Sun Java™ System

Portal Server 6
Technical Reference Guide

2005Q4

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

Part No: 819-4159-10

Contents

Preface

The *Technical Reference Guide* provides detailed information on the Sun Java™ System Portal Server technical concepts (such as Display Profile, Rewriter), command line utilities, tag libraries (in the software), and files (such as templates and JSPs). Use this book in conjunction with the other books in the Portal Server documentation set (see "Books in This Documentation Set.")

Who Should Use This Book

The *Technical Reference Guide* is intended for use by administrators and individuals responsible for:

- Administering and configuring the Portal Server software
- Customizing pieces of the software

Readers should have admin-level permissions to the software. This guide is not intended for end users.

Administrators and individuals using this guide must be familiar with basic Solaris™ administrative procedures and must understand the following technologies:

- Lightweight Directory Access Protocol (LDAP)
- Java™ technology
- JavaServer Pages™ (JSP™) technology
- Hypertext Transfer Protocol (HTTP)
- Hypertext Markup Language (HTML)
- Extensible Markup Language (XML)

Before You Read This Book

Portal Server is a component of Sun Java Enterprise System, a software infrastructure that supports enterprise applications distributed across a network or Internet environment. You should be familiar with the documentation provided with Sun Java Enterprise System, which can be accessed online at http://docs.sun.com/coll/entsys_05q1.

Because the Portal Server software components work together with Sun Java™ System Identity Server for user, service, and policy management and authentication, single sign-on, and logging services, you should be familiar with the documentation provided with that product. Identity Server software documentation can be accessed online at http://docs.sun.com/doc.

Because Sun Java™ System Directory Server is used as the data store for primary configuration information and user profile data in a Portal Server deployment, you should be familiar with the documentation provided with that product. Directory Server software documentation can be accessed online at http://docs.sun.com/coll/DirectoryServer_05q1.

How This Book Is Organized

The first chapter of this book provides an overview of the entire Portal Server product. The rest of the book is divided into parts; each part includes multiple chapters detailing a concept. The following table summarizes the various parts of this book.

Table 0-1 How This Book Is Organized

Part Title	Part Description
Part I, “Attributes”	Describes the various attributes in the Portal Server software.
Part II, “Command Line Utilities”	Describes the various command line utilities in the Portal Server software.
Part III, “Configuration Files”	Describes properties in the properties file for the Desktop and Search service.
Part IV, “Display Profile”	Describes the display profile document, objects, priorities, merge semantics, and properties.
Part V, “Desktop JavaServer Pages”	Describes the various JSPs installed in the `default` and `sampleportal` directories.
Part VI, “Rewriter”	Describes the supported URLs and provide background information on defining rules and rulesets.

Table 0-1 How This Book Is Organized *(Continued)*

Part Title	Part Description
Part VII, “Sample Portal”	Describes the software Sample Portal and its associated containers and themes. The sample portal is an authentication-less desktop that consists of of sample containers, channels, portlets, services, and templates which demonstrates the Portal Server software’s capabilities.
Part VIII, “Search Engine Robot”	Describes the Search engine robot and its application functions. The Search Engine robot is included to discover, convert, and summarize document resources. Users can then locate or browse content contained in the database.
Part IX, “Desktop Tag Library”	Describes the tag libraries used by the Desktop and Search service.
Part X, “Desktop Templates”	Describes the Desktop templates installed in the `default` and `sampleportal` directories.

Conventions Used in This Book

The tables in this section describe the conventions used in this book.

Typographic Conventions

The following table describes the typographic changes used in this book.

Table 2 Typographic Conventions

Typeface	Meaning	Examples
AaBbCc123 (Monospace)	API and language elements, HTML tags, web site URLs, command names, file names, directory path names, onscreen computer output, sample code.	Edit your `.login` file. Use `ls -a` to list all files. `% You have mail.`
AaBbCc123 (Monospace bold)	What you type, when contrasted with onscreen computer output.	`% su` `Password:`

Table 2 Typographic Conventions *(Continued)*

Typeface	Meaning	Examples
AaBbCc123 (Italic)	Book titles, new terms, words to be emphasized. A placeholder in a command or path name to be replaced with a real name or value.	Read Chapter 6 in the *User's Guide*. These are called *class* options. Do *not* save the file. The file is located in the *install-dir*`/bin` directory.

Symbols

The following table describes the symbol conventions used in this book.

Table 3 Symbol Conventions

Symbol	Description	Example	Meaning
`[ ]`	Contains optional command options.	`ls [-l]`	The `-l` option is not required.
`{ | }`	Contains a set of choices for a required command option.	`-d {y|n}`	The `-d` option requires that you use either the `y` argument or the `n` argument.
-	Joins simultaneous multiple keystrokes.	Control-A	Press the Control key while you press the A key.
+	Joins consecutive multiple keystrokes.	Ctrl+A+N	Press the Control key, release it, and then press the subsequent keys.
>	Indicates menu item selection in a graphical user interface.	File > New > Templates	From the File menu, choose New. From the New submenu, choose Templates.

Default Paths and File Names

The following table describes the default paths and file names used in this book.

Table 0-4 Default Paths and File Names

Term	Description
PortalServer-base	Represents the base installation directory for Sun Java System Portal Server software. The Portal Server software default base installation and product directory depends on your specific platform: Solaris™ systems: /opt/SUNWps
AccessManager-base	Represents the base installation directory for Sun Java System Identity Server software. The Identity Server software default base installation and product directory depends on your specific platform: Solaris™ systems: /opt/SUNWam
DirectoryServer-base	Represents the base installation directory for Sun Java System Directory Server software. The Directory Server software default base installation is `/var/opt/mps/serverroot`.
ApplicationServer-base	Represents the base installation directory for Sun Java™ System Application Server software. The Application Server software default base installation is `/opt/SUNWappserver8`.
WebServer-base	Represents the base installation directory for Sun Java™ System Web Server software. The Web Server software default base installation is `/opt/SUNWwbsvr`.
PortalServer-ContentFiles	Represents the directory where JSPs, templates and property files, and tag libraries are installed. By default, this is `/etc/opt/SUNWps`.
PortalServer-SEdb	Represents the Portal Server software Search Engine database. By default, this is `/var/opt/SUNWps`.

Shell Prompts

The following table describes the shell prompts used in this book.

Table 5 Shell Prompts

Shell	Prompt
C shell on UNIX or Linux	*machine-name*%
C shell superuser on UNIX or Linux	*machine-name*#
Bourne shell and Korn shell on UNIX or Linux	$
Bourne shell and Korn shell superuser on UNIX or Linux	#
Windows command line	C:\

Related Documentation

The http://docs.sun.com℠ web site enables you to access Sun technical documentation online. You can browse the archive or search for a specific book title or subject.

Books in This Documentation Set

The following table summarizes the books included in the Portal Server core application documentation set.

Table 0-6 Books in This Documentation Set

Book Title	Description
Sun Java System Portal Server Administration Guide http://docs.sun.com/db/doc/817-7691	Describes how to administer the Portal Server software using the administration console and the command line.
Portal Server Secure Remote Access Administration Guide http://docs.sun.com/db/doc/817-7693	Describes how to administer Portal Server Secure Remote Access software.
Sun Java System Portal Server Desktop Customization Guide http://docs.sun.com/db/doc/817-7694	Describes how to customize the Portal Server software Desktop.
Sun Java System Portal Server Developer's Guide http://docs.sun.com/db/doc/817-7695	Describes how to extend the Portal Server software APIs.
Portal Server Deployment Planning Guide http://docs.sun.com/db/doc/817-7697	Describes how to plan for and deploy Portal Server software.
Portal Server Migration Guide http://docs.sun.com/db/doc/817-7698	Describes how to migrate from previous releases of the Portal Server software to this release.

Other Server Documentation

For other server documentation, go to the following:

- Directory Server documentation: http://docs.sun.com/coll/DirectoryServer_04q2
- Web Server documentation: http: //docs.sun.com/coll/S1_websvr61_en
- Application Server documentation: http://docs.sun.com/coll/s1_asseu3_en
- Identity Server documentation: http://docs.sun.com/

Accessing Sun Resources Online

For product downloads, professional services, patches and support, and additional developer information, go to the following:

- Download Centerhttp://wwws.sun.com/software/download/
- Professional Services: http://www.sun.com/service/sunps/sunone/index.html
- Sun Enterprise Services, Solaris Patches, and Support: http://sunsolve.sun.com/
- Developer Information: http://developers.sun.com/prodtech/index.html

Contacting Sun Technical Support

If you have technical questions about this product that are not answered in the product documentation, go to http://www.sun.com/service/contacting.

Related Third-Party Web Site References

Sun is not responsible for the availability of third-party web sites mentioned in this document. Sun does not endorse and is not responsible or liable for any content, advertising, products, or other materials that are available on or through such sites or resources. Sun will not be responsible or liable for any actual or alleged damage or loss caused or alleged to be caused by or in connection with use of or reliance on any such content, goods, or services that are available on or through such sites or resources.

Sun Welcomes Your Comments

Sun is interested in improving its documentation and welcomes your comments and suggestions.

To share your comments, go to http://docs.sun.com and click Send Comments. In the online form, provide the document title and part number. The part number is a seven-digit or nine-digit number that can be found on the title page of the book or at the top of the document. For example, the title of this book is *Sun Java System 6 2005Q4 Technical Reference Guide* and the part number is 819-4159-10.

Part I

Attributes

Chapter 1

Desktop Attributes

This chapter contains the following sections:

Introduction

The Desktop Service consists of global and dynamic attributes. The values applied to the global attributes are applied across the Sun Java System Identity Server configuration and are inherited by every configured organization. They cannot be applied directly to roles or organizations as the goal of global attributes is to customize the Identity Server application. Values applied to the dynamic attributes are assigned to a role or organization. When the role is assigned to a user or a user is created in an organization, the dynamic attribute then becomes a characteristic of the user.

Desktop Global Attributes

Table 1-1 on page 28 describes the global attributes for the Desktop Service.

The table contains three columns: the first column identifies the attribute, the second column provides the default value for the attribute, and the third column describes the attribute.

Table 1-1 Desktop Service - Global Attributes

Attribute	Default Value	Description
Enable XML Parsing Validation (Use with Caution)	True (checked)	Specifies whether to enforce validation while parsing the display profile XML document. Unchecking this attribute can improve system performance. However, this can potentially introduce corruption in the display profile document because the resulting XML document might include some fragments that do not conform to the DTD.
Namespace URI	http://www.iplanet.com	Specifies the unique identifier for the XML namespaces or Uniform Resource Indicator (URI) in the form of a URL. This guarantees that XML tags will be unique.
Enable Federation	False (unchecked)	Enables Identity Federation so that a user can associate, connect or bind multiple internet service providers' local identities, enabling them to have one network identity.
Hosted Provider ID	None	Specifies the unique identifier of the host that provides the network identity of a user.
Client Session Reap Interval (seconds)	1800	Defines in seconds the time interval between checks for removing inactive client sessions.
Client Session Maximum Inactive Session Time Before Reap (seconds)	3600	Specifies the maximum number of seconds a client session can be idle before it is considered inactive. If a session is idle for more than this value, it is made a candidate for session reaping and can be removed the next time the client session times out.

Table 1-1 Desktop Service - Global Attributes *(Continued)*

Attribute	Default Value	Description
Display Profile	The default value depends on the type of installation performed. If the sample portal is installed, the Display Profile contains the definitions for the built-in providers (the basic providers of Portal Server), such as bookmark and notes. If the sample portal was not installed, the global Display Profile is blank.	Displays several controls for manipulating the global display profile, an XML document that defines the container management, channel attributes, and provider definitions for the organization. The controls include: • Disable Authentication-less Access for Federated Users--Prevents a user with a federated network identity to access the portal without entering a user name and password. • Upload XML--Allows you to upload an XML file containing display profile information to the Portal Server. • Download XML--?Allows you to download the display profile to your local drive. • Channel and Container Management--Provides a graphical user interface to manage container channels and channels without the need to edit the XML file. These links are not attributes. Selecting these links allows you to manipulate the display profile. Display profile elements defined in the global display profile are inherited by all users on the system, regardless of the organization or role to which they belong.

Table 1-1 Desktop Service - Global Attributes *(Continued)*

Attribute	Default Value	Description
Authentication-less Portal Desktop Configuration	Enable	Displays several controls for configuring authentication-less configuration of the portal desktop. The controls are: • Disable Authentication-less Access for Federated Users--Allows you to prevent users with a network identity on a hosted provided to access the portal desktop with providing a user name and password. • DefaultAuthentication-less User ID--Defines the User IDs that are authorized to access the Desktop without authenticating. • Authorized Authentication-less User ID--Defines the User IDs that are authorized to access the Desktop without authenticating.

Desktop Dynamic Attributes

Table 1-2 on page 31 describes the dynamic attributes for the Desktop Service.

The table contains three columns: the first column identifies the attribute, the second column provides the default value for the attribute, and the third column describes the attribute.

Table 1-2 Desktop Service - Dynamic Attributes

Attribute	Default Value	Description
Conflict Resolution Level	Highest	Sets the conflict resolution level for the Desktop service template used to resolve conflicts when multiple Desktop templates are merged. There are seven conflict resolution settings available ranging from Highest to Lowest. Do not confuse this setting with the display profile document priority. The display profile document priority is a numeric value that is set in the XML file with the priority= syntax tag. When a merge occurs, it starts with the lowest display profile priority document (lowest number) and proceeds in increasing priority number, until it arrives at the user (base), the highest priority display profile. When an attribute conflict occurs, the attribute on the template set with the highest conflict resolution level is returned.
Default Channel Name	JSPTabContainer	Identifies which default channel is rendered when the Desktop is called with an unspecified provider.
Default Edit Channel Name	JSPEditChannel	Specifies which default edit channel to use to wrap the content when one is not specified in the URL. When a channel is edited, an “Edit” request URL is sent to the Desktop Servlet. The URL generated for the “Edit” of each of the channels inside a container depends on the property “editContainerName” defined in the display profile. If you have migrated containers from iPlanet™ Portal Server 3.0, you must specify the default edit channel with which to wrap the content using this attribute because the URL format has changed.

Table 1-2 Desktop Service - Dynamic Attributes *(Continued)*

Attribute	Default Value	Description
Desktop Type	default	Retrieves template files for the specified Desktop type when different Desktop configurations are needed and when different sets of templates and JSPs are required for those configurations. The Desktop type attribute of the Desktop service is a comma-separated string. type, that the Portal Desktop uses as an ordered 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. If the sample portal is not installed, then the Desktop type attribute value is set to default. Most sites will not use the default Desktop type, as they will have different channels, different logo, different look and feel, and the like.

Table 1-2 Desktop Service - Dynamic Attributes *(Continued)*

Attribute	Default Value	Description
Display Profile	The default value depends on the type of installation performed. If the sample portal was installed, a sample display profile document is installed at the organization level that contains channels that display the built-in providers defined in the global display profile.	Displays several links for manipulating the display profile, an XML document that defines the container management, channel attributes, and provider definitions for this specific node (role, organization, suborganization). Links are: • Edit XML--Allows you to edit the entire display profile XML file. • Upload XML--Allows you to upload an XML file containing display profile information to the Portal Server. • Download XML--Allows you to download the display profile to your local drive. These links are not attributes. Selecting these links allows you to manipulate the display profile.
Show Desktop Service Attributes	True (checked)	Specifies whether the Desktop Service attributes are displayed to the users associated with the role. This dynamic attribute is mainly used for role-based delegated administration, Values applied to this attribute are only in effect for a role. When the role is assigned to a user and the value of this attribute is false, users (usually delegated administrators) cannot see any Desktop Service attributes except the Channel and Container Management link when they navigate into all the roles within the organization.

Chapter 2

NetMail Attributes

This chapter contains the following sections:

Introduction

The NetMail Service consists solely of dynamic attributes. Values applied to the dynamic attributes are assigned to a role or organization. When the role is assigned to a user or a user is created in an organization, the dynamic attribute then becomes a characteristic of the user.

NetMail Dynamic Attributes

Table 2-1 describes the dynamic attributes for the NetMail Service.

The table contains three columns: the first column identifies the attribute, the second column provides the default value for the attribute, and the third column describes the attribute.

Table 2-1 NetMail Service - Dynamic Attributes

Attribute	Default Value	Description
Incoming mail (IMAP) server	Set by the administrator)	Specifies the host name of the IMAP server to which NetMail should connect.
Outgoing mail (SMTP) server	Set by the administrator)	Specifies the server that NetMail uses to send outgoing messages through SMTP.

Table 2-1 NetMail Service - Dynamic Attributes *(Continued)*

Attribute	Default Value	Description
Default mail domain	Set by the administrator)	Specifies the name of the default mail domain.
IMAP top-level folder	Mail	Specifies the folder in which user mail folders reside on the IMAP mail server.
Cache folder list	False (unchecked)	Specifies whether to load list of folders into user memory caches automatically when they disconnect from the mail server. By caching the folder list, users can move and copy messages when they use NetMail in disconnected mode. False (unchecked) activates loading. True (checked) turns loading off.

Table 2-1 NetMail Service - Dynamic Attributes *(Continued)*

Attribute	Default Value	Description
LDAP server details to use in address book search	(Set by the administrator)	Specifies what LDAP server information to use when performing an address book search. The information is used only in the NetMail applet. Each entry is a comma separated list of name/value pairs in the following format: name="value." Quotation marks are not allowed within any value. Users cannot modify this value. The valid names and corresponding preferences are: • name--The name of the LDAP server that is visible to users. • server--The LDAP server and port number, in the form `ldapserver[:ldapportnumber]`. • base--The string used as base to search for the users and groups, for example: `c=US,o=sesta`. • searching--A comma separated list, for example: `cn,gn,sn`. • result--LDAP attribute name, for example, mail. • filter--LDAP search filters; see RFC2254. • referral-- LDAP referrals, follow or ignore (does not apply). Example: name="Sesta LDAP",server= "ldap-server.sesta.com",base="dc=sesta, dc=com"
Initial Headers downloaded from the IMAP server	10	Controls the number of initial headers that are downloaded from the IMAP server when a user opens a folder. Applies only to NetMail Java.
Message headers displayed per page	10	Specifies the number of initial headers that are downloaded from the IMAP server when users open folders. Applies only to NetMail Lite.

Table 2-1 NetMail Service - Dynamic Attributes *(Continued)*

Attribute	Default Value	Description
Check new mail every (mins)	5	Determines the frequency, in minutes, that NetMail checks for new messages in the currently selected folder. If set to 0, NetMail does not check for new messages.
Do not load attachments larger than (KB)	0	Specifies the maximum size of attachments, in kilobytes, that NetMail loads automatically into the memory cache when users disconnect. Specify 0 to load all attachments. Applies only to NetMail Java.
Types of messages to load on disconnect	All	Specifies the type of messages that are automatically loaded into the memory cache when users disconnect. Valid values are: All, None, New, Unread, New and Unread, and Found. For example, if Unread is selected, all unread messages in the inbox are loaded to memory cache Users can then read these messages in disconnected mode. Applies only to NetMail Java.
Sort by newest messages first	True (checked)	Identifies what messages are to appear first in the selected folder. True (checked) specifies that newest messages are to appear first. False (unchecked) specifies oldest messages first. Applies only to NetMail Lite.
Open messages in separate windows	False (unchecked)	Specifies whether to open a new read window for each new message that users view. True (checked) specifies that a new read window should be opened for each message. False (unchecked) specifies that new messages be viewed in the current window. Applies only to NetMail Java.

