244
template files 216

V

Validator tags 313
Variables used in Guide 15

W

Widths
channel width 145
modifying column width 149
WSDL document 75

X

XML files,Display Profile
`dp-anon.xml` 44, 88
`dp-org.xml` 43, 87, 88
`dp-providers.xml` 43, 87
XMLProvider / channel
Display Profile properties 57