Table 2-1 NetMail Service - Dynamic Attributes *(Continued)*

Attribute	Default Value	Description
Purge deleted messages on exit	False (unchecked)	Specifies whether to remove messages from the inbox that are flagged as deleted when users exit or disconnect. True (checked) specifies that flagged messages be deleted upon disconnect. False (unchecked) specifies that messages not be deleted.
Sent messages folder (on server)	Mail/Sent	Specifies the folder in which outgoing messages are logged.
Save sent messages in the Sent folder (on server)	False (unchecked)	Specifies to save user messages in the Sent folder stored on the IMAP server. True (checked) specifies to save messages in the Sent folder on the server. False (unchecked) specifies not to save messages in the Sent folder.
Keep copy of sent messages in local cache	True (checked)	Specifies whether to save the cache to disk at exit automatically. If not enabled, users are prompted to Save the cache to disk before exiting. True (checked) specifies to save the cache to disk automatically. False (unchecked) specifies to prompt the user whether to cache. Applies to NetMail locally installed applet only.
Quote prefix for replies	>	Specifies which character string precedes each text line of a reply message.
Append signature to outgoing mail	False (unchecked)	Specifies whether to append user signatures to outgoing messages. True (checked) specifies to append the signature. False (unchecked) specifies not to append the signature.

Table 2-1 NetMail Service - Dynamic Attributes *(Continued)*

Attribute	Default Value	Description
Include the author in reply	False (unchecked)	Specifies whether to include the author of the original message with a reply message. True (checked) specifies to include the author of the original message. False (unchecked) specifies not to include the author of the original message. Applies only to NetMail Java.
Include the Date of the original message in reply	False (unchecked)	Specifies whether to include the date of the original message with a reply message. True (checked) specifies to include the date of the original message. False (unchecked) specifies not to include the date of the original message. Applies only to NetMail Java.
Include the Body of the original message in reply	True (checked)	Specifies whether to include the body of the original message with a reply message. True (checked) specifies to include the body of the original message. False (unchecked) specifies not to include the body of the original message. Applies only to NetMail Java.
Preferences the user cannot change	(Set by the administrator)	Specifies the NetMail preferences attributes that the end user cannot change. The valid values are: IMAPPassword, IMAPServerName, IMAPUserName, SMTPMailServer, autoFolderLoad, autoload, autopurge, autosave, backgroundColor, inactivityInterval, indentPrefix, initialHeaders, saveSentMessage, maxAttachLen, multipleReadWindows, replyFields, replyToAddress, resetSize, rootFolder, sentMessagesFolder, signature, textColor, textSize, and textStyle.

Chapter 3

Rewriter Attributes

The Rewriter provides a Java™ class library for rewriting URL references in various web languages such as HTML, JavaScript, and WML, and in HTTP Location headers (redirections). The Rewriter Service does not consist of any attributes.

To implement the service, you create Rewriter rules that define how rewriting is to be done and the data to be rewritten. You can create and edit Rewriter rules through the administration console.

Chapter 4

Search Attributes: Server

This chapter contains the following sections:

Introduction

This chapter describes attributes that you can configure for the search engine through the Sun Java System Identity Server administration console.

When you select Search Properties from the Service Management View, a two-toned tabbed menu bar is displayed. This chapter is organized according to the topics or tabs on the upper portion of the menu bar.

When one of these tabs is selected, the menu bar below lists the related subtopics for the topic. The default Search page selects Server / Settings. Each subtopic uses one or more tables to explain the attributes for that subtopic. The tables are divided into three columns: Attribute, Default Value and Description. The Attribute gives the descriptive text found on the page; the Default Value provides the default value for the Attribute; and the Description explains the Attribute and its format.

Every Search Properties page gives you the Select Server attribute as described in the Table 4-1. The table contains three columns: the first column identifies the attribute, the second column provides the default value for the attribute, and the third column describes the attribute.

Table 4-1 Search Properties

Attribute	Default Value	Description
Select Server	http://servername:80/portal	Fully qualified server name of your Search server.

Server

The Server section is where you configure the preferences for your server. You select what directory to use for temporary files, what information to log and how much detail should be in the logs. The Server attributes are displayed on two pages:

- Settings
- Robot

Settings

This page contains the basic settings for the administration and operation of the search server. Table 4-2 lists the Server Settings Attributes. The table contains three columns: the first column identifies the attribute, the second column provides the default value for the attribute, and the third column describes the attribute.

Table 4-2 Server Settings Attributes

Attribute	Default Value	Description
Server Root	`/var/opt/SUNWps/https-servernamefull/portal`	Houses the configuration, log, database, and robot information files. Also it is the root directory for all of the search files that are generated and updated when conducting a search. This is not configurable.
Temporary Files	`/var/opt/SUNWps/https-servernamefull/portal/tmp`	Contains all temporary files used to manage a search during the search. It includes newly generated resource descriptions that have not yet been added to the main database. These are removed when the search is completed.

Table 4-2 Server Settings Attributes *(Continued)*

Attribute	Default Value	Description
Document level security	Off	Controls who can access documents. When this setting is changed, the server must be restarted. Values: • Off (default) means all users have access to the RDs. • On means that the ReadACL field in an RD is checked to see if the user asking for the RD has permission because the user is in an acceptable organization or role, or is an acceptable individual user. The ReadACL field is set in the Database, Resource Descriptor page.

Robot

This page contains the advanced settings for the administration and operation of the search server. Here is where you configure the log files for user queries, index maintenance, resource description management, and debugging. Table 4-3 describes the Server Advanced Settings Attributes. The table contains three columns: the first column identifies the attribute, the second column provides the default value for the attribute, and the third column describes the attribute.

Table 4-3 Server Advanced Settings Attributes

Attribute	Default Value	Description
Search (rdm)	`/var/opt/SUNWps/https-servername/portal/logs/rdm.log`	Logs the queries end users make of the database. You can check the Disable Search Log checkbox to suppress this logging. If you do, you cannot view the User Queries (rdm) report.

Table 4-3 Server Advanced Settings Attributes *(Continued)*

Attribute	Default Value	Description
Disable Search Log	False (unchecked) - enabled	Controls use of query log. In the report section, you can generate a report the lists the most popular queries based on this log. Values: • Checked--disabled • Unchecked--enabled. Every user query is entered in this log.
Index Maintenance	`/var/opt/SUNWps/https-servername/portal/logs/searchengine.log`	Logs the transactions involving the search engine, except for not registration of resource descriptions.
RD Manager	`/var/opt/SUNWps/https-servername/portal/logs/rdmgr.log`	Logs the registration of resource descriptions from the robot or import agents into the database. You can view this log as a RD Manager (rdmgr) report.
RDM Server	`/var/opt/SUNWps/https-servername/portal/logs/rdmserver.log`	Logs debugging information on RDM transactions. The level of detail is controlled by the Log Level. You can view this log as a RDM Server (rdmsvr) report.
Log Level	1	Controls the amount of detail the RDM Server log file contains. The possible levels are 2, 10, 20, 50, 100, and 999. A setting of 1 (default) logs only severe errors. The higher the number, the more detail the RDM Server log file contains.

Chapter 5

Search Attributes: Robot

The properties for the robot are quite complex. You can select the sites to be searched or crawled, check to see if a site is valid, define what types of documents should be picked up, and schedule when the searches take place.

This chapter contains the following sections:

Overview

The Robot Overview panel is where you can see what the robot is doing: if it is Off, Idle, Running, or Paused; and if it is Running, what progress it is making in the search since the panel is refreshed about every 30 seconds. The refresh rate is defined using the robot-refresh parameter in the search.conf file.

The two buttons on the top right are appropriate for its state. If the robot is Off, the buttons are Start and Remove Status. If it is Running or Idle, the two buttons are Stop and Pause. If it is Paused, the two buttons are Stop and Resume. By selecting on any of the Attributes, you go to the Reports section where you can get a detailed up-to-the-minute report of that Attribute.

Table 5-1 describes the attributes in the Robot Overview panel. The table contains three columns: the first column identifies the attribute, the second column provides the default value for the attribute, and the third column describes the attribute.

Table 5-1 Robot Overview Attributes

Attribute	Default Value	Description
The Robot is	Current activity	The Robot's state. Value can be Idle, Running, Paused, or Off
Updated at date	Date and time last refreshed.	This page is refreshed to keep you aware of what progress the robot is making.
Starting Points	Number defined	Number of sites you have selected to be searched. A site is disabled (not included in a search) on the Robot, Site page.
URL Pool	Number URLs waiting	Number of URLs yet to be investigated. When you begin a search, the starting point URLs are entered into the URL pool. As the search progresses, the robot discovers links to other URLs. These URLs get added to the pool. After all the URLs in the pool have been processed, the URL pool is empty and the robot is idle.
Extracting	Number connections per second	Number of resources looked at in a second. Extracting is the process of discovering or locating resources, documents or hyperlinks to be included in the database and filtering out unwanted items.
Filtering	Number URLs rejected	Total number of URLs that are excluded.
Indexing	Number URLs per second	Number of resources or documents turned into a resource description in a second. Indexing is the phase when all the information that has been gathered on a document is turned into a resource description for inclusion in the search database.
Excluded URLs	Number URLs excluded by filters	Number of URLs that did not meet the filtering criteria.
	Number URLs excluded by errors	Number of URLS where the robot encountered errors as file not found.

Table 5-1 Robot Overview Attributes *(Continued)*

Attribute	Default Value	Description
Resource Descriptions	Number RDs contributed	Number of resource descriptions added to the database.
	Number Bytes of RDs contributed	Number of bytes added to the database.
General Stats	Number URLs retrieved	Number of URLs retrieved during run.
	Number Bytes average size of RDs	Average number of bytes per resource description.
	Time in days, hours, minutes, and seconds running	The amount of time the robot has been running.

Sites

The initial page in this section shows what sites are available to be searched.

A site can be enabled (On) and disabled (Off) by using the radio buttons. A disabled site is not searched when the robot is run. The Edit link displays a page where you can change how a search site is defined.

To delete a site, check the checkbox and select Delete.

To add a new site, select New. Add a URL or Domain in the text box and select a depth for the search. Select Create to use the default values. Otherwise, select Create and Edit to select non-default values and go to the Edit page to define the search site.

Table 5-2 describes the attributes in the Robot Manage Sites page. The table contains three columns: the first column identifies the attribute, the second column provides the default value for the attribute, and the third column describes the attribute.

Table 5-2 Robot Manage Sites Attributes

Attribute	Default Value	Description
Lock or cluster graphic	Status of site	Lock open means that the URL is accessible. The closed lock means that the site is a secure web server and uses SSL. The cluster means that the site is a domain.
On/Off	On	Choose to search this site or not when the robot is run.

The New Site page allows you to set up an entire site for indexing. Table 5-3 contains the attributes for the Robot New Site page. The table contains three columns: the first column identifies the attribute, the second column provides the default value for the attribute, and the third column describes the attribute.

Table 5-3 Robot New Site Attributes

Attribute	Default Value	Description
New site	URL	URL - format: http://www.sesta.com Domain - format: *.sesta.com
Depth	10	You have a choice of 1 for this URL only, 2 for this URL and first links, 100 for the robot, 3 - 10 or unlimited. The default value is set on the Robot, Crawling page.

The edit page is where you can define the search site more completely. You can specify what type of server it is, redefine the depth of the search, and select what type of files to add to the search database. The attributes for URL and Domain sites are mostly the same. The additional column in this table shows which attributes are shared and which are unique.

A number of actions are performed on this page. You can verify the server name for the search site you entered. You can add more servers to the server group by selecting Add in the Server Group section. You can add more starting points by selecting Add in the Starting Points section. In the Filter Definition section, you can add or delete, exclude or include certain types of files as well as change the order the filters for these files are applied.

Table 5-4 contains the attributes for the edit page. The table contains three columns: the first column identifies the attribute, the second column provides the default value for the attribute, and the third column describes the attribute.

Table 5-4 Robot Sites Edit Attributes

Attribute	URL/ Domain	Default Value	Description
Site Nickname	URL/D	Site entered - www.sesta.com	
Checkbox to select site for deletion or verification	URL/D	Unchecked	Unchecked--not selected Checked--selected

Table 5-4 Robot Sites Edit Attributes *(Continued)*

Attribute	URL/ Domain	Default Value	Description
Server Group - Name	URL	URL - www.sesta.com	Is either a single server or a part of a single server. The entry must include the full host name. If you specify just a host name, the site is limited to that host. If you provide directory information in addition to the host name, the site is defined as only that directory and any of its subdirectories.
Domain Suffix	D	Domain entered - *.sesta.com	Includes all servers within a domain, such as *.sesta.com.
Port	URL/D	80 for URL; blank for Domain	If the site you are searching uses a different port, enter it here.
Type	URL	Web Server	Web Server, File Server, FTP Server, Secure Web Server
Allowed Protocols	D	Checkboxes all checked	Checkboxes for http, file, ftp, https
Starting Points- Checkbox to select site for deletion	URL/D	Unchecked	Unchecked--not selected Checked--selected
Starting Points- URL	URL/D	http:// URL:80	URL or domain
Starting Points - Depth	URL/D	10	1 - this URL only 2 - this URL and first links 3-10 unlimited
Filter Definition - Checkbox to select file type for deletion	URL/D	Unchecked	Unchecked - not selected Checked - selected
Filter Definitions	URL/D	In this order, the defaults are Archive Files; Audio Files; Backup Files; Binary Files; CGI Files; Image Files; Java, Javascript, Style Sheet Files; Log Files; Revision Control Files; Source Code Files; Temporary Files; Video Files.	The possible choices are Archive Files; Audio Files; Backup Files; Binary Files; CGI Files; Image Files; Java, Javascript, Style Sheet Files; Log Files; Power Point Files; Revision Control Files; Source Code Files; Temporary Files; Video Files; Spreadsheet Files; Plug-in Files; Lotus Domino Documents; Lotus Domino OpenViews; System Directories (UNIX); System Directories (NT).
Comment	URL/D	Blank	Text field that describes the site to you. It is not used by the robot.

Table 5-4 Robot Sites Edit Attributes *(Continued)*

Attribute	URL/ Domain	Default Value	Description
DNS Translation	URL	Blank	The DNS translation modifies the URL and the way it is crawled by replacing a domain name or alias with a cname. Format: alias1->cname1,alias2->cname1

Filters

The initial page in this section shows all the defined filter rules and the site definitions that use them. Each filter name is proceeded by a checkbox to select that document type and two radio buttons to turn the Filter Rule On and Off. If a checkbox is checked, the filter is selected and can be deleted. You can add a new filter by selecting New. The new filter page is an abbreviated Edit page, requiring only a Nick Name and one rule. Another option is selecting the Edit link, which takes you to a page where you define the rules for that file type or what that filter does. Each rule is made up of a drop down list of Filter Sources, a drop down list to Filter By, and a text box to enter the filter string specifics in.

Table 5-5 contains the Robot Filter Edit attributes. The table contains three columns: the first column identifies the attribute, the second column provides the default value for the attribute, and the third column describes the attribute.

Table 5-5 Robot Filter Edit Attributes

Attribute	Default Value	Description
Filter Name	Prompts for new name. File name of the file type you choose to edit.	A descriptive name that reflects the type of file the filter applies to.
Drop down list of Filter Sources	URL for new filter. Displays previously chosen information for that particular file type.	URL, protocol, host, path, MIME type
Drop down list of positions	is for new filter. Displays previously chosen information for that particular file type. For example, Binary Files ends with exe.	is, contains, begins with, ends with, regular expression
Text box for type (directory, protocol, file extensions) specifics	Blank for new filter. Displays previously entered information for that particular file type. For example, Temporary Files contains /tmp/.	In this text box, list what you want to match. What would match in this example - http://docs.sesta.com/manual.html protocol is http; host contains sesta; file ends with html.

Table 5-5 Robot Filter Edit Attributes *(Continued)*

Attribute	Default Value	Description
Description	Prompts for new description. Displays previously entered description for that particular file type.	Describe the filter rule for yourself. The robot does not use it.
New Site	True (checked) for new filter. Displays previously chosen value for that particular file type.	Use this as one of the default filters when creating new sites. If you do not check this, you can still add this filter to a new site by editing the site on the Robot, Sites page.
By Default	Nothing selected for a new filter. Default selected previously for a defined file type.	Exclude documents matching this filter. Include documents matching this filter. Selection for a new filter does not affect existing site definitions. To use your new filter on an existing site, you must add it by editing the site on the Robot, Sites page.
Deployment	Lists the sites that use this filter.	

Crawling

The settings on this page control the robot's operational parameters and defaults. It is divided into these sections: Speed, Completion Actions, Logfile Settings, Standards Compliance, Authentication Parameters, Proxying, Advanced Settings and Link Extraction.

Table 5-6 contains the Robot Crawling attributes. The table contains three columns: the first column identifies the attribute, the second column provides the default value for the attribute, and the third column describes the attribute.

Table 5-6 Robot Crawling Attributes

Attribute	Default Value	Description
Server Delay	No Delay	No Delay (default), 1 second, 2 seconds, 5 seconds, 10 seconds, 30 seconds, 1 minute, 5 minutes.
Maximum Connections - Max concurrent retrieval URLs	8	1, 2, 4, 8 (default), 10, 12, 16, 20.
Maximum Connections per Site	2	(no limit), 1, 2, 4, 8, 10, 12, 16, 20.

Table 5-6 Robot Crawling Attributes *(Continued)*

Attribute	Default Value	Description
Send RDs to Indexing every	30 minutes	3 minutes, 5 minutes, 10 minutes, 15 minutes, 30 minutes (default), 1 hour, 2 hours, 4 hours, 8 hours.
Script to Launch	nothing (default)	nothing (default). For sample files, see the `cmdHook` files in the /opt/SUNWps/samples/robot directory (for the default installation).
After Processing all URLs	go idle (default)	go idle (default), shut down, start over.
Contact Email	user@domain	Enter your own.
Log Level	1 - Generation	0 Errors only; 1 Generation (default); 2 Enumeration, Conversion; 3 Filtering; 4 Spawning; 5 Retrieval
User Agent	SunJavaSystemRobot/6.0	Version of the search server.
Ignore robots.txt protocol	False (unchecked)	Some servers have a robot.txt file that says robots do not come here. If your search robot encounters this file on a site and this attribute is false, it does not search the site. If this attribute is true, the robot ignores the file and searches the site.
Perform Authentication	Yes	Yes No
Robot Username	anonymous	Robot uses the anonymous user name to gain access to a site.
Password	user@domain	Frequently a site that allows anonymous users requires a email address as a password. This address is in plain text.
Proxy Username	anonymous	Robot uses the anonymous user name to gain access to a site.
Password	user@domain	Frequently a site that allows anonymous users requires an email address as a password. This address is in plain text.
Proxy Connection Type	Direct Internet Connection	Direct Internet Connection, Proxy--Auto Configuration, Proxy--Manual Configuration
Auto Proxy Configuration Type	Local Proxy File	Local Proxy File, Remote Proxy File

Table 5-6 Robot Crawling Attributes *(Continued)*

Attribute	Default Value	Description
Auto Proxy Configuration Location	Blank	The auto proxy has a file that lists all the proxy information needed. An example of a local proxy file is robot.pac. An example of a emote proxy file is http://proxy.sesta.com:8080/proxy.pac
Manual Configuration HTTP Proxy	Blank	Format: server1.sesta.com:8080 These three manual configuration values are put in the robot.pac file in the `/var/opt/SUNWps/https-servername/portal/config` directory.
Manual Configuration HTTPS Proxy	Blank	This manual configuration value is put in the robot.pac file. Format: server1.sesta.com:8080
Manual Configuration FTP Proxy	Blank	This manual configuration value is put in the robot.pac file. Format: server1.sesta.com:8080
Follow links in HTML	True (checked)	Extract hyperlinks from HTML
maximum links	1024	Limits the number of links the robot can extract from any one HTML resource. As the robot searches sites and discovers links to other resources, it could conceivably end up following huge numbers of links a great distance from its original starting point.
Follow links in plain text	False (unchecked)	Extract hyperlinks from plain text.
maximum links	1024	Limits the number of links the robot can extract from any one text resource.
Use Cookies	False (unchecked)	If checked, the robot uses cookies when it crawls. Some sites require the use of cookies in order for them to be navigated correctly. The robot keeps its cookies in a file called cookies.txt in the robot state directory. The format of cookies.txt is the same format as used by the Netscape™ Communicator browser.

Table 5-6 Robot Crawling Attributes *(Continued)*

Attribute	Default Value	Description
Use IP as Source	True (checked)	In most cases, the robot operates only on the domain name of a resource. In some cases, you might want to be able to filter or classify resources based on subnets by Internet Protocol (IP) address. In that case, you must explicitly allow the robot to retrieve the IP address in addition to the domain name. Retrieving IP addresses requires an extra DNS lookup, which can slow the operation of the robot. If you do not need this option, you can turn it off to improve performance.
Smart Host Heuristics	False (unchecked)	If checked, the robot converts common alternate host names used by a server to a single name. This is most useful in cases where a site has a number of servers all aliased to the same address, such as www.sesta.com, which often have names such as www1.sesta.com, www2.sesta.com, and so on. When you select this option, the robot will internally translate all host names starting with `wwwn` to `www`, where `n` is any integer. This attribute only operates on host names starting with wwwn. This attribute cannot be used when CNAME resolution is OFF (false).

Table 5-6 Robot Crawling Attributes *(Continued)*

Attribute	Default Value	Description
Resolve hostnames to CNAMEs	False (unchecked)	If checked, the robot validates and resolves any host name it encounters into a canonical host name. This allows the robot to accurately track unique RDs. If unchecked, the robot validates host names without converting them to the canonical form. So you may get duplicate RDs listed with the different host names found by the robot. For example, devedge.sesta.com is an alias for developer.sesta.com. With CNAME resolution on, a URL referenced as devedge.sesta.com is listed as being found on developer.sesta.com. With CNAME resolution off, the RD retains the original reference to devedge.sesta.com. Smart Host Heuristics cannot be enabled when CNAME resolution is OFF (false).
Accepts commands from ANY host	False (unchecked)	Most robot control functions operate through a TCP/IP port. This attribute controls whether commands to the robot must come from the local host system (false), or whether they can come from anywhere on the network (true). It is recommended that you restrict direct robot control to the local host (false). You can still administer the robot remotely through the Administration Console.
Default Starting Point Depth	10	1- starting points only, 2- bookmark style, 3-10, unlimited. Default value for the levels of hyperlinks the robot traverses from any starting point. You can set the depth for any given starting point by editing the site on the Robot, Sites page.
Work Directory	`/var/opt/SUNWps/https-servernamefull/portal/tmp`	Full pathname of a temporary working directory the robot can use to store data. The robot retrieves the entire contents of documents into this directory, often many at a time, so this space should be large enough to handle all of those at once.

Table 5-6 Robot Crawling Attributes *(Continued)*

Attribute	Default Value	Description
State Directory	`/var/opt/SUNWps/https-servernamefull/portal/robot`	Full pathname of a temporary directory the robot uses to store its state information, including the list of URLs it has visited, the URL pool, and so on. This database can be quite large, so you might want to locate it in a separate partition from the Work Directory.

Indexing

The robot searches sites and collects documents based on the filters you have selected. The documents collected are in many different formats. To make them uniform and easily readable they need to be in one format, which is HTML. This page controls some of the parts that go into each resource description.

Table 5-7 contains the Robot Index attributes. The table contains three columns: the first column identifies the attribute, the second column provides the default value for the attribute, and the third column describes the attribute.

Table 5-7 Robot Index Attributes

Attribute	Default Value	Description
Full Text or Partial Text	Partial Text	Full uses the complete document in the resource description. Partial text only uses the specified number of bytes in the resource description.
extract first # bytes	4096	Enter the number of bytes.
Extract Table Of Contents	True (checked)	True includes the Table of Contents in the resource description.
Extract data in META tags	True (checked)	True includes the META tags in the resource description.

Table 5-7 Robot Index Attributes *(Continued)*

Attribute	Default Value	Description
Document Converters	All checked (true); if false, that type of document cannot be indexed.	Adobe PDF
		Corel Presentations
		Corel Quattro Pro
		FrameMaker
		Lotus Ami Pro
		Lotus Freelance
		Lotus Word Pro
		Lotus 1-2-3
		Microsoft Excel
		Microsoft Powerpoint
		Microsoft RTF
		Microsoft Word
		Microsoft Works
		Microsoft Write
		WordPerfect
		StarOffice™ Calc
		StarOffice™ Impress
		StarOffice™ Writer
		XyWrite
Converter Timeout	600	Time in seconds allowed for one document to be converted to HTML. If this time is exceeded, the URL is excluded.

Simulator

This page is a debugging tool that performs a partial simulation of robot filtering on a URL. You can type in a new URL to check. It checks the URL, DNS translations (including Smart Host Heuristics), and site redirections. It does not check the contents of the document specified by the URL, so it does not detect duplications, MIME types, network errors, permissions, and the like. The simulator indicates whether the listed sites would be accepted by the robot (ACCEPTED) or not (WARNING).

Table 5-8 contains the Robot Simulator properties. The table contains three columns: the first column identifies the attribute, the second column provides the default value for the attribute, and the third column describes the attribute.

Table 5-8 Robot Simulator Properties

Attribute	Default Value	Description
URL	URLs you have already defined and one blank text box.	You can check access to a new site by typing its URL in the blank text box. This checks to see if the new site accepts crawling. Format http://www.sesta.com:80/
Check for DNS aliases	True (checked)	True (checked) checks for number of servers aliased to the same address.
Check for Server Redirects (302)	True (checked)	True (checked) checks for any server redirects.

Site Probe

This page is a debugging tool that checks for DNS aliases, server redirects, and virtual servers. This tool returns information about site but does not test its acceptance of crawling.

Table 5-9 contains the Robot Site Probe attributes. The table contains three columns: the first column identifies the attribute, the second column provides the default value for the attribute, and the third column describes the attribute.

Table 5-9 Robot Site Probe Attributes

Attribute	Default Value	Description
Site	Blank	Type in URL in format http://www.sesta.com:80
Show advanced DNS information	False (unchecked)	True (checked) displays more information about the site including IP addresses.

Schedule

This page is where you set up the automatic search schedule for the robot. Table 5-10 contains the Robot Schedule attributes. The table contains three columns: the first column identifies the attribute, the second column provides the default value for the attribute, and the third column describes the attribute.

Table 5-10 Robot Schedule Attributes

Attribute	Default Value	Description
Start Robot Time in hours and minutes	00:00	This is the time that the robot starts to search.
Days	none selected	Sun, Mon, Tue, Wed, Thu, Fri, or Sat Check at least one day.
Stop Robot Time in hours and minutes	00:00	If you plan to run the robot continuously, it is recommended that you stop and restart it at least once per day. This gives the robot a chance to release resources and reinitialize itself.
Days	none selected	Sun, Mon, Tue, Wed, Thu, Fri, or Sat

Chapter 6

Search Attributes: Database

The Database attributes are divided as follows:

- Management
- Import Agents
- Resource Descriptions
- Schema
- Analysis
- Schedule

NOTE To partition the database, you must use the command line function because stopping the search server is required.

Management

The initial Management page lists the available databases. You can create a new one, reindex, purge, or expire an existing one. Use the checkbox to select a database on which to perform an action. Use the small icons above the checkbox to select or deselect all the databases. When you select Reindex, Purge or Expire, a prompt confirming that you want to perform the action with a list of database names displays. To perform the action, select OK.

You should reindex the database if you have edited the schema to add or remove an indexed field (as author), or if a disk error has corrupted the index. You need to restart the server after you change the schema.

Because the time required to reindex the database is proportional to the number of RDs in the database, a large database should be reindexed when the server is not in high demand.

When you purge the contents of the database, disk space used for indexes will be recovered, but disk space used by the main database will not be recovered; instead, it is reused as new data is added to the database.

Expiring a database deletes all RDs that are deemed out-of-date. It does not decrease the size of the database. By default, an RD is scheduled to expire in 90 days from the time of creation.

You can also edit the database by selecting the Edit link which takes you to a page where you define the database attributes.

Table 6-1 lists the Database Management attributes. The table contains three columns: the first column identifies the attribute, the second column provides the default value for the attribute, and the third column describes the attribute.

Table 6-1 Database Management Attributes

Attribute	Default Value	Description
Name	Default	Name for the database used by Search.
Title	Blank	A title for the database.
Description	Blank	Describe the database for yourself.

Import Agents

Import agents are the processes that bring resource descriptions from other servers or databases and merge them into your search database.

The initial Import page lists the available import agents. You can create a new one, or run, edit or delete an existing one. Use the checkbox to select an agent to delete. Use the small icons above the checkbox to select or deselect all import agents. Use the radio buttons to turn an Agent Action On or Off. To schedule the import agents, select Schedule on the lower menu bar.

If you choose to edit or modify an existing import agent or create a new one, the following attributes are displayed.

Table 6-2 on page 65 lists the Database Import Agent attributes. The table contains three columns: the first column identifies the attribute, the second column provides the default value for the attribute, and the third column describes the attribute.

Table 6-2 Database Import Agent Attributes

Attribute	Default Value	Description
Charset	Blank for new	Specifies the character set of the input SOIF stream. For example, ISO8859-1, UTF-8, UTF-16. Character sets ISO8859-1 through ISO8859-15 are supported.
Import From	Local File	Select either Local File or Search Server (if one is enabled).
Local File Path	Blank for new	Gives the full path name of local file that contains valid resource descriptions in SOIF (Summary Object Interchange Format). This can be a file on another server, as long as the path is addressable as if it were locally mounted.
Database Name	Default	Name of the destination database.
Remote Server	Blank for new	Gives the URL of the search server to retrieve resource descriptions from; format http://www.sesta.com:80
Instance Name	Blank for new	Server instance name used by the search server. You can find this instance name in the Server Preferences for the server you are importing from. Value must be 3.01C or 3.01C SP1.
Search URI	Blank for new	Enter full path and file names. Use /portal/search.
Is Compass Server 3.01X?	False (unchecked)	Is the server you are importing from a Compass Server 3.01X?
Enable SSL	False (unchecked)	If this is a server-to-server transaction, select if the servers should use the SSL (Secure Sockets Layer) protocol.
Authentication	None (default)	None (default) or Use User/Password This specifies how the import agent should identify itself to the system it imports from. By default, no authentication is used. If the server you want to import from requires authentication, you can specify a user name and password for the import agent to use. Importing from 3.01C does not require authentication. Importing data from 3.01C SP1 requires authentication.

Table 6-2 Database Import Agent Attributes *(Continued)*

Attribute	Default Value	Description
User	Blank for new or none	If you selected Use User/Password, enter a user.
Password	Blank for new or none	If you selected Use User/Password, enter a password (shown as *).
Content Transfer	Use Incremental Gathering of Full Contents (default)	Choice of Use Incremental Gathering of Full Contents (default) or Use Search Query These specify which resource descriptions to import from the source. By default, an import agent asks for all resource descriptions added or changed since its last import from the same source. The search query specifies that the import agent should request only certain resource descriptions from the source. This is much the same way that users request listings of resources from the search database. Use Scope, View-Attributes and View-Hits fields to specify the query. Use Collect All RDs to collect new resource descriptions since the last run and remove the timestamp in Newest Resource Description.
Scope	Blank for new	The text of the query. The query syntax is identical to that used for end-user queries from the server.
View-Attributes	Blank for new	Lists which fields (not case sensitive) you want to import in each resource description. For example, title and author. The default is all.
View-Hits	Blank for new	The maximum number of matching resource descriptions to import. If no hits are specified, it defaults to 20.

Table 6-2 Database Import Agent Attributes *(Continued)*

Attribute	Default Value	Description
Agent Description	Blank for new	Appears in the list of available import agents on the initial Import page. It is ignored by the program. If this field is blank, the Resource Description Source file name or server name is used to identify the import agent. Note here if user name and password are needed.
Newest Resource Description	Blank for new	The date of the creation of the newest resource description previously imported by this import agent. This date is used by the Use Incremental Gathering of Full Contents option to determine which resources are new and should be imported.
Network Timeout in seconds	Blank for new	Specifies the number of seconds the import agent will allow before timing out the connection over the network. You can adjust this to allow for varying network traffic and quality.

Resource Descriptions

The initial Resource Descriptions page allows you to search the Resource Descriptions in the database. For example, you can correct a typographical error in an RD or manually assign RDs discovered by the robot to categories.

Table 6-3 lists the Resource Descriptions attributes. The table contains three columns: the first column identifies the attribute, the second column provides the default value for the attribute, and the third column describes the attribute.

Table 6-3 Resource Descriptions Attributes

Attribute	Default Value	Description
Search For	All RDs	All RDs, Uncategorized RDs, Categorized RDs, RDs by category, Specific RD by URL, RDs that contain???
Text box	Blank	Enter a unique text string to identify the RDs searched for. Use with the RDs by category, Specific RD by URL, and RDs that contain attribute values.
Database	Default	Name of the database to search.

Table 6-3 Resource Descriptions Attributes *(Continued)*

Attribute	Default Value	Description
Select Category		Browse and select a category from the category tree.
Delete		Delete one or more selected RDs that are returned from an RD search.
Next		Display the next set of RDs returned from an RD search
Previous		Display the previous set of RDs returned from an RD search
Edit Selected		Edit the attributes of one or more RDs that are returned from an RD search.
Edit All		Edit the attributes of the current set of RDs that are returned from an RD search.

To limit the search by category, select Select Category. A Category Editor page displays allowing you to specify the category from the taxonomy for the search. You can specify the category in the Selected Category text box or browse the taxonomy to select it. After specifying the category, select OK to return to the RD search page.

Table 6-4 lists the Category Editor attributes. The table contains three columns: the first column identifies the attribute, the second column provides the default value for the attribute, and the third column describes the attribute.

Table 6-4 Category Editor Attributes

Attribute	Default Value	Description
Selected Categories	Blank	Text field that displays the selected categories
Expand All		Expands the taxonomy so that all entries in the hierarchy display for browsing.
Collapse All	Blank	Collapses the taxonomy so that only categories within the first two levels of the hierarchy display for browsing.
Categories per page	25	Drop down list of the number of categories to display per page. Values are 25, 50, 100, 250, 500, and all.

A successful search displays the Number of RDs found and a list box with the RDs found. After clicking on the Edit link of an RD, the following attributes, which you can edit, and partial text of the RD are displayed. All these attributes except Classification are set to editable in the Database / Schema page.

Table 6-5 lists the Database RD Editable attributes. The table contains three columns: the first column identifies the attribute, the second column provides the default value for the attribute, and the third column describes the attribute.

Table 6-5 Database RD Editable Attributes

Attribute	Default Value	Description
Author	Blank	Author(s) of the document.
Author e-mail	Blank	Email address to contact the Author(s) of the document.
Classification	Category name of selected RD.	Category name if classified; No Classification if not classified.
ReadACL	Blank	Related to document level security.
Content-Charset		Content-Charset information from HTTP Server.
Content-Encoding	Blank	Content-Encoding information from HTTP Server.
Content-Language	Blank	Content-Language information from HTTP Server.
Content-Length	Blank	Content-Length information from HTTP Server.
Content-Type	Blank	Content-Type information from HTTP Server.
Description	Description from the selected RD.	Description from RD.
Expires	Valid date.	Date on which resource description is no longer valid.
Full-Text	Blank	Entire contents of the document.
Keywords	Keywords, if any, from the selected RD.	Keywords taken from meta- tags.
Last-Modified	Last modification date	Date when the document was last modified.
Partial-text	Partial text of the document	Partial selection of text from the document
Phone	Blank	Phone number for Author contact

Table 6-5 Database RD Editable Attributes *(Continued)*

Attribute	Default Value	Description
Title	Title of the selected RD.	Title of RD
URL	Blank	Uniform Resource Locator for the document

Schema

The schema determines what information is in a resource description and what form that information is in. You can add new attributes or fields to an RD and set which ones can be edited and which ones can be indexed. When importing new RDs, you can convert schemas embedded in new RDs into your own schema.

Table 6-6 lists the Database Schema Edit attributes. The table contains three columns: the first column identifies the attribute, the second column provides the default value for the attribute, and the third column describes the attribute.

Table 6-6 Database Schema Edit Attributes

Attribute	Description
Author	Author(s) of the document.
Author-EMail	Email address to contact the Author(s) of the document.
Content-Charset	Content-Charset information from HTTP Server.
Content-Encoding	Content-Encoding information from HTTP Server.
Content-Language	Content-Language information from HTTP Server.
Content-Length	Content-Length information from HTTP Server.
Content-Type	Content-Type information from HTTP Server.
Description	Brief one-line description for document.
Expires	Date on which resource description is no longer valid.
Full-Text	Entire contents of the document.
Keywords	Keywords that best describe the document.
Last-modified	Date when the document was last modified.
Partial-Text	Partial selection of text from the document.
Phone	Phone number for Author contact.
ReadACL	Used by Search servers to enforce security.

Table 6-6 Database Schema Edit Attributes *(Continued)*

Attribute	Description
Title	Title of the document.
URL	Uniform Resource Locator for the document
Aliases Name Description	When you import new RDs, you can convert schemas embedded in new RDs into your own schema. You would use this conversion when there are discrepancies between the names used for fields in the import database schema and the schema used for RDs in your database. An example would be if you imported RDs that used Writer as a field for the author and you used Author in your RDs as the field for the author. The conversion would be Writer to Author, so you would enter Writer in this text box.
Data Type	Defines the data type.
Editable	If true (checked), the selected attribute (field) appears in the Database RD Editor, so you can change its values. Description, Keywords, Title and ReadACL are editable.
Indexable	If true (checked), the selected attribute (field) can be used as a basis for indexing. Author, Title and URL appear in the menu in the Advanced Search screen for the end user. This allows end users to search for values in those particular fields. Author, Expires, Keywords, Last Modified, Title, URL and ReadACL can be used as the basis for indexing.
Score Multiplier	A weighting field for scoring a particular element. Any positive value is valid.

Analysis

The Analysis page shows a sorted list of all sites and the number of resources from that site currently in the search database. Select Update Analysis to update the analysis on file.

Table 6-7 lists the Database Analysis attributes. The table contains three columns: the first column identifies the attribute, the second column provides the default value for the attribute, and the third column describes the attribute.

Table 6-7 Database Analysis Attributes

Attribute	Default Value	Description
Total number of RDs	Current number of RDs in database.	Lists current total number of resource descriptions in the database.
Number of servers	Current number of servers that the database is partitioned across.	The database can be partitioned and placed on a number of servers.

Table 6-7 Database Analysis Attributes *(Continued)*

Attribute	Default Value	Description
Site	URL or domain that the robot has successfully searched.	A URL or domain that has added resource descriptions to the database.
Number of RDs	Current number of RDs from that site.	Lists current number of RDs from that site.
Type	Type of RD	Resource descriptions can be of many different types, for example, http.
Percentage	Type of RD/ Total number of RDs	Percentage of this type of document compared to the total number of resource descriptions.

Schedule

This page is where you set up the schedule for running the import agents. Table 6-8 lists the Database Import Schedule attributes. The table contains three columns: the first column identifies the attribute, the second column provides the default value for the attribute, and the third column describes the attribute.

Table 6-8 Database Import Schedule Attributes

Attribute	Default Value	Description
Start Import Time in hours and minutes	00:00	Time that the import agent starts to import.
Days	none selected	Sun -Sat Check at least one day.

Chapter 7

Search Attributes: Categories

This chapter contains the following sections:

Overview

End users interact with the search database in two distinct ways: They can type direct queries to search the database, or they can browse through the database contents using a set of categories you design. You assign resources in a search database to categories to clarify complexity. If a large number of items are in the database, it is helpful to group related items together. Your primary concern in setting up your categories should be usability so that end users can more quickly locate specific kinds of items.

The search server uses a hierarchy of categories called a taxonomy. The term taxonomy in general describes any system of categories. In the context of a networked resource database such as the search server database, it describes any method you choose of categorizing network resources to facilitate retrieval.

Category Editor

The Category Editor page displays a listing the categories in the taxonomy allowing you to browse the categories. After browsing to the category, you may select the category link to bring up the Classification Rules Editor to set up the Robot collections under specific categories.

Table 7-1 lists the Category Editor attributes. The table contains three columns: the first column identifies the attribute, the second column provides the default value for the attribute, and the third column describes the attribute.

Table 7-1 Category Editor Attributes

Attribute	Default Value	Description
Expand All		Expands the taxonomy so that all entries in the hierarchy display for browsing.
Collapse All		Collapses the taxonomy so that only categories within the first two levels of the hierarchy display for browsing.
Reindex		Reindexes the database. If you have just created your taxonomy, you need to index the database to make category search available to your end users. If you have changed your categories, you need to reindex the database to make it up-to-date. Save the categories tree before you reindex the database. Load the new taxonomy.
Categories per page		Drop down list of the number of categories to display per page. Values are 25, 50, 100, 250, 500, and all.
Name	25	Displays the name of the selected category to edit
Description	Blank	Displays the description of the selected category.
Matching Rule	Blank	Displays the matching rule to use with the selected category.
Update		Updates the category definition.
Add as a Subcategory (PER BUG ID 5070893)		Adds the category as a child or a sibling.

Classification Rules Editor

After you set up the categories for your database, Click New to set or change the rules the robot for selected categories to assign resources to categories. Table 7-2 lists the Categories Classification Rules Editor attributes. The table contains three columns: the first column identifies the attribute, the second column provides the default value for the attribute, and the third column describes the attribute.

Table 7-2 Categories Classification Rules Editor Attributes

Attribute	Default Value	Description
Source	Author	The valid attributes include: • Author • Author-EMail • Content-Charset • Content-Encoding • Content-Language • Content-Length • Content-Type • Description • Expires • Full-Text • Keywords • Last-modified • Partial-Text • Phone • ReadACL • Title • URL • Host • Protocol • IP • Path • Type
Method	is	is, contains, begins with, ends with, regular expression
Criteria	Blank	Specifies the criteria for the rule.
Classification	Blank	Category to in which to classify the RD if the rule conditions are met. Type the category or use the Select Category Edit page to browse to it.

Chapter 8

Search Attributes: Reports

This chapter contains the following sections:

Introduction

The Reports section allows you to monitor your search server. You can see a summary of its activity: what sites were searched, what URLs were excluded and why, detailed information about URLs visited by the robot, and what your end users are interested in.

Starting Points

The robot will visit all the enabled sites every time it starts. Table 8-1 on page 78 lists the Reports Starting Points attributes. The table contains three columns: the first column identifies the attribute, the second column provides the default value for the attribute, and the third column describes the attribute.

Table 8-1 Reports Starting Points Attributes

Attribute	Default Value	Description
Enabled	Current value of site.	Yes or No. This is set on the Robot/ Sites page.
Starting Point	Chosen URL:80	Link brings up chosen URL.
in site definition	Chosen URL	Links to Robot/ Sites edit page.
Depth	Lists selected level of search.	1-n Set on the Robot/ Sites edit page.

Excluded URLs

This page shows a list of robot runs. To display a list of reasons URLs were excluded, select a robot run to examine, select View Selected, then select one of the Reasons for Exclusion. Displayed is a list of the excluded URLs for that reason. Duplicate and warning exclusions have been removed.

Table 8-2 lists the Reports Excluded URLs attributes. The table contains three columns: the first column identifies the attribute, the second column provides the default value for the attribute, and the third column describes the attribute.

Table 8-2 Reports Excluded URLs Attributes

Attribute	Default Value	Description
Log	Lists log from most recent run.	Lists all run logs available.
Count	Numbers	List of numbers with reasons for exclusion.
Reason for Exclusion	List of reasons sites have not been allowed. Each reason is linked to a list of all the URLs that were excluded for that reason.	Filter rules, file not found, site not allowed, protocol not allowed, errors, duplication are some of the reasons URLs were excluded.

Robot Advanced Reports

This page gives you access to a number of different reports from the robot. Select from a drop down list to get information for chosen report to show up. The Refresh button gets the current information.

Table 8-3 lists the Reports Robot Advance Reports attributes. The table contains three columns: the first column identifies the attribute, the second column provides the default value for the attribute, and the third column describes the attribute.

Table 8-3 Reports Robot Advance Reports Attribute

Attribute	Default Value	Description
Advanced Robot Reports	Version	Version, DNS Cache Dump, Performance, Servers Found-All, Server Found-RDM, Status-Current Configuration, Status -Database (internal), Status-Libnet, Status -Modules, Status-Overview, URLs-ready for extraction, URLs-ready for indexing, URLs- waiting for filtering (URL pool), URLs- waiting for indexing, all reports.

Log Files

This page allows you to view the entries or specific lines from a log file. Drop down list of log files. Enter the number of lines you want to be displayed when you select View button.

Table 8-4 lists the Reports View Log Files attributes. The table contains three columns: the first column identifies the attribute, the second column provides the default value for the attribute, and the third column describes the attribute.

Table 8-4 Reports View Log Files Attributes

Attribute	Default Value	Description
View this logfile	Excluded URLs (filter)	Excluded URLs (filter), RD Manager (rdmgr), RDM Server (rdmsvr), Robot Activities (robot), Search Engine (searchengine), User Queries (rdm).
Number of lines	25	A number you can enter to display the most current entries in the log file.

Popular Searches

This page allows you to see what users are searching for. The most frequent searches appear first in the report. Table 8-5 lists the Reports Popular Searches attributes. The table contains three columns: the first column identifies the attribute, the second column provides the default value for the attribute, and the third column describes the attribute.

Table 8-5 Reports Popular Searches Attribute

Attribute	Default Value	Description
Exclude Browsing	False (unchecked)	False (unchecked) includes what categories users browse in. True (checked) excludes browsing statistics.

Chapter 9

Subscriptions Attributes

The Subscriptions Service consists of dynamic, organization and user attributes.

- Subscriptions Dynamic Attributes
- Subscriptions Organization Attributes
- Subscriptions User Attributes

Subscriptions Dynamic Attributes

Table 9-1 describes the dynamic attributes for the Subscriptions Service (when viewed from the Service Configuration or Identity Management tab).

The table contains three columns: the first column identifies the attribute, the second column provides the default value for the attribute, and the third column describes the attribute.

Table 9-1 Subscriptions Service - Dynamic Attributes

Attribute	Default Value	Description
Maximum number of Categories subscriptions	5	Specifies the maximum number of subscriptions on categories that can be defined and stored in the Identity Server.
Maximum number of Discussion subscriptions	5	Specifies the maximum number of subscriptions on discussions that can be defined and stored in the Identity Server.
Maximum number of Saved searches	5	Specifies the maximum number of saved search subscriptions that can be defined and stored in the Identity Server.

Table 9-1 Subscriptions Service - Dynamic Attributes *(Continued)*

Attribute	Default Value	Description
Conflict Resolution Level	Highest	Sets the conflict resolution level for the Subscriptions service template used to resolve conflicts when multiple Subscriptions templates are merged. There are seven conflict resolution settings available ranging from Highest to Lowest. Do not confuse this setting with the display profile document priority. The display profile document priority is a numeric value that is set in the XML file with the priority= syntax tag. When a merge occurs, it starts with lowest display profile priority document (lowest number) and proceeds by increasing priority number until it arrives at the user (base), the highest priority display profile. When an attribute conflict occurs, the attribute on the template set with the highest conflict resolution level is returned.

Subscriptions Organization Attributes

Table 9-2 and Table 9-3 describes the organization attributes for the Subscriptions Service when Services is selected under the Identity Management tab.

The tables contain three columns: the first column identifies the attribute, the second column provides the default value for the attribute, and the third column describes the attribute

Start Profiler

This page is where you set up the automatic start time for the profiler.

Table 9-2 Subscription Service - Organization Attributes - Start Profiler

Attribute	Default Value	Description
Times	00:00	This is the time when the profiler starts to search.

Table 9-2 Subscription Service - Organization Attributes - Start Profiler *(Continued)*

Attribute	Default Value	Description
Days	none selected	Sun, Mon, Tue, Wed, Thu, Fri, or Sat

Stop Profiler

This page is where you set up the automatic stop time for the profiler.

Table 9-3 Subscription Service - Organization Attributes - Stop Profiler

Attribute	Default Value	Description
Time		This is the time when the profiler stops.
Days	none selected	Sun, Mon, Tue, Wed, Thu, Fri, or Sat

Subscriptions User Attributes

Table 9-4 describes the user attributes for the Subscriptions Service when Users is selected under the Identity Management tab.

The table contains three columns: the first column identifies the attribute, the second column provides the default value for the attribute, and the third column describes the attribute.

Table 9-4 Subscriptions Service - User Attributes

Attribute	Default Value	Description
Category Subscriptions	No default subscriptions	This field defines the subscriptions details. The format is label|target category|scope|lapsed time|minimum rating|search server url|database where: • label is a free-form string • target category is the colon-separated string representation of the category ID. • scope is the search query criteria. • lapsed time is how far back in time content is the object of the subscription from the time of subscription is evaluated. Subscriptions evaluation is done when users access the Subscriptions channel. • minimum rating is the threshold rating above which subscription yields content. This field is a numeric value (-1,0,1,2,3) that corresponds to end user rating choices (Irrelevant, Routine, Interesting, Important, Must read). • search server url is the URL of the target search server. This file is URL encoded. • database is the database to be searched.

Table 9-4 Subscriptions Service - User Attributes *(Continued)*

Attribute	Default Value	Description
Discussion Subscriptions	No default subscriptions	This field defines the subscriptions details. The format is label|target discussion RD's URL|scope|lapsed time|minimum rating| search server url |target database where: • label is a free-form string • target discussion RD's URL is the space-separated string representation of the discussion ID. • scope is the search query criteria. • lapsed time is how far back in time content is the object of the subscription from the time of subscription is evaluated. Subscriptions evaluation is done when users access the Subscriptions channel. • minimum rating is the threshold rating above which subscription yields content. This field is a numeric value (-1,0,1,2,3) that corresponds to end user rating choices (Irrelevant, Routine, Interesting, Important, Must read). • search server url is the URL of the target search server. This file is URL encoded. • database is the database to be searched.

Table 9-4 Subscriptions Service - User Attributes *(Continued)*

Attribute	Default Value	Description
Saved Search Subscriptions	No default subscriptions	This field defines the subscriptions details. The format is label|scope|lapsed timel|minimum ratng|search server url|target database where: • label is a free-form string • scope is the search query criteria. • lapsed time is how far back in time content is the object of the subscription from the time of subscription is evaluated. Subscriptions evaluation is done when users access the Subscriptions channel. • minimum rating is the threshold rating above which subscription yields content. This field is a numeric value (-1,0,1,2,3) that corresponds to end user rating choices (Irrelevant, Routine, Interesting, Important, Must read). • search server url is the URL of the target search server. This file is URL encoded. • target database is the database to be searched.

Part II

Command Line Utilities

Chapter 10

deploy

This chapter contains the following sections:

Description

The `deploy` command packages the source files for the Portal Server web application files and deploys the package to the web container that hosts the portal server software.

The source files for the Portal Server are stored in the *PortalServer-base*`/web-src/` directory. The `WEB-INF/xml` subdirectory contains `web.xml` fragment files that are combined by the `deploy` command to form the `web.xml` file for the Portal Server web application. The corresponding sections of the `web.xml` fragment files are combined based on the alphabetic order of the `web.xml` fragment files. Once the final `web.xml` file is formed, the files in the *PortalServer-base*`/web-src/` directory are put into a web application archive (WAR) file using the `jar` command. This WAR file is deployed to a web container using the `deploy` command.

Syntax

PortalServer-base`/bin/deploy [redeploy]`

Subcommands

The `deploy` command takes the `redeploy` subcommand. If the `deploy` command is invoked without the `redeploy` option, it prompts for configuration information from standard input.

redeploy

Description

The `redeploy` subcommand specifies that the `deploy` command reuse the Uniform Resource Indicator (URI) and other information associated with the Portal Server web application from the current deployment.

Syntax

```
deploy redeploy
```

Chapter 11

pdeploy

This chapter contains the following sections

Description

The `pdeploy` is a command line tool that can be used to deploy and undeploy the portlet web application into the portal server.

The pdeploy command requires:

- A subcommand
- A user distinguished name and password to access the directory server
- A distinguished name to identify the LDAP node or the global option for the global level (the node where the portlets need to be added)
- The administrator password for the web container
- Portlet War File.

Some of the default settings used by the pdeploy command to deploy portlet applications are available in the PDConfig.properties file. This file is installed into the `/etc/opt/SUNWps/portlet` directory.

When the `pdeploy` command deploys the portlet application, it refers to the following parameters in the `PDConfigure.properties` file:

`logger.log.level=SEVERE`	By default, the log level is set to SEVERE. Valid values are ALL, OFF, INFO, WARNING, SEVERE.
`logger.file.dir=/var/opt/SUNWam/debug`	This parameter specifies the directory where the log file for the deployed portlet application is stored.
`validate_schema=true`	This parameter specifies whether schema validation should be performed during deployment.

Syntax

This section describes the `pdeploy` command syntax.

Short-Named Format

`pdeploy deploy -u` *uid* `-w` *password* `{-g|-d dn} -p` *webcontainerpassword* `-V -r` *rolesfile* `-f` *userinfofile* `-v -l` *warfile*

`pdeploy undeploy -u` *uid* `-w` *password* `{-g|-d dn} -p` *webcontainerpassword* `-V -v -l` *warfile*

Long-Named Format

`pdeploy deploy --runasdn` *uid* `--password` *password* `{--global|--dn dn} --wc_password` *webcontainerpassword* `--rolesfile` *rolesfile* `--userinfofile` *userinfofile* `--verbose --locale` *warfile*

`pdeploy deploy --help`

`pdeploy deploy --version`

`pdeploy undeploy --runasdn` *uid* `--password` *password* `{--global|--dn dn} --wc_password` *password* `--verbose --locale` *portletwebapp*

`pdeploy undeploy --help`

`pdeploy undeploy --version`

Subcommands

The pdeploy command takes these subcommands:

deploy	deploys the portlet application
undeploy	removes the portlet application

deploy

Description

If the subcommand is deploy, the pdeploy command deploys the portlet web application into the portal server. After this command completes, you can create channels based on the portlets defined in the deployed portlet web application.

Syntax

```
pdeploy deploy -u uid -w password {-g|-d dn} -p webcontainerpassword warfile
pdeploy deploy -h|--help
```

Options

The following table, which describes what options are supported, contains two columns: the first column lists the possible options for the deploy subcommand; the second column gives a brief description.

-v or --verbose	Generates debug messages.
-d or --dn	Specifies the distinguished name in the LDAP node to access the display profile document. The -d or -g option is required.
-f or --userinfofile	Specifies the file containing the user information mapping information.
-g or --global	Specifies the global level node in LDAP to access the display profile document. The -d or -g option is required.
--help	Prints help message to stdout.
-l or --locale	Prints the locale information.

-p or --wc_password	Specifies the web container password This option is required.
-r or --rolesfile	Specifies the file containing the Sun Java System Identity Server software and portlet application role mapping information.
-u or --runasdn	Specifies the distinguished name of the user to use to bind to the Sun Java System Directory Server. This option is required.
-V or --version	Generates version information.
-w or --password	Specifies the password of the distinguished name of the user used to bind to the Directory Server. This option is required.

The following table contains two columns. The first column lists the operands for the deploy subcommand; the second column gives a brief description.

warfile	Specifies the path to the WAR file.

Examples

Example 1 In the following example, the pdeploy command deploys the /tmp/SamplePortletApp.war into the portal server.

```
pdeploy deploy -u "uid=amAdmin,ou=people,o=sesta.com,o=isp" -w admin -p
sunone -g /tmp/SamplePortletApp.war
```

Example 2 Sometimes the portlet application defines logical roles in the portlet.xml file. During deployment, the logical roles need to be mapped to the actual roles defined in the system. To accomplish this, supply a role mapping file.

The role mapping file is expected to contain ActualRole=LogicalRole entries. The file supplied must follow Java™ property file format. For example:

```
cn—HRManager,dc—iplnaet,dc—com=Manager

cn—Emp,dc—iplnaet,dc—com=Employee
```

The following pdeploy command will provide the role mapping file for deploying the SamplePortletApp.war file into the portlet application.

```
pdeploy deploy -u "uid=amAdmin,ou=People,o=sesta.com,o=isp" -w admin -p
sunone -r /tmp/RoleMaps -g /tmp/SamplePortletApp.war
```

Example 3 Sometimes the portlet application will need access to the information associated with each user. During deployment, logical user information entry name must be mapped to the actual user information entry name defined in the system. To accomplish this during deployment, a user information entry map can be supplied.

The user information file is expected to contain `ActualEntryName=LogicalEntryName` entries. For example:

```
lastname=lname
```

```
firstname=fname
```

The following `deploy` command will provide the user information file for deploying the `SamplePortletApp.war` file into the portlet application.

```
pdeploy deploy -u "uid=amAdmin,ou=People,o=sesta.com,o=isp" -w admin -p
sunone -f /tmp/UserInfoMaps -g /tmp/SamplePortletApp.war
```

undeploy

Description

The `undeploy` subcommand removes the portlet application from the portal server. However, it does not remove all the channel definitions already created for portlets defined in the portlet web application. All channels associated with the portlet web application (being removed) must be manually removed.

Syntax

`pdeploy undeploy -u` *uid* `-w` *password* `{-g|-d dn} -p` *webcontainerpassword* `-v` *portletwebapp*

`pdeploy undeploy -h|--help`

Options

The following table describes what options are supported and contains two columns: the first column lists the possible options for the `undeploy` subcommand; the second column gives a brief description.

`-v or --verbose`	Generates debug messages.

-d or --dn	Specifies the distinguished name in the LDAP node to access the display profile document. The -d or -g option is required.
-g or --global	Specifies the global level node in LDAP to access the display profile document. The -d or -g option is required.
--help	Prints the help message to stdout.
-l or --locale	Provides locale information.
-p or --wc_password	Specifies the web container password This option is required.
-u or --runasdn	Specifies the distinguished name of the user to use to bind to the Directory Server. This option is required.
-V or --version	Generates version information.
-w or --password	Specifies the password of the distinguished name of the user used to bind to the Directory Server. This option is required.

The following table describes what operands are supported and contains two columns: the first column lists the possible operands for the undeploy subcommand; the second column gives a brief description.

portletwebapp	Specifies the name of the deployed portlet web application. Typically, it is the same name as the war file name without the .war extension.

Example

The following pdeploy command undeploys the portlet web application named SamplePortletApp from the portal server.

```
pdeploy undeploy -u "uid=amAdmin,ou=People,o=sesta.com,o=isp" -w admin -g
```

Chapter 12

dpadmin

This chapter contains the following sections:

Description

The `dpadmin` command enables display profile objects to be retrieved, added, modified, and removed from a display profile document by using the subcommands. All interactions with display profile objects must be in their native XML format. The dpadmin command can operate on a single display profile document only.

The `dpadmin` command requires:

- A subcommand (see Subcommands)
- A user distinguished name and password to access the directory server.
- A target display profile document. A distinguished name to identify the LDAP node or `--global` (`-g`) option for the global level display profile document.

NOTE A display profile document is uniquely identified by the `-d` or `-g` option:

global: `-g`

organization: `-d "dc=org,dc=com"`

sub-organization: `-d "o=sub-org,dc=org,dc=com"`

role: `-d "cn=rolename,dc=org,dc=com"`

user: `-d "uid=username,ou=people,dc=org,dc=com"`

Syntax

This section describes the `dpadmin` command syntax. You cannot mix long-named options with short ones in one command line.

Short-Named Format

`dpadmin list|merge|modify|add|remove [command-specific-options] -u` *uid* `-w` *password* `{-g|-d` *dn*`} [-l` *locale*`] [-r] [-b] [-V] [-h] [`*file*`]`

`dpadmin batch [-c] -f` *batch-script-filename* `[-l locale] [-b] [-h]`

Long-Named Format

`dpadmin list|merge|modify|add|remove [command-specific-options] --runasdn` *uid* `--password` *password* `{--global|--dn` *dn*`} [--locale` *locale*`] [--dryrun] [--verbose] [--version] [--help] [`*file*`]`

`dpadmin --version`

`dpadmin batch [--continue] --file` *batch-script-filename* `[--locale` *locale*`] [--verbose] [--help]`

Subcommands

The `dpadmin` command takes these subcommands:

list

Description

This subcommand retrieves the specified display profile node object from the specified display profile document. If no display profile node object is specified, the entire display profile document is retrieved. The display profile object is displayed in its native XML format on standard out.

The list subcommand takes these options:

- Administrator's distinguished name and password for accessing the LDAP database. These options are required.
- Name of the display profile node object to display.
- Display profile node object to display defined by the `-g` or `--global` option for the global level node, or `-d` or `--dn` option with a specific non-global level node specified. The `-g` or `-d` option is required. In the absence of the command-specific `-n` or `--name` option, it displays the entire display profile document. The `-g` or `--global` option displays the entire root display profile document.

Syntax

`dpadmin list -u|--runasdn` *uid* `-w|--password` *password* `{(-g|--global)|(-d|--dn` *dn*`)} [-n|--name` *name*`]`

`dpadmin list -h|--help`

Options

The following table contains two columns: the first column lists the possible options, arguments or operands for the `list` subcommand; the second column gives a brief description. The following options are supported:

`-b or --verbose`	Specify this option to produce debugging messages.
`-d or --dn`	Specifies the distinguished name in the LDAP node to access the display profile document. The `-d` or `-g` option is required.
`-g or --global`	Specifies the global level node in LDAP to access the display profile document. The `-d` or `-g` option is required.
`-h or --help`	Specify this option to dpadmin to print out a brief help page to standard output. If no subcommand is present, a generic help page for dpadmin is printed. If one of the dpadmin subcommands is present, then a brief help page that is specific to the subcommand is printed.
`-l or --locale`	Use this option to have all debug / error messages localized in the specified locale. If not specified, it defaults to system locale.
`-n or --name`	Specifies the fully qualified name of the display profile container, channel, or provider object to display. This option is not required.
`-r or --dryrun`	Reports error or success of subcommand to sysout. Does not put the resulting changes of the subcommand in LDAP.
`-u or --runasdn`	Specifies the distinguished name of the user to use to bind to the Directory Server. This option is required.
`-w or --password`	Specifies the password of the distinguished name of the user used to bind to the Directory Server. This option is required.

Examples

Example 1 This example gets the named TemplateTableContainer from the `dc=org,dc=com` organization node and prints it to standard out.

```
dpadmin list -n TemplateTableContainer -u
"uid=amAdmin,ou=people,dc=org,dc=com" -w joshua -d "dc=org,dc=com"
```

Example 2 This example goes to the global level only to get mailcheck and if found, prints it to standard out.

```
dpadmin list -n mailcheck -u "uid=amAdmin,ou=people,dc=org,dc=com" -w
joshua -g
```

Example 3 This example gets the channel named Bookmark2 located in the container TemplateTableContainer and prints it to standard out.

```
dpadmin list -n TemplateTableContainer/Bookmark2 -u
"uid=amAdmin,ou=people,dc=org,dc=com" -w joshua -d "dc=org,dc=com"
```

merge

Description

This subcommand retrieves and displays the merged result of the specified DP node objects. Objects are displayed in their native XML format. The object to be displayed is sent to standard out. If you do not use the `-n` or `--name` option, then an error is reported.

The `merge` command accepts the `name` argument to specify the fully-qualified name of the display profile container, channel, or provider object to display. If the `name` argument is absent, then the entire DP document is displayed. If the `name` argument does not identify a display profile node object, then an error is reported.

NOTE The `merge` subcommand merely displays the merged view of the object and does not persist the result. The underlying data does not get affected by running this subcommand.

Syntax

```
dpadmin merge [-n|--name name] -u|--runasdn uid -w|--password password
{(-g|--global)|(-d|--dn dn)}
```

Options

The following table contains two columns: the first column lists the possible options, arguments or operands for the `list` subcommand; the second column gives a brief description. The following options are supported:

`-b or --verbose`	Specify this option to produce debugging messages.

`-h or --help`	Specify this option to dpadmin to print out a brief help page to standard output. If no subcommand is present, a generic help page for dpadmin is printed. If one of the dpadmin subcommands is present, then a brief help page that is specific to the subcommand is printed.
`-l or --locale`	Use this option to have all debug / error messages localized in the specified locale. If not specified, it defaults to system locale.
`-n or --name`	Specifies the fully qualified name of the display profile container, channel, or provider object to display or remove. This option is not required.

Example

The following is the merged result of the Bookmark channel for the user hradmin who is assigned to the HR Role.

Code Example 12-1 Example of the merge Subcommand

```
$ dpadmin list -n "Bookmark"  -u "uid=amAdmin,ou=People,dc=iplanet,dc=com" -w joshua -d
"dc=iplanet,dc=com"
<Channel name="Bookmark" provider="BookmarkProvider">
    <Properties merge="fuse" lock="false" name="_properties">
        <String name="title" value="My Bookmarks" merge="replace" lock="false"/>
        <String name="refreshTime" value="600" merge="replace" lock="false"/>
        <Collection name="targets" merge="fuse" lock="false">
            <String value="Sun home page|http://www.sun.com" merge="replace" lock="false"/>
            <String value="Everything you want to know about Sun Java
System...|http://www.sun.com/software/products/portal_srvr/home_portal.htm l"
merge="replace" lock="false"/>
            <String value="Sun Java System home page|http://www.sun.com/software"
advanced="false" merge="replace" lock="false"/>
        </Collection>
    </Properties>
</Channel>

$ dpadmin list -n "Bookmark"  -u "uid=amAdmin,ou=People,dc=iplanet,dc=com" -w joshua -d
"cn=HR Role,dc=iplanet,dc=com"
<Channel name="Bookmark" provider="BookmarkProvider">
    <Properties merge="fuse" lock="false" name="_properties">
        <String name="title" value="HR Admin Bookmarks" merge="replace" lock="false"/>
        <Collection name="targets" merge="fuse" lock="false">
            <String value="HR Admin home page|http://hr.acme.com" merge="replace"
lock="false"/>
```

Code Example 12-1 Example of the merge Subcommand *(Continued)*

```
        </Collection>
    </Properties>
</Channel>

$ dpadmin merge -n "Bookmark" -u "uid=amAdmin,ou=People,dc=iplanet,dc=com" -w joshua -d
"uid=hradmin,ou=people,dc=iplanet,dc=com"
<Channel name="Bookmark" provider="BookmarkProvider">
    <Properties merge="fuse" lock="false" name="_properties">
        <String name="title" value="HR Admin Bookmarks" merge="replace" lock="false"/>
        <Collection name="targets" merge="fuse" lock="false">
            <String value="Sun home page|http://www.sun.com" merge="replace" lock="false"/>
            <String value="Everything you want to know about Sun Java System
...|http://www.sun.com/software/products/portal_srvr/home_portal.html" merge="replace"
lock="false"/>
            <String value="Sun Java System home page|http://www.sun.com/software"
advanced="false" merge="replace" lock="false"/>
            <String value="HR Admin home page|http://hr.acme.com" merge="replace"
lock="false"/>
        </Collection>
        <Collection name="GlobalThemes" merge="fuse" lock="false">
            <Collection name="theme1" merge="fuse" lock="false">
                <String name="description" value="Sun Java System" merge="replace"
lock="false"/>
                ...
            </Collection>
        </Collection>
        <Collection name="locales" merge="fuse" lock="false" propagate="true"
advanced="false">
            <String name="en_US" value="English (United States)" merge="replace"
lock="false"/>
        </Collection>
        <String name="docroot" value="/docs/" merge="replace" lock="false"/>
        <String name="helpURL" value="desktop/usedesk.htm" merge="replace" lock="false"/>
    </Properties>
</Channel>
```

The output from the merge subcommand is comprised of an aggregated result, meaning that all DP objects that are available will be listed. For instance, properties such as GlobalThemes and locales are not specifically defined in Bookmark definition yet they show up in the output because these were merged in from one or more parent of the Bookmark channel.

modify

Description

This subcommand changes the value of an existing display profile object. The data that is supplied to the `dpadmin modify` command is either from one or more input files or from standard input (an XML fragment that follows the command).

This XML data always requires a proper XML header as well as a name that uniquely defines which display profile object is to be modified. An example of proper XML header is:

```
<?xml version="1.0" encoding="utf-8" standalone="no"?>
```

```
<!DOCTYPE DisplayProfile SYSTEM "jar://resources/psdp.dtd">
```

Subcommand Semantics

The semantics of the `modify` subcommand vary depending on the type of the display profile object being modified. When the `combine` option is specified, the new elements (like properties) in the display profile object are combined with the existing ones rather than replacing them. The variations of the `modify` subcommand are discussed below.

Display Profile An entire display profile document can be changed to the new object value specified by using a file. When the `combine` option is specified, every display profile object in the display profile document is recursively combined. See the information below for how the `combine` option works for each display profile object.

Channel or Container A channel or container can be changed to the new object value. When modifying a channel or container, if the `parent` option is:

- Specified, the specified parent container is searched for a channel or container that matches the name of the new display profile object. If it is found, it is replaced by the new display profile object.
- Absent, the root display profile object is assumed to be the parent container. So the root is searched for a channel or container that matches the name of the new display profile object. If it is found, it is replaced with the new display profile object.

When the `combine` option is specified, the existing Properties, Available, and Selected objects are combined with the new display profile objects.

Properties A display profile object's properties can be changed to the new value. The `parent` option is required to modify a display profile object's properties. A display profile node (either channel or container) or display profile provider object that matches the specified name is searched for under the specified parent. If found, the object's properties object is replaced by the new display profile object. When the `combine` option is specified, the existing properties are combined with the new display profile object.

Available or Selected The Available or Selected list in a container can be replaced with the new display profile object. The `parent` option is required to modify display profile objects of this type. A display profile container that matches the parent name is searched for. Then the Selected or Available object is replaced by the new display profile object. When the `combine` option is specified, the existing Selected or Available object is combined with the new display profile object.

String, Boolean, Integer, Collection, or Locale A String, Boolean, Integer, Collection, or Locale property in a display profile object can be replaced by new display profile object property.

If the `parent` option is specified, the display profile node (either a channel or container) or display profile provider (in that order) that matches the specified name is searched for. If found, a property that matches the name of the new property is searched for. If found, the property in the display profile object is replaced by new display profile object property.

If the `parent` option is absent, then the display profile root node is used and the property is replaced at the root node.

When the `combine` option is specified, the existing Collection or Locale object is combined with the new display profile object. The `combine` option is not supported for atomic display profile properties such as String, Boolean, and Integer.

The atomic display profile properties such as String, Boolean, and Integer need not be named. If unnamed, the name is assumed to be equal to the string representation of the value. For example, the following two display profile integer objects are equal:

```
<Integer value="3"/>
```

```
<Integer name="3" value="4"/>
```

Provider An existing display profile Provider object can be replaced with the display profile provider of the same name. A display profile provider object that matches the name of the new display profile provider object is searched for under the root display profile node. If present, the new display profile provider object is inserted under the root display profile object, replacing the existing display profile provider of the same name. Since providers can only exist under the root node (the root node is an implicit container), the `parent` option must not be specified.

Subcommand Options

The `modify` subcommand takes these options:

- Administrator's distinguished name and password for accessing the LDAP database by using the `-u` or `--runasdn` and `-w` or `--password` options respectively. These options are required.
- Display profile node object to modify defined by the `-g` or `--global` option for the global level node, or `-d` or `--dn` option with a specific non-global level node specified. The `-g` or `-d` option is required.
- Name of the file where the XML input is included. This argument is optional. When not used, XML input is expected from standard input.
- Fully qualified name of the parent of the display profile object to be modified.
- The option to perform a merge of display profile objects.

Syntax

`dpadmin modify -u|--runasdn` *uid* `-w|--password` *password* `{(-g|--global)|(-d|--dn` *dn*`)} [-p|--parent` *parent*`] [-m|--combine]` *file*`|EOF`

`dpadmin modify -h|--help`

Options

The following table contains two columns: the first column lists the possible options, arguments or operands for the `modify` subcommand; the second column gives a brief description. The following options are supported:

`-b or --verbose`	Specify this option to produce debugging messages.
`-d or --dn`	Specifies the distinguished name in the LDAP node to access the display profile document. The `-d` or `-g` option is required.

file	If present, the file argument must be the last argument on the command line. It specifies a path to an XML file that contains an XML fragment that conforms to the display profile DTD. If the file argument is absent from the modify subcommand, then input must be redirected into dpadmin from standard input.
`-g or --global`	Specifies the global level node in LDAP to access the display profile document. The `-d` or `-g` option is required.
`-h or --help`	Specify this option to dpadmin to print out a brief help page to standard output. If no subcommand is present, a generic help page for dpadmin is printed. If one of the dpadmin subcommands is present, then a brief help page that is specific to the subcommand is printed.
`-l or --locale`	Use this option to have all debug / error messages localized in the specified locale. If not specified, it defaults to system locale.
`-m or --combine`	Combines the specified display profile objects with the new display profile objects. The combine option can only be used with these display profile objects: Display Profile root, Channel, Container, Properties, Available, Selected, Collection, and Locale. This is an optional option.
`-p or --parent`	Specifies the fully qualified name of the parent of the display profile object to be modified. This is an optional option.
`-r or --dryrun`	Reports error or success of subcommand to `sysout`. Does not put the resulting changes of the subcommand in LDAP.
`-u or --runasdn`	Specifies the distinguished name of the user to use to bind to the Directory Server. This option is required.
`-w or --password`	Specifies the password of the distinguished name of the user used to bind to the Directory Server. This option is required.

Examples

Example 1 In the following example, modify (replace) the channel named NewNews in the container TemplateTableContainer with value specified as XML text on standard input.

```
$ dpadmin modify -p TemplateTableContainer -u
"uid=amAdmin,ou=people,dc=org,dc=com" -w joshua -d "dc=org,dc=com" <<EOF
<?xml version="1.0" encoding="utf-8" standalone="no"?>
<!DOCTYPE DisplayProfile SYSTEM "jar://resources/psdp.dtd">
<Channel name="NewNews" provider="newsprovider">
    <Properties>
        <String name="title" value="News Channel"/>
        <String name="description" value="This channel is all about news"/>
    </Properties>
</Channel>
EOF
```

Example 2 In this example, in the channel NewNews, replace the property named in the file `farble.xml` with the new object in `farble.xml` file.

```
dpadmin modify -p TemplateTableContainer/NewNews -u
"uid=amAdmin,ou=people,dc=org,dc=com" -w joshua -d "dc=org,dc=com"
farble.xml
```

The `farble.xml` file contains the following:

```
<?xml version="1.0" encoding="utf-8" standalone="no"?>
<!DOCTYPE DisplayProfile SYSTEM "jar://resources/psdp.dtd">
<String name="welcome" value="Hi, welcome to farble land!!!!"/>
```

Example 3 In the following example, using the combine option, a new property named “msg2” is added to the collection named “bar”. Note that the existing property, “msg” still remains in the result.

```
dpadmin list -n TemplateTableContainer -u
"uid=amAdmin,ou=people,dc=org,dc=comp" -w joshua -d "dc=org,dc=com"
```

```
...
<Collection name="news">
<Collection name="bar">
```

```
<String name="msg" value="hi"/>
</Collection>
</Collection>
...
```

```
$ dpadmin modify -p TemplateTableContainer -u "uid=amAdmin,dc=org,dc=com" -w
joshua -d "dc=org,dc=com" -m <<EOF
<?xml version="1.0" encoding="utf-8" standalone="no"?>
<!DOCTYPE DisplayProfile SYSTEM "jar://resources/psdp.dtd">
<Collection name="news">
<Collection name="bar">
<String name="msg2" value="woo hoo"/>
</Collection>
</Collection>
EOF
```

```
dpadmin list -n TemplateTableContainer -u "uid=amAdmin,dc=org,dc=com" -w
joshua -d "dc=org,dc=com"
```

```
...
<Collection name="news">
<Collection name="bar">
<String name="msg" value="hi"/>
<String name="msg2" value="woo hoo"/>
</Collection>
</Collection>
...
```

Example 4 In the following example, the value of “title” and “msg1” are replaced with the new values. Available and Selected have both had a Reference value added. The “news” collection has added “msg3”. This example shows that the -m or -combine option with the modify subcommand can be used to combine and replace as necessary.

```
dpadmin list -n test -u "uid=amAdmin,ou=people,dc=org,dc=com" -w joshua -d
"dc=org,dc=com"
```

```
<Container name="test" provider="testprovider">
<Properties>
<String name="title" value="test"/>
</Properties>
<Available/>
<Selected/>
<Channels>
<Channel name="test1" provider="test1provider">
<Properties>
<Collection name="news">
<String name="msg1" value="blah"/>
<Collection name="bar">
<String name="msg2" value="hi"/>
</Collection>
</Collection>
</Properties>
</Channel>
</Channels>
</Container>
```

```
$ dpadmin modify -u "uid=amAdmin,ou=people,dc=org,dc=com" -w joshua -d
"dc=org,dc=com" -m <<EOF
<?xml version="1.0" encoding="utf-8" standalone="no"?>
<!DOCTYPE DisplayProfile SYSTEM "jar://resources/psdp.dtd">
<Container name="test" provider="testprovider">
<Properties>
<String name="title" value="Test Container"/>
</Properties>
<Available>
<Reference value="test1"/>
</Available>
<Selected>
<Reference value="test1"/>
</Selected>
<Channels>
<Channel name="test1" provider="test1provider">
<Properties>
<Collection name="news">
<String name="msg1" value="123"/>
<Collection name="bar">
<String name="msg3" value="123/>
</Collection>
</Collection>
</Properties>
</Channel>
</Channels>
</Container>
EOF
```

```
dpadmin list -n test -u "uid=amAdmin,ou=people,dc=org,dc=com" -w joshua -d
"dc=org,dc=com"
```

```
<Container name="test" provider="testprovider">
<Properties>
<String name="title" value="Test Container"/>
</Properties>
<Available>
<Reference value="test1"/>
</Available>
<Selected>
<Reference value="test1"/>
</Selected>
<Channel name="test1" provider="test1provider">
<Properties>
<Collection name="news">
<String name="msg1" value="123"/>
<Collection name="bar">
<String name="msg2" value="hi"/>
<String name="msg3" value="123/>
</Collection>
</Collection>
</Properties>
</Channel>
</Channels>
</Container>
```

Example 5 In the following example, the combine option is used to add conditional properties.

```
dpadmin list -n test -u "uid=amAdmin,ou=People,dc=iplanet,dc=com" -w joshua
-d "dc=iplanet,dc=com"
```

```
<Channel name="test" provider="testprovider">
<Properties>
<Collection name="foo">
<String name="foo1" value="bar"/>
</Collection>
</Properties>
</Channel>
```

```
$ dpadmin modify -p test -u "uid=amAdmin,ou=People,dc=iplanet,dc=com" -w
joshua -d "dc=iplanet,dc=com" -m <<EOF
<?xml version="1.0" encoding="utf-8" standalone="no"?>
<!DOCTYPE DisplayProfile SYSTEM "jar://resources/psdp.dtd">
<ConditionalProperties condition="client" value="nokia">
<Collection name="foo">
<String name="foo1" value="nokia bar"/>
</Collection>
</ConditionalProperties>
EOF
```

```
dpadmin list -n test -u "uid=amAdmin,ou=People,dc=iplanet,dc=com" -w joshua
-d "dc=iplanet,dc=com"
```

```
<Channel name="test" provider="testprovider">
<Properties>
<Collection name="foo">
<String name="foo1" value="bar"/>
</Collection>
<ConditionalProperties condition="client" value="nokia">
<Collection name="foo">
<String name="foo1" value="nokia bar"/>
</Collection>
</ConditionalProperties>
</Properties>
</Channel>
```

```
$ dpadmin modify -p test -u "uid=amAdmin,ou=People,dc=iplanet,dc=com" -w
joshua -d "dc=iplanet,dc=com" -m <<EOF
<?xml version="1.0" encoding="utf-8" standalone="no"?>
<!DOCTYPE DisplayProfile SYSTEM "jar://resources/psdp.dtd">
<ConditionalProperties condition="client" value="nokia">
<ConditionalProperties condition="locale" value="en">
<String name="abc" value="nokia en abc"/>
</ConditionalProperties>
</ConditionalProperties>
EOF
```

```
dpadmin list -n test -u "uid=amAdmin,ou=People,dc=iplanet,dc=com" -w joshua
-d "dc=iplanet,dc=com"
```

```
<Channel name="test" provider="testprovider">
<Properties>
<Collection name="foo">
<String name="foo1" value="bar"/>
</Collection>
<ConditionalProperties condition="client" value="nokia">
<Collection name="foo">
<String name="foo1" value="nokia bar"/>
</Collection>
<ConditionalProperties condition="locale" value="en">
<String name="abc" value="nokia en abc"/>
</ConditionalProperties>
</ConditionalProperties>
</Properties>
</Channel>
```

add

Description

This subcommand adds a new display profile object to the display profile. This subcommand requires that the object to be added does not exist in the display profile. The `add` subcommand reads data for the new object from standard input or from one or more files specified as an argument to the command. Data for the new object must be XML and conform the Sun Java System Portal Server display profile DTD.

This XML data always requires a proper XML header as well as a name that uniquely defines which display profile object is to be modified. An example of proper XML header is:

```
<?xml version="1.0" encoding="utf-8" standalone="no"?>
```

```
<!DOCTYPE DisplayProfile SYSTEM "jar://resources/psdp.dtd">
```

Subcommand Semantics

The semantics of the add subcommand vary depending on the type of the display profile object being added. That is:

Display profile An entire display profile document can be added to the specified LDAP node. If the document already exists at the node, then an error is reported. The `parent` option must not be specified when adding a new display profile document.

Channel or container A channel or container can be added. If the `parent` option is present, the parent display profile object is located for the given name and under that parent container, the specified Channel or Container is added. If the `parent` option is absent, the parent display profile object is assumed to be the root display profile object; so under root the specified Channel or Container, the object is added.

Properties Because a properties bundle is required for all display profile nodes and display profile provider objects, they already exist and cannot be added. Use the `modify` subcommand.

Available or selected Because Available and selected objects are required for a display profile container, they already exist and cannot be added. Use the `modify` subcommand.

String, Boolean, Integer, Collection, or Locale The display profile object String, Boolean, Integer, Collection, or Locale properties can be added. The `parent` option must be specified to add display profile object properties of this type. Under the specified parent, a display profile node (either a channel or container) or display profile provider is searched for (in that order) that matches the name. If found, the given display profile property is added to the display profile node or display profile provider.

The atomic display profile properties such as String, Boolean, and Integer need not be named. If unnamed, the name is assumed to be equal to the string representation of the value.

Provider A display profile provider is inserted under the root node. Because providers can only exist under the root node, do not use the `parent` option. If an object of the same name already exists, then an error is reported.

Subcommand Options

The add subcommand takes these options:

- Administrator's distinguished name and password for accessing the LDAP database. These options are required.
- Display profile document to add or the display profile document where the object must be added (the `-d` or `--dn` option). The display profile object to add defined by the `-g` or `--global` option for the global level node. The `-g` or `-d` option is required.
- Name of the file where the XML input is included (the file argument).
- Fully qualified name of the parent where the display profile node object is to be added to (the `-p` or `--parent` option).

Syntax

```
dpadmin add -u|--runasdn uid -w|--password password {(-g|--global)|(-d|--dn
dn)} [-p|--parent parent] file|<<EOF

dpadmin add -h|--help
```

Options

The following table contains two columns: the first column lists the possible options, arguments or operands for the add subcommand; the second column gives a brief description. The following options are supported:

`-b or --verbose`	Specify this option to produce debugging messages.
`-d or --dn`	Specifies the distinguished name in the LDAP node to access the display profile document. The `-d` or `-g` option is required.
`file`	If present, the file argument must also be the last argument on the command line. It specifies a path to an XML file that contains an XML fragment that conforms to the display profile DTD. If the file argument is absent for the add subcommand, then input must be redirected into dpadmin from standard input.
`-g or --global`	Specifies the global level node in LDAP to access the display profile document. The `-d` or `-g` option is required.
`-h or --help`	Specify this option to dpadmin to print out a brief help page to standard output. If no subcommand is present, a generic help page for dpadmin is printed. If one of the dpadmin subcommands is present, then a brief help page that is specific to the subcommand is printed.
`-l or --locale`	Use this option to have all debug / error messages localized in the specified locale. If not specified, it defaults to system locale.
`-p or --parent`	Specifies the fully qualified name of the parent of the display profile object to add.

`-r or --dryrun`	Reports error or success of subcommand to `sysout`. Does not put the resulting changes of the subcommand in LDAP.
`-u or --runasdn`	Specifies the distinguished name of the user to use to bind to the Directory Server. This option is required.
`-w or --password`	Specifies the password of the distinguished name of the user used to bind to the Directory Server. This option is required.

Example

In this example, the command adds the collection property named “zipCodes” specified on standard input to the channel named Postal in the container named SampleTabPanelContainer.

```
$ dpadmin add -p SampleTabPanelContainer/Postal -u
"uid=amAdmin,ou=people,o=sesta.com,o=isp" -w joshua -d "o=sesta.com,o=isp"
<<EOF
<?xml version="1.0" encoding="utf-8" standalone="no"?>
<!DOCTYPE DisplayProfile SYSTEM "jar://resources/psdp.dtd">
<Collection name="zipCodes">
<Integer value="98012"/>
<Integer value="98036"/>
<Integer value="94025"/>
<Integer value="95112"/>
</Collection>
EOF
```

remove

Description

This subcommand removes an existing display profile object from the display profile. If the object to be removed does not exist in the specified display profile document, an error is reported. This subcommand takes `type`, `parent`, and `name` options.

The type option specifies the type of display profile object to remove. The parent option specifies the fully qualified name of the parent display profile object to remove the display profile object from. The parent display profile object type varies depending on the type of display profile object being removed. The name option specifies the name of the object to remove.

Subcommand Semantics

The semantics of the parent and name options vary depending on the type of display profile object being removed. The following table contains two columns: the first column lists the possible values for the type option; the second column gives a brief description of exactly what is removed.

root	Removes the entire display profile document from the LDAP node as specified by the distinguishedname option or global level display profile if -g | --global option is supplied. The name option is not needed when type=root.
channel	The name option is required. If the parent option is absent, then the parent container is assumed to be the root display profile node. Otherwise, the parent option is assumed to be the parent container name of the channel to remove. The name option specifies the name of the channel or container to remove.
provider	The parent option must not be specified since providers reside under the root display profile node. The name option is required and specifies the provider to remove.
property	The parent option specifies the fully-qualified name of the parent container, channel, or provider object to remove the property from. If the parent option is absent, then the root display profile node is assumed to be the parent object. The name option specifies the name of the property to remove. If the name option is absent, an error is reported. For unnamed display profile properties, the name is equal to the string representation of the value.

`available or selected`	Both parent and name options are required. The parent option is assumed to name the parent container or channel object to remove the available (selected) reference from. The name option gives the value of the reference to be removed. If the name option is absent, then an error is reported.

Subcommand Options

The `remove` subcommand takes these options:

- Administrator's distinguished name and password for accessing the LDAP database. These options are required.
- Name of the display profile node object to remove. This option is required except when `type=root`.
- Display profile document node in the LDAP database where the display profile document containing the object to be removed exists. Either the `-d` (`--dn`) or `-g` (`--global`) option is required.
- Type of display profile node object to remove. This option is required.
- Fully qualified name of the parent of the display profile node object to remove.

Syntax

`dpadmin remove -u|--runasdn` *uid* `-w|--password` *password* `{(-g|--global)|(-d|--dn` *dn*`)} [-n|--name` *name*`] [-p|--parent` *parent*`] -t|--type` *type*

`dpadmin remove -h|--help`

Options

The following table contains two columns: the first column lists the possible options, arguments or operands for the `remove` subcommand; the second column gives a brief description. The following options are supported:

`-b or --verbose`	Specify this option to produce debugging messages.
`-d or --dn`	Specifies the distinguished name in the LDAP node to access the display profile document. The `-d` or `-g` option is required.

`-g or --global`	Specifies the global level node in LDAP to access the display profile document. The `-d` or `-g` option is required.
`-h or --help`	Specify this option to dpadmin to print out a brief help page to standard output. If no subcommand is present, a generic help page for dpadmin is printed. If one of the dpadmin subcommands is present, then a brief help page that is specific to the subcommand is printed.
`-l or --locale`	Use this option to have all debug / error messages localized in the specified locale. If not specified, it defaults to system locale.
`-n or --name`	Specifies the display profile container, channel, or provider object to remove. This option is required except when `type=root`.
`-p or --parent`	Specifies the fully qualified name of the parent of the display profile object to remove.
`-r or --dryrun`	Reports error or success of subcommand to sysout. Does not put the resulting changes of the subcommand in LDAP.
`-t or --type`	Specifies the type of display profile object being removed. This option is required.
`-u or --runasdn`	Specifies the distinguished name of the user to use to bind to the Directory Server. This option is required.
`-w or --password`	Specifies the password of the distinguished name of the user used to bind to the Directory Server. This option is required.

Examples

Example 1 In this example, the command removes the property named locations from the channel or container named Bookmarks.

```
dpadmin remove -t property -p Bookmarks -n locations -u
"uid=amAdmin,ou=people,o=sesta.com,o=isp" -w joshua -d "o=sesta.com,o=isp"
```

Example 2 Remove the provider pctest from the global display profile

```
dpadmin remove -t provider -n "pctest" -u
"uid=amAdmin,ou=people,o=sesta.com,o=isp" -w joshua -g
```

Example 3 In this example, the command removes the channel named Test that is in the parent container named TemplateTableContainer.

```
dpadmin remove --type channel --parent TemplateTableContainer --name "Test"
--runasdn "uid=amAdmin,ou=people,o=sesta.com,o=isp" --password joshua --dn
"o=sesta.com,o=isp"
```

Example 4 In the following example, execute a sequence of remove subcommand to remove channel z.

```
dpadmin list -n X -u "uid=amAdmin,ou=people,o=sesta.com,o=isp" -w joshua -d
"o=sesta.com,o=isp"
```

```
<Container name="X" ...>
<Channels>
<Container name="Y" ...>
<Channels>
<Channel name="z" .../>
</Channels>
</Container>
</Channels>
</Container>
```

```
$ dpadmin remove -t channel -p X -n Y/z -u
"uid=amAdmin,ou=people,o=sesta.com,o=isp" -w joshua -d "o=sesta.com,o=isp"
```

```
$ dpadmin remove -t channel -p X/Y -n z -u
"uid=amAdmin,ou=people,o=sesta.com,o=isp" -w joshua -d "o=sesta.com,o=isp"
```

```
$ dpadmin remove -t channel -n X/Y/z -u
"uid=amAdmin,ou=people,o=sesta.com,o=isp" -w joshua -d "o=sesta.com,o=isp"
```

batch

Description

The `batch` subcommand enables the processing of multiple display profile commands in an optimized fashion. The subcommands are listed in a batch script file (required) and executed consecutively. If an error occurs, the default is to report the error and exit. The `-c` or `--continue` option denotes continuous processing mode. In this mode, if an error occurs, it is reported and the `dpadmin` command continues with the next subcommand.

Batch Script

The command batch scripts must be text (ASCII) documents and can contain any number of subcommands for input into dpadmin, except a batch subcommand. A subcommand must be entered on a single line (the new line character indicates the end of the command). For each subcommand, the administrator's distinguished name and password must be specified in the command line. The syntax for the subcommand is exactly the same as if the subcommand was entered directly into a shell (without the dpadmin part). The script cannot contain XML, so subcommands that require XML input must have it in a file. If the distinguished name (or DN) contains space(s), use double-quotes around it.

In the example batch script file below, note that each command is on a single line.

```
add -p PostalMailer -u uid=amAdmin,ou=People,dc=iplanet,dc=com -w joshua -d
dc=iplanet,dc=com zipCodes.xml

add -p PostalStamps -u uid=amAdmin,ou=People,dc=iplanet,dc=com -w joshua -d
dc=iplanet,dc=com zipCodes.xml stampRates.xml

add -p PostalRates -d "cn=hr role,dc=iplanet,dc=com" zipCodes.xml
stampRates.xml
```

Subcommand Options

The `batch` subcommand takes the following options:

- Administrator's distinguished name and password for accessing the LDAP database. These options are required.
- Option to denote continuous processing mode.
- Name of the batch script file.

Syntax

`dpadmin batch [-c|--continue] -u|--runasdn` *uid* `-w|--password` *password* `-f|--file` *batch-script-file*

`dpadmin batch -h|--help`

Options

The following table contains two columns: the first column lists the possible options, arguments or operands for the `batch` subcommand; the second column gives a brief description. The following options are supported:

`-b or --verbose`	Specify this option to produce debugging messages.
`-c or --continue`	Indicates a continuous operation mode. Errors are reported, but dpadmin continues with the next subcommand if this option is specified. By default, dpadmin exits after reporting an error.
`-f or --file`	Specifies the batch script file. This argument is required.
`-h or --help`	Specify this option to `dpadmin` to print out a brief help page to standard output. If no subcommand is present, a generic help page for dpadmin is printed. If one of the dpadmin subcommands is present, then a brief help page that is specific to the subcommand is printed.
`-l or --locale`	Use this option to have all debug / error messages localized in the specified locale. If not specified, it defaults to system locale.
`-u or --runasdn`	Specifies the distinguished name of the user to use to bind to the Directory Server. This is used with the list, modify, add, and remove subcommands only. This option is optional. If specified, the distinguished name is used to authenticate throughout the batch process. In addition, each subcommand in the batch script can also have its own authentication which will override this distinguished name.

`-w or --password`	Specifies the password of the distinguished name of the user used to bind to the Directory Server. This option is optional. If specified, the distinguished name is used to authenticate throughout the batch process. In addition, each subcommand in the batch script can also have its own authentication which will override this password.

Options

The following table contains a summary of the `dpadmin` command options. The table contains two columns: the first column lists the possible options, arguments or operands; the second column gives a brief description. The following options are supported:

`-V or --version`	Specify this option to `dpadmin` to print descriptive information about the utility, such as its version, legal notices, and other similar information to standard output. Any subcommand and all other options are ignored when this option is present.

Guidelines

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,

  PortalServer-base`/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,

  PortalServer-base`/bin/dpadmin list -u` *uid* `-w` *password* `-d` *distinguishedname* `> dp-org.xml`

  Edit the `dp-org.xml` file.

  PortalServer-base`/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.

Chapter 13

par

This chapter contains the following sections:

Description

The `par` command performs functions involving the specified `.par` file. It can be used for exporting and importing channels or providers to and from the Sun Java System Portal Server.

Syntax

The `par` command syntax is described in this section. Mixing long-named options with short ones in one command line is not recommended.

Short-Named Format

`par containers -r` *uid* `-p` *password* `[-d]` *dn*`|global`

`par describe [-d]` *parfile*

`par export -r` *uid* `-p` *password* `[-m] [-d] -s` *staticdir* `[-v]` *parfile* *dn*`|global {`*exportfile*`|from=}...`

`par import -r` *uid* `-p` *password* `[-o] [-d] -s` *staticdir* `[-v]` *parfile* `[`*dn*`|global [op...]]`

`par import -r` *uid* `-p` *password* `-a [-d] -s` *staticdir* `[-v]` *parfile* `[`*dn*`|global]`

Long-Named Format

`par containers --runasdn` *uid* `--password` *password* `[--debug]` *dn*`|global`

`par describe [--debug]` *parfile*

`par export --runasdn` *uid* `--password` *password* `[--modify] [--debug] --staticdir` *staticdir* `[--verbose]` *parfile* *dn*`|global {`*exportfile*`|from=}...`

`par import --runasdn` *uid* `--password` *password* `[--overwrite] [--debug] --staticdir` *staticdir* `[--verbose]` *parfile* `[`*dn*`|global [op...]]`

`par import --runasdn` *uid* `--password` *password* `--auto [--debug] --staticdir` *staticdir* `[--verbose]` *parfile* `[`*dn*`|global]`

Subcommands

The following subcommands are supported:

- containers
- describe
- export
- import

containers

Description

Lists all available containers and channels in a particular display profile document, indicated by the specified directory server name (or global). This can be used as an aid to formulating other commands.

Syntax

```
par containers -r|--runasdn uid -p|--password password [-d|--debug]
[-v|--verbose] dn|global
```

Example

```
par containers -r "uid=amAdmin,ou=people,o=sesta.com,o=isp" -p joshua -d
"o=sesta.com,o=isp"
```

In this example, the command lists all available containers in the display profile document residing in the LDAP node "o=sesta.com,o=isp".

describe

Description

Describes the contents of the specified .par file, including the entries, and any built-in autoextract operations defined for the entries.

Syntax

```
par describe parfile
```

Example

```
par describe myfile.par
```

In this example, the command output or description of the myfile.par may be something like the following:

```
Class Root: /

Property Based File Root: /pbfiles

Display Profile Root: /dp
```

```
Static Content Root: /static

Entry: mychannel

AutoExtract:dpnode=o%3Dsesta.com%2Co%3Disp,channel,entry=mychannel

DP Document: this my JSP based channel.

Channel: SampleJSP.a

Includes: Property Based File, root templateBaseDir, path
default/mychannel/samplecontent.jsp (channel)

Includes: Property Based File, root templateBaseDir, path
default/mychannel/sampledoedit.jsp (channel)

Includes: Property Based File, root templateBaseDir, path
default/mychannel/sampleedit.jsp (channel)

Includes: Property Based File, root templateBaseDir, path
default_en_US/mychannel/samplecontent.jsp

(channel)

Includes: Property Based File, root templateBaseDir, path
default_en_US/mychannel/sampledoedit.jsp

(channel)

Includes: Property Based File, root templateBaseDir, path
default_en_US/mychannel/sampleedit.jsp

(channel)
```

export

Description

Populates the specified `.par` file by exporting the provider or channel information from the portal server. The command takes a `.par` file, a directory server name argument (or keyword global) corresponding to the desired display profile document to update, and any number of (requires at least one) exportfile or `from` specifications. The `from` specifications contain exactly the same information as an export file; the only difference is that the “lines” are separated by semicolons.

The `par export` command without the `-m` option creates a `.par` file. The `par export` command with the `-m` option is used to update and / or add to an already existing `.par` file that defines a provider, channel or container.

Syntax

`par export -r|--runasdn` *uid* `-p|--password` *password* `[-d|--debug] -s|--staticdir` *staticdir* `[-v|--verbose]` *parfile dn*`|global {`*exportfile*`|from=}...`

`par export -r|--runasdn` *uid* `-p|--password` *password* `[-d|--debug] -s|--staticdir` *staticdir* `[-v|--verbose] -m|--modify` *parfile dn*`|global {`*exportfile*`|from=}...`

Example

```
par export -r "uid=amAdmin,ou=people,dc=sesta,dc=com" -p joshua
mychannel.par "o=sesta.com,o=isp" myexport.txt
```

Here `myexport.txt` contains:

```
from: channel mychannel
directory: templateBaseDir . mychannel
directory: templateBaseDir . mychannel
description: this is my JSP based channel
```

In this example, the command exports the channel definition and template files for mychannel into `mychannel.par` from the `"dc=sesta,dc=isp" dn`. Also, if it were a JSPProvider channel, the directory line transfers all of the `.jsp` files, including locale-specific versions.

import

Description

Imports objects from the specified `.par` file into the portal server. The command takes a `.par` file, and optional arguments for the display profile document to import the objects into the indicated display node in the directory server (or the root display profile indicated by the keyword global), and operations to be performed. If these things are not specified, they are taken from the `.par` file. The auto option can be used to indicate that you wish to simply perform the autoextract operations already contained in the `.par` file.

If you want to add a new channel, you can use the `par import` command with or without the `-o` option. If a channel already exists, you must use the `-o` option with the `par import` command to completely replace (overwrite) the old channel. You can use this subcommand to import providers as well as channels.

Syntax

```
par import -r|--runasdn uid -p|--password password [-o] [-d|--debug]
-s|--staticdir staticdir [-v|--verbose] parfile [dn|global [op...]]
```

```
par import -r|--runasdn uid -p|--password password -a|--auto [-d|--debug]
-s|--staticdir staticdir [-v|--verbose] parfile [dn|global]
```

Examples

Example 1 In this example, the command extracts the channel from myfile.par file, if that is the automatic operation defined in the `myfile.par` PAR file.

```
par import -r "uid=amAdmin,ou=people,o=sesta.com,o=isp" -p joshua --auto
myfile.par "o=sesta.com,o=isp"
```

Example 2 In this example, the command extracts the channel explicitly, installing it with a different name in the target `dn`, and making it available in container topcontainer.

```
par import -r "uid=amAdmin,ou=people,o=sesta.com,o=isp" -p joshua
myfile.par "o=sesta.com,o=isp"
"entry=mychannel,channel=anothername,avail=topcontainer"
```

Options

The following table contains two columns: the first column lists the possible options for the `par` command; the second column gives a brief description of the corresponding option. This command supports the following options (listed in alphabetical order):

`-a or --auto`	Use with the `import` subcommand to apply the autoextract operations from the `.par` file. There should be no operations specified on the command line in this case. The `dn` argument may still be specified; if specified, it replaces the `dn` in the autoextract operations. If operations are specified on the command line, they are ignored.
`-d or --debug`	Specify this to produce extra debugging information on error messages.

`-m or --modify`	Use with the `export` subcommand to update an existing `.par` file rather than replace it. Any new files added for an entry supplement or replace the old ones. Also, use this command to add new files to an existing provider or channel by using a `.par` file.
`-o or --overwrite`	Use with the `import` subcommand to replace existing channels.
`-p or --password`	Specifies the password for authentication. Required for all of the subcommands except `describe`. If not specified, the `par` utility prompts for it.
`-r or --runasdn`	Specifies the distinguished name of the user for authentication. Required for all of the subcommands except `describe`. If not provided, the `par` utility prompts for it. Use the format `uid=`*`userName`*`,ou=people,dc=`*`organizationName`*`,dc=`*`organizationalUnit`*
`-s or --staticdir`	Defines the host-specific directory of the static content directory to be used for import or export.
`-v or --verbose`	Describes operations as they are executed. Use with the `import` and `export` subcommands.
`-V or --version`	Specify this option to the `par` command to print descriptive information about the utility, such as its version, legal notices, and other similar information to standard output. Any subcommand and all other options are ignored when this option is present.
`-?`	Obtain help for any subcommand.

Arguments

The following table contains two columns: the first column lists the possible arguments for the `par` command; the second column gives a brief description of the corresponding argument. This command takes the following arguments:

`dn`	Specifies the distinguished node in the directory server to access. Use the format "`o=`*organizationName*`,o=`*organizationalUnit*" - SHOULD'NT THIS BE "dc=organizationName,dc=organizationalUnit"?
`global`	Specifies the global level node in LDAP to access the display profile document.
`exportfile`	These files each correspond to an entry (provider, channel, or provider/channel combination) in the `.par` file, and simply specify the data to be inserted into the specified `.par` file. It can be a small file if the information is too large to list on the command line. See Export Files for more information.
`from`	Specified on the command line, this is taken as equivalent to an export file containing the `from` line, followed by an equal to ("=") sign, and any other lines separated by a semicolon (";"). See Export Files for more information on line properties.
`op`	Specifies the operation to perform. See Operations for more detail.
`parfile`	Specifies the PAR file to operate upon; that is, indicates the PAR file to import, export, or describe.

Export Files

These files simply specify data to be inserted into a `.par` file. The file consists of lines containing a keyword, followed by a colon and white space delimited fields. The `from:` line is required and it must be the first line of the file. Lines beginning with "#" are treated as comments.

The following table contains two columns: the first column lists the possible line keywords or properties; the second column gives a brief description of the corresponding property.

`from:` *types name*	The `from` line indicates what entity is being exported. The `types` can be “channel”, “provider”, or “channel,provider”, and “channel+provider”. The name indicates the channel name, or a provider name if a provider is being exported. The name must be URL encoded if the name contains white space (+), commas (%2C), colons (%3A), semicolons (%3B), plus signs (%2B), or percent signs (%25).
`auto:` *none* `auto:` *op*	The `auto` line specifies the autoextract operation for the entry. It takes the `op` argument followed by the operation. The `none` argument can also be entered, suppressing the autoextract. If no `auto:` line is specified, a default autoextract is produced. The default operation is to extract the channel and / or provider with its original names.
`file:` *root*`|.` *path* [*types*]	The `file` line indicates that a file, based on a property setting, is to be included. The property can come from either the desktop properties file, located by default in `/etc/opt/SUNWps/desktop/desktopconfig.properties` file or from the display profile visible to a `getProperty()` call for the item being exported or imported. The `root` specifies the root of the file location and path specifies the path to the rest of the file. The `root` is a property name that corresponds to a directory (like). If `root` is given as `"."`, the file is assumed to be static content located at the web server’s doc root. You can also specify the types of operation the file is to be associated with, defaulting to “channel”. The types can be “channel”, “provider”, or “channel,provider”, and “channel+provider”.
`class:` *class* [*types*]	The `class` line indicates that a class file is to be packaged with the entry, and you may optionally specify the types of operations that the class file are associated with. If not specified, “provider” is assumed. types can be “channel”, “provider”, or “channel,provider”, and “channel+provider”; also, when specifying both, you can use a space.

`directory:` *root*`|.` *dir* `+|.|`*filter* `[`*types*`]`	The `directory` line implies an entire directory search with all non-directory files to be included as if entered as `file` lines. It includes the capability of specifying a `filter`, that is a directory component which must be present in recursive directory searches. The `root` specifies the root of the directory, or "`.`" to indicate static content. dir is the directory underneath the root to search from, which can be given as "." to start at the root itself. The `filter` specifies the filter component which must be in the directory, which implies a recursive descent. It can be given as + for a recursive descent with no filter, or "." for no recursive descent (just the contents of the actual directory). You can also specify the types of operation, which default to "channel". types can be "channel", "provider", or "channel,provider", and "channel+provider".
`entry:` *name*	The `entry` specifies the entry name used in the `.par` file. If not specified, it defaults to the name from the `from:` line.
`desc:` *text*	Any number of `desc` lines may appear, and are concatenated together as a user-visible description packaged with the entry.

Operations

Each operation (`op`), in the export file or on the command line, must be specified as a comma separated list of keywords that can have values, most of which are optional. The operations are in a blank or space separated list. Each operation is in the following format.

`dpnode=`*dn*`,entry=`*name*`,provider[=`*name*`],channel[=`*name*`],container=`*name*`[,avail=`*name*`,selected]`

dpnode

This specifies the distinguished name in the directory server (or the keyword global) for the display profile document that this operation is targeted at. This may not apply if the context it is being specified in has already provided this. For example, if the `import` subcommand defines the distinguished name, the distinguished name in the file is ignored.

entry

This specifies the entry name in the `.par` file. This is not needed if the:

- `.par` file only contains one entry, as the operation defaults to that entry
- Operation is already associated with an entry such as the autoextract option for an entry.

The `par` utility defaults to the first entry in the file if an entry is not specified.

provider

This indicates that a provider extraction is to take place. If the name is missing, it simply uses the name packaged with the provider in the `.par` file.

channel

This indicates that a channel extraction is to take place. If the name is missing, it simply uses the name provided with the channel in the `.par` file.

container

This applies only to channel extractions and indicates which container the channel is to be inserted into. If omitted, the channel is inserted into the `channels` element at the display profile document root.

avail

This applies only to channel extractions and indicates a container whose `avail` (or available) list is to receive a reference to the new channel. If omitted, no new channel reference is created.

selected

This applies only if `avail` was used. It indicates that the container whose `avail` list received a reference, also has a reference placed in its `selected` list.

If the `op` information is in both the `par import` command and in the `.par` file, the command information takes precedence.

PAR Files

This section contains supplemental information on the PAR file format. You do not require this information to run the `par` command.

Overview

The PAR file is a JAR file with manifest entries for transporting channels, providers, and their associated files. It is intended allow flexibly when installing providers, channels, or both. The `.par` file contains 4 major types of files:

1. XML documents containing the provider and / or channel information for the display profile. This document is a `parEntry`, as described in the display profile DTD. This `parEntry` contains a channel, a provider, or a channel / provider combination.
2. Class files associated with the provider and / or channel.
3. Property based files. These are general files associated with the channel, portlet, or provider (usually the channel), which have to be deployed underneath some configurable root on the portal server.
4. Static content files. These are files deployed as documents on the web server.

Par File Contents

The `.par` file must contain the headers listed in the following table. This table contains two columns: the first column lists the required global headers; the second column gives a brief description of the corresponding header.

`PS-Version`	Specifies a portal server specific version number of the `.par` file. Also verifies that this is a `.par` file.
`PS-DefaultEntry`	Names the entry used for operations involving an unnamed entry.

`PS-DPRoot`
`PS-ClassRoot`
`PS-PBFileRoot`
`PS-StaticRoot`
	Indicates the root directories in the archive for `parEntry` documents, classes, property based files, and static content, respectively. If unspecified, the corresponding files are rooted at the top of the archive.

In the `.par` file, there must be a named entry for each `parEntry` XML file. The section for each named entry may contain the headers in the following table. This table contains two columns: the first column lists the possible headers; the second column gives a brief description of the corresponding header.

`PS-EntryName`	Specifies the command visible name of the entry.
`PS-AutoExtract`	Specifies the autoextract operation for the entry, if one exists.
`PS-Include`	Contains a comma separated list of archived files specified by their actual archive path. The path implies what type of file they are according to the “root” specifications. The files are appended with a parenthesized number which corresponds to the types of operations the file applies to (a mask with 1 for provider, 2 for channel). This can be ignored if there are no files other than the XML document associated with the entry.

If the `.par` file contains only one entry, entries need not be named in manipulating the file since the default entry is the entry used if none is named.

Chapter 14

rwadmin

This chapter contains the following sections:

Description

The `rwadmin` command enables the administrator to manage the Rewriter data available in the Sun Java System Identity Server Rewriter service.

Syntax

The `rwadmin` command syntax is described in this section.

Short Named Format

`rwadmin list -u` *uid* `-w` *password* `[-l` *locale*`] [-b] [-h]`

`rwadmin store -u` *uid* `-w` *password* `[-l` *locale*`] [-b] [-h]` *filename*

`rwadmin get -r` *rulesetname* `-u` *uid* `-w` *password* `[-l` *locale*`] [-b] [-h] [`*filename*`]`

`rwadmin remove -r` *rulesetname* `-u` *uid* `-w` *password* `[-l` *locale*`] [-b] [-h]`

Long Named Format

`rwadmin list --runasdn` *uid* `--password` *password* `[--locale` *locale*`] [--verbose] [--version] [--help]`

`rwadmin store --runasdn` *uid* `--password` *password* `[--locale` *locale*`] [--verbose] [--version] [--help]` *filename*

`rwadmin get --rulesetid` *rulesetname* `--runasdn` *uid* `--password` *password* `[--locale` *locale*`] [--verbose] [--version] [--help] [`*filename*`]`

`rwadmin remove --rulesetid` *rulesetname* `--runasdn` *uid* `--password` *password* `[--locale` *locale*`] [--verbose] [--version] [--help]`

Subcommands

These subcommands are supported:

- list
- store
- get
- remove

list

Description

This subcommand lists all the available ruleset names.

Syntax

`rwadmin list -u|--runasdn` *uid* `-w|--password` *password*

Example

In the following example, the command displays names of all available rulesets.

```
rwadmin list -u "uid=amAdmin,ou=people,o=sesta.com,o=isp" -w joshua
```

store

Description

This subcommand stores the Rules available in the local file system into Identity Server. If you want to store the DefaultRuleSet, use the following command:

`rwadmin store -u` *uid* `-w` *password* `/resources/DefaultRuleSet.xml`

where `/resources/DefaultRuleSet.xml` is the location of RuleSet stored in `rewriter.jar` file. Note that when this command is executed, if a ruleset with the same ID already exists, no new data is stored. Delete the existing ID and try again.

Syntax

`rwadmin store -u|--runasdn` *uid* `-w|--password` *password filename*

Example

In the following example, the command stores the Rules available at `/opt/data/ExampleRuleSet.xml` into the Identity Server.

```
rwadmin store -u "uid=amAdmin,ou=people,o=sesta.com,o=isp" -w joshua
/opt/data/ExampleRuleSet.xml
```

get

Description

This subcommand gets the RuleSet from Identity Server. If the filename is provided, the retrieved RuleSet is stored in the specified file, or else it is displayed on `stdout` (or the console).

Syntax

`rwadmin get -r|--rulesetid` *ruleset* `-u|--runasdn` *uid* `-w|--password` *password* [*filename*]

Examples

Example 1 In the following example, the command retrieves the RuleSet named ExampleRuleSet from Identity Server and displays it on the console.

```
rwadmin get -r "ExampleRuleSet" -u
"uid=amAdmin,ou=people,o=sesta.com,o=isp" -w joshua
```

Example 2 In the following example, the command retrieves the RuleSet named ExampleRuleSet from Identity Server and saves it in the file `abc.xml` in the `/tmp` directory.

```
rwadmin get -r "ExampleRuleSet" -u
"uid=amAdmin,ou=people,o=sesta.com,o=isp" -w joshua /tmp/abc.xml
```

remove

Description

This subcommand deletes the RuleSet from Identity Server. This command deletes the RuleSet without any warning.

Syntax

`rwadmin remove -r|--rulesetid` *ruleset* `-u|--runasdn` *uid* `-w|password` *password*

Example

In the following example, the command deletes the RuleSet whose name is ExampleRuleSet from Identity Server.

```
rwadmin remove -r "ExampleRuleSet" -u
"uid=amAdmin,ou=people,o=sesta.com,o=isp" -w joshua
```

Options

The following table contains a summary of the `rwadmin` command options. It contains two columns: the first column lists the possible options; the second column gives a brief description of the corresponding option. The `rwadmin` command supports the following options (listed in alphabetical order):

`-b or --verbose`	Specify this argument to `rwadmin` to provide detailed information on what is happening when the command is executed
filename	Specify this option with the `store` subcommand to indicate the file to get the RuleSet data from when importing into the Identity Server software. Specify this with the `get` subcommand to indicate the file in which the retrieved RuleSet data should be stored.

`-h or --help`	Specify this option to `rwadmin` to print out a brief help page to standard output. If no subcommand is present, a generic help page for `rwadmin` is printed. If one of the `rwadmin` subcommands is present, then a brief help page that is specific to the subcommand is printed.
`-l or --locale`	Use this option to have all output messages localized in the specified locale. If not specified, it defaults to system locale.
`-r or --rulesetid`	Use this option to specify the name of the RuleSet to operate upon
`-u or --runasdn`	Specify this option with the distinguished name of the user to use to bind to the Directory Server.
`--version`	Specify this option to `rwadmin` to print descriptive information about the utility, such as its version, legal notices, and other similar information to standard output. Any subcommand and all other arguments are ignored when this option is present.
`-w or --password`	Specify this option with the password of the distinguished name of the user used to bind to the Directory Server.

Chapter 15

rdmgr

This chapter contains the following sections:

Description

The `rdmgr` command is the main command used to work with the Search service. It gives the administrator two types of subcommands: ones that are used to work with resource descriptions (RDs); and ones that are used for database maintenance. The `rdmgr` command is normally run in a search-enabled Sun Java System Portal Server instance directory, which is the */server-instance-directory/deployment_uri* directory. This is the deployment URI path you selected at install time. If you chose the default Portal Server install, this is the `/var/opt/SUNWps/https-`*servername*`/portal` directory. Where the value of the servername is the default Portal Server instance name--the fully qualified name of your Portal Server.

Syntax

The general syntax of the `rdmgr` command is:

```
rdmgr  [subcommand]  [options]  [input]
```

The RD subcommands more specifically follow this syntax:

`rdmgr [-umgdnUL] [-ACSTNPq] [-a` *att*`,`*att*`,...] [-b` *number*`] [-c search.conf] [-i` *charset*`] [-o` *charset*`] [-j` *number*`] [-l` *number*`] [-p` *progress*`] [-r` *number*`] [-s` *schema*`] [-y` *dbname*`] [`*filename*`|-Q` *query*`]`

The database maintenance subcommands more specifically follow this syntax:

`rdmgr [-OXIERGBL] [-ASTDVNP] [-a att,att,...] [-b` *number*`] [-c search.conf] [-j` *number*`] [-l` *number*`] [-p` *progress*`] [-r` *number*`] [-s` *schema*`] [-y` *dbname*`]`

You can use `-l` number to set the log level number for any RD or database subcommand. A setting of 1 (default) logs all the `rdmgr` commands. The higher the number the more detail the log file contains. The possible levels are 1- 100. If this option is not specified, this command assumes the setting defined by the `debug-loglevel` in the `search.conf` file. The log file name is defined by the `rdmgr-logfile` in the `search.conf` file.

Where the `-c search.conf` option gives the location of the `search.conf` file. If you do not use this option, the default value is `config/search.conf` in the current directory. The `search.conf` file lists all the specific search values you have set.

You can use `-p` progress to show the progress of any RD or database subcommand. If you only enter `-p`, the progress is displayed on `stdout`.

Subcommands

The following subcommands are supported:

- Resource Description Subcommands
- Database Maintenance Subcommands
- Usage Message and Version Subcommands

Resource Description Subcommands

Description

The RD subcommands allow an administrator a batch process to insert or replace RDs, merge RDs filtered by a view, retrieve RDs filtered by a view, delete RDs and count RDs.

Subcommands

The following table lists the subcommand in the first column with a brief description in the second column.

`-d`	Delete RDs.
`-g`	Retrieve RDs filtered by a view.
`-m`	Merge RDs filtered by a view.
`-n`	Count the RDs
`-u`	Insert or replace RDs. This subcommand is the default subcommand if none is stated.
`-U`	Dump database in SOIF to stdout.
`-L`	Lists selected fields from the database to stdout. Requires the -a att option.

NOTE If you enter `rdmgr` with no subcommand, the command assumes the `-u` subcommand. If you enter `rdmgr` with no subcommand and a query (`-Q`), the command assumes the `-g` subcommand.

Syntax

`rdmgr [-u|m|g|d|n|U|L] [-ACSTNPq] [-a att,att,...] [-b` *number*`] [-c search.conf] [-i` *charset*`] [-o` *charset*`] [-j` *number*`] [-l` *number*`] [-p progress] [-r` *number*`] [-s` *schema*`] [-y` *dbname*`] [`*filename*`|-Q` *query*`]`

Options

The following is a two-column table that lists the options or arguments in the first column with a brief description in the second column. The following options are supported:

`-A`	Do not use schema aliases in `config/schema.rdm` file in the default search directory. Use with the `u` and `m` subcommands.
`-C`	Do not create database if database is missing. Use with the `u` and `m` subcommands.

`-S`	Disable schema checking. Use with the `u` and `m` subcommands.
`-T`	Operate on the taxonomy. The taxonomy is used for browsing and classifying the database contents and is in the `config/taxonomy.rdm` file in the default search directory. Use with any resource description command.
`-N`	The function you specified in the command works only on the non-persistent data in the resource description. RDs in the database are a merge of persistent and non-persistent data.
`-P`	The function you specified in the command works only on the persistent data in the resource description. RDs in the database are a merge of persistent and non-persistent data.
`-q`	Delete SOIF input file on exit. Use with the `u`, `m`, `g` and `d` subcommands.
`-a` *att*`,`*att*`...`	Specifies attribute view list. The att names are not case sensitive and can be any attribute whether or not they are defined in the schema; for example, author or title. If you have a multi-valued att like class-1, class-2, and class-3, only enter class as the att name.
`-b` *number*	Set the indexing batch size to this number of RDs. Use with the `u` and `m` subcommands.
`-c search.conf`	Specify where the `search.conf` file is. If you do not use this option, the default is the `config/search.conf` file in the default search directory. Other wise, you have to give the full path to the file.

`-i` *charset*`|-o` *charset*	The `-i` option specifies the character set of the input SOIF stream. The `-o` option specifies the character set of the output SOIF stream. For example, ISO8859-1, UTF-8, UTF-16. Character sets ISO8859-1 through ISO8859-15 are supported. Use `-i` with the `u`, `m`, and `d` subcommands. Use `-o` with the `g`, `U`, and `L` subcommands.
`-j` *number*	Limits the number of retrieved results. Use with the `u` subcommand. If not stated, the default value is unlimited except with the Q option (default 20).
`-l` *number*	Set log level to number. A setting of 1 (default) logs all the rdmgr commands. The higher the number the more detail the log file contains. The possible levels are 1- 100. This works with all subcommands.
`-p {stdout|stderr|`*filename*`}`	Prints or displays progress to stdout, `stderr` or the *filename* file. This works with all subcommands. Timing information is reported in seconds.
`-r number`	Use with the `progress` option. A report is generated every number of RDs. The default is 500. Use with the `u`, `m`, `g`, `d` and `U` subcommands.
`-s` *schema*	Specifies the schema definition file. If you do not use this option, the default is the `config/schema.rdm` file in the search server instance directory.
`-y` *dbname*	Specifies the search database name. If you are running this command on any database other than the default one, you need to use this option. The default database is the database defined in the `config/search.conf` file labeled `datbase-name=`*logicaldbname*.
filename`|-Q query`	This input option is used with the `u`, `m`, `g` and `d` subcommands. The *filename* is a file of RDs using the default schema (use `-s` option for any other schema) in SOIF format. The *query* is any regular search query.

Examples

Example 1 In the following example, the entire default database of resource descriptions is printed out to `stdout` in UTF-8 SOIF format.

Set environment variable `LD_LIBRARY_PATH` to `/opt/SUNWps/lib`. In the `/var/opt/SUNWps/https-sesta.com/portal` directory, type:

PortalServer-base`/bin/rdmgr -U`

Example 2 In the following example, all the resource descriptions that have java any where in them are deleted.

In the default search directory of `/var/opt/SUNWps/https-sesta.com/portal`, type:

PortalServer-base`/bin/rdmgr -d -Q java`

Database Maintenance Subcommands

Description

The database subcommands allow an administrator to optimize a search database, to truncate or empty a database, to reindex a database, to delete expired RDs from a database, and recover a database.

Subcommands

The following table is a two-column table that lists the subcommand in the first column with a brief description in the second column.

`-O`	Optimize the database. If you are running this subcommand on any database other than the default one, you need to use the `-y` option. The default database is the database defined in the `config/search.conf` file labeled `datbase-name=`*logicaldbname*. For example, the default value is `datbase-name=default` and the default database directory is `db/default`. Databases do not normally need to be optimized.

`-X`	Truncate or empty a database. If you are running this subcommand on any database other than the default one, you need to use the `-y` option. Disk space used for indexes is recovered, but disk space used by the main database is not recovered, instead, it is reused as new data is added to the database.
`-I`	Reindex a database. If you are running this subcommand on any database other than the default one, you need to use the `-y` option.
`-E`	Delete expired RDs from a database. If you are running this subcommand on any database other than the default one, you need to use the `-y` option.
`-R`	Recover all databases. This is a global command and takes no options. All database processes, including other `rdmgr` instances and the main search server must be stopped before running this command.
`-G`	Repartition the database. This command takes no options. The partitions are defined in the `config/search.conf` file labeled database-partitions=p1,p2,p3,.... where p1, p2, and p3 are the filenames of the partitions. The server needs to be restarted after running this command.
`-B`	Completely deletes the database. Recovers all the disk space. There should be no indexing happening and the Portal server has to be off when you run this subcommand.
`-L`	Lists selected fields from the database to stdout. Requires the `-a` att option. If you are running this subcommand on any database other than the default one, you need to use the `-y` option.

Syntax

```
rdmgr   [-OXIERGBL]  [-ASTDVNP]   [-a att,att,...] [-b number] [-c
search.conf] [-j number] [-l number] [-p progress]  [-r number] [-s schema]
[-y dbname]
```

Options

The following is a two-column table that lists the options or arguments in the first column with a brief description of the corresponding option in the second column. The following options are supported:

`-A`	Do not use schema aliases in `config/schema.rdm` file in the default search directory. Use with the `I` subcommand.
`-S`	Disable schema checking. Use with the `I` subcommand.
`-T`	Operate on the taxonomy. The taxonomy is used for browsing and classifying the database contents and is in `config/taxonomy.rdm` file in the default search directory. Use with `O`, `X`, `I`, `E`, `B`, `U`, and `L` subcommands.
`-D`	Update the database only; do not update the index. Use with `E` and `X` subcommands.
`-V`	Update the index only; do not update the database. Use with `E` and `X` subcommands.
`-N`	The function you specified in the command works only on the non-persistent data in the resource description. RDs in the database are a merge of persistent and non-persistent data. Use with `I`, `E`, `U`, and `L` commands.
`-P`	The function you specified in the command works only on the persistent data in the resource description. RDs in the database are a merge of persistent and non-persistent data. Use with `I`, `E`, `U`, and `L` subcommands.
`-a` *att*`,`*att*`...`	Specifies attribute view list. The `att` names are not case sensitive and can be any attribute whether or not they are defined in the schema; for example, `author` or `title`. If you have a multi-valued `att` like class-1, class-2, and class-3, only enter class as the `att` name.
`-b` *number*	Set the indexing batch size to this number of RDs. Use with the `I` subcommand.

`-c search.conf`	Specify where the `search.conf` file is. If you do not use this option, the default is the `config/search.conf` file in the default search directory. Other wise, you have to give the full path to the file.
`-j` *number*	Limits the number of retrieved results. Use with the `E` subcommand. If not stated, the default value is unlimited.
`-l` *number*	Set log level to number. A setting of 1 (default) logs all the rdmgr commands. The higher the number the more detail the logfile contains. The possible levels are 1- 100. This works with all subcommands.
`-p {stdout|stderr|`*filename*`}`	Prints or displays progress to `stdout`, `stderr`, or *filename*. This works with all subcommands.
`-r` *number*	Use with the `progress` option. A report is generated every number of RDs. The default is 500. Use with the `u`, `m`, `g`, `d`, and `U` subcommands.
`-s` *schema*	Specifies the schema definition file. The default is the `config/schema.rdm` file in the default search directory.
`-y` *dbname*	Specifies the search database name. If you are running this command on any database other than the default one, you need to use this option. You do not need to use this option for the default database. The default database is the database defined in the `config/search.conf` file labeled `datbase-name=`*filename*.

Examples

Example 1 In the following example, up to 13 RDs are removed from the database if they are expired. The progress report to `stdout` prints the elapsed time in seconds and the number of RDs processed so far after every five resource descriptions.

In the default search directory, type:

PortalServer-base`/bin/rdmgr -E -j 13 -p stdout -r 5`

Example 2 The following example shows how to recover all the search databases and makes the Search engine available again. Use this command to release stale locks in the database and to roll back incomplete data transactions. Stale locks and incomplete transactions can result from a database process being abnormally terminated.

If the Search engine is ‘hung’ or not responding, in the default search directory, type:

PortalServer-base`/bin/rdmgr -R`

Usage Message and Version Subcommands

Description

These subcommands can be used to display the usage message and view the version information.

Options

The following table lists the subcommands in the first column and a brief description of the corresponding subcommand in the second column.

`-h or -?`	Show the usage message.
`-v`	Show version information

Return Codes

The `rdmgr` command returns the following return codes to the shell.

`0`	Success
`1`	Failure

Chapter 16

sendrdm

This chapter contains the following sections

Description

The `sendrdm` command provides a mechanism for a CGI or command-line based search. An RDM (resource description manager) request is sent in SOIF format to the Search server. This command is normally run in a search-enabled Sun Java System Portal Server instance directory, which is the */server-instance-directory/deployment_uri* directory. This is the deployment URI path you selected at install time. If you chose the default Portal Server install, this is the `/var/opt/SUNWps/https-`*servername*`/portal` directory. Where the value of the *servername* is the default Portal Server instance name--the fully qualified name of your Portal Server.

NOTE For the default installation, set the environment variable `LD_LIBRARY_PATH` to *PortalServer-base*`/lib`.

Syntax

The syntax of the sendrdm command is:

```
sendrdm [-dv] [-t n] [-u uri] RDM-in [RDM-out]
```

Options

The following table contains a summary of the sendrdm command options. It contains two columns: the first column lists the possible options; the second column gives a brief description. The command supports the following options (listed in alphabetical order):

`-d`	Debug mode. The default is off. This option turns it on.
`-t` *n*	Specifies time in seconds. Command times out after n seconds. Default is 300 seconds.
`-u` *uri*	Designates the uri directory (enter full path) for the server you are importing from.
`-v`	Version.
`RDM-in`	The RDM request file name. This is a required argument.
`RDM-out`	The RDM result file name. The default is standard out.

Example

The following example imports from a Compass Server 3.01x using /rdm/incoming as the URI with the timeout set to one hour. As root, in the /var/opt/SUNWps/https-*servername*/portal directory, type:

PortalServer-base/lib/sendrdm -t 3600 -u /rdm/incoming rdmquery.soif result.soif

The content of rdmquery.soif is:

```
@RDMHEADER { -
    catalog-service-id{48}: x-catalog://frankie.sesta.com:89/Compass-2
    rdm-type{10}: rd-request
    rdm-version{3}: 1.0
    rdm-query-language{8}: gatherer
}
@RDMQUERY { -
    scope{3}: all
}
```

Example

Chapter 17

StartRobot and StopRobot

This chapter contains the following sections

StartRobot

Description

The `StartRobot` script can be used by an administrator to start the robot manually. Usually this script is used by the scheduler to start the robot at set times (cron job). The `StartRobot` command is in the `/var/opt/SUNWps/https-`*`servername`*`/portal` directory.

Syntax

```
StartRobot
```

Options

There are no options.

StopRobot

Description

The `StopRobot` script can be used by an administrator to stop the robot manually. Usually this script is used by the scheduler to stop the robot at set times (cron job). The `StopRobot` command is in the `/var/opt/SUNWps/https-`*servername*`/portal` directory.

Syntax

```
StopRobot
```

Options

There are no options.

Part III

Configuration Files

Desktop Configuration Properties File

This chapter contains the following sections:

Overview

The `desktopconfig.properties` file defines server-specific parameters that the Desktop reads during initialization. Any changes to this file require a server restart in order to go into effect. By default, this file is in the `/etc/opt/SUNWps/desktop` directory.

Parameters

Parameters marked as Internal are not customizable. So you can only configure the debug level and the base directory for additional classes. The following is a two column table; the first column lists the parameter with its default value and the second column gives a description of its function and possible values.

debugLevel=error	Level of debugging messages to be produced by Desktop. Debug output is stored in /var/opt/SUNWam/debug/desktop.debug file. Use caution when increasing the value of logLevel for excessive logging causes intensive IO operations leading to performance degradation. Possible values are: (in the ascending order of less to more logging) off, error, warning, message, or on. • off: No logging • error: Log errors only • warning: Log errors and warning • message: Log everything
perfLevel=off	[Internal] Level of performance metrics to be logged by Desktop. Output is stored in /var/opt/SUNWam/debug/desktop.perf file. Under production environment, this parameter should always be off. Possible values are: off or message. • off: No performance metrics logged • message: Log all performance metrics
serviceAppContextClassName=com.sun.portal.desktop.context.DSAMEServiceAppContext	[Internal]
templateBaseDir=/etc/opt/SUNWps/desktop/	Root directory under which all template files are located.
providerClassBaseDir=/etc/opt/SUNWps/desktop/classes	Root directory under which the customer is allowed to place provider classes, whether those are overriding the bundled providers, or their own new providers (usually the case). They must be placed in this directory, either in a jar at the top level, or in a com (or whatever) package directory.

jjspCompilerWARClassPath=<Used only on application server> jjspCompilerWARClassPath=/export/home/ias60sp3/ias/APPS/modules/ps/WEB-INF/lib	[Internal] Used only on application server.
defaultDesktopType=default	[Internal] Default desktop type used by the ErrorProvider when DesktopAppContext is available but DesktopContext is not available.
getterPoolMinSize=0	[Internal]
getterPoolMaxSize=0	[Internal]
getterPoolPartitionSize=0	[Internal]
callerPoolMinSize=0	[Internal]
callerPoolMaxSize=0	[Internal]
callerPoolPartitionSize=0	[Internal]
cookiePrefix=desktop	[Internal] Prefix used for all desktop cookies.
templateScanInterval	Defines number of seconds between scans (checking for changes) of the template files in the /etc/opt/SUNWps directory. This interval can improve the performance and scalability because the server uses the cached information between scans. The default value is 30 seconds.

Search Configuration Properties File

This chapter contains the following sections:

Overview

In the default installation, the `search.conf` file is in the `/var/opt/SUNWps/https-instancename/portal/config` directory. The `search.conf` file lists all the specific search values you have set. The `/opt/SUNWps/samples/config` directory contains a sample `search.conf` file.

The default install assigns `$CSROOT` to `/var/opt/SUNWps/http-instancename/portal`, `$CSBIN` to `/opt/SUNWps/bin`, and `$CSLIB` to `/opt/SUNWps/lib`. Most of these parameters can be changed in the Sun Java System Application Server Enterprise Edition administration console under Search Server Settings or Search Server Advanced Settings.

Parameters

The following is a two column table. Column one gives the possible parameters you can change and the default value of the parameter, and column two provides a brief description of the corresponding parameter.

Table 19-1 `search.conf` File Parameters

`csid=x-catalog://$ HOST:$PORT/$N ICK`	Defined at installation. Server identifier string, mainly for backward compatibility with Compass Server.
`bindir=$CSBIN`	Defined at installation. Location of binaries.
`database-directory=$CSROOT/db`	Defined at installation. Location of database (used by server).
`database-root=$CSROOT/db`	Defined at installation. Location of database (used by indexer).
`database-max-concurrent=8`	Limits the number of server threads that can access the database at any one time. You can change this value for performance reasons, but it should be set to about 1.25 times the number of index threads for best performance.
`database-name=default`	The logical database name. You can change this value to another database including an external one.
`database-logdir=db`	Directory where database transaction logs are kept.
`security-mode=OFF`	Enables or disables document level security. Can be reset in the administration console under Server Settings.
`security-manager=com.sun.porta l.search.rdms erver.DSameSe curityManager`	Security manager class name. Do not edit.
`security-dsame-group=OFF`	Whether to use group in addition to user role for security control.
`debug-logfile=$CSROOT/logs/rdmserver.l og`	Logs internal server activity. Defined at installation. Can be reset in the administration console under Server Advanced Settings.
`debug-loglevel=1`	Sets the default log level. Can be reset in the administration console under Server Advanced Settings.
`filters-check-dns=on`	Checks for number of servers aliased to the same address. Can be reset in the administration console under Robot Simulator.
`filters-check-redirect=on`	Checks for any server redirects. Can be reset in the administration console under Robot Simulator.

Table 19-1 search.conf File Parameters *(Continued)*

import-config=$CSROOT/config/import.conf	Defined at installation. Contents generated by the Search server when you define an import agent in the administration console under Database Import.
libdir=$CSLIB	Defined at installation.
logfile=$CSROOT/logs/rdm.log	Log of RDM server requests. Defined at installation. Can be reset in the administration console under Server Advanced Settings.
disable-rdm-log=false	Disables RDM request logging. Can be reset in the administration console under Server Advanced Settings.
classification-stats-during-browse=true	If true, server records how many documents are found in each browse category.
browse-root-classification=false	Whether to browse for documents at the root of the category tree.
search-logfile=$CSROOT/logs/searchengine.log	Search engine logfile. Defined at installation. Can be reset in the administration console under Server Advanced Settings.
search-max-index-batch=2000	Maximum number of documents in each index batch.
search-query-threads=6	Number of search query threads. Should be set to 3-6 threads per CPU that you wish to utilize.
search-index-threads=1	Number of search index threads. Usually left at 1.
search-index-type=AWord	The format of the search engine index. Do not edit.
search-index-partition-size=32	The blocking factor used during index merges. Do not edit.
search-dictionary-type=partial	Format of the search dictionary. Do not edit.
search-lookup-limit=-1	Controls the timeout (milliseconds) of slow wildcard searches. -1 means unlimited.
search-highlights=true	Enable search result highlighting.
search-max-passages=3	Maximum number of dynamic summary passages to generate.

Table 19-1 search.conf File Parameters *(Continued)*

search-passage-context=6	Size of context (in words) around each highlight passage.
#search-field-multipliers="title 1.0"	Search weights assigned to different document fields. Can be a comma separated list.
rdmgr-logfile=$CSROOT/logs/ rdmgr.log	Log file for the indexer process. Defined at installation. Can be reset in the administration console under Server Advanced Settings.
schema-description=$CSROOT/config/schema.rdm	The default Search Engine Schema. Defined at installation.
server-description=$CSROOT/config/server.rdm	The RDM server description returned by server description requests. Defined at installation.
server-root=$CSROOT	Server instance root directory. Defined at installation. Can be reset in the administration console under Server Settings.
taxonomy-database-name=taxonomy	The logical name of the taxonomy index database.
taxonomy-description-refresh-rate=3600 -> 60	Polling interval for automatic taxonomy reloads.
taxonomy-description=$CSROOT/config/taxonomy.rdm	The RDM Taxonomy definition. Edit using the Category Editor under Categories. Defined at installation.
tmpdir=$CSROOT/tmp	Defined at installation. Can be reset in the administration console under Robot Crawling.
robot-refresh=30000	Number of milliseconds between refreshes of the Robot Control page of the administration console.
admin-category_editor_nodes_per_page=25,50,100,250 ,500,-1	List of available choices, defining the maximum number of categories displayed per page. -1 = display all tree.
admin-category_editor_max_combo_element=10	Maximum number of elements in the category editor drop down select list of target categories.

The following parameters are not used:

filters-check-virtual

```
multiple-classifications
reports-exclude-gv-queries
reports-exclude-browse
rdmgr-pidfile
rlog-max-logs
```

Chapter 20

XML and Schema Files

The Sun Java System Portal Server registers its services into the Sun Java System Identity Server Service Management Services (SMS) framework. This occurs during the pre-installation of the Portal Server and post-installation for Identity Server software.

NOTE In general, any service-related data that is not server-specific is stored in the Identity Server directory. Server-specific data can be stored in properties files that are local to the specific server.

SMS provides a mechanism for services to define and manage their configuration data by using an Extensible Markup Language (XML) file that adheres to the SMS Document Type Definition (DTD). The definition of the configuration parameters through the XML file is called the schema for the service. Each Portal Server service (Desktop, Netmail, Rewriter, and Search) has its own XML and properties files for presenting and modifying service specific data.

Within the Identity Server framework, Portal Server defines services related to the following functional areas:

Desktop The SunPortalDesktopService includes data associated with the Desktop component, including the display profile and other configuration parameters associated with the Desktop.

Search Engine The SunPortalSearchService defines the data associated with the Search component, such as the search person and search instances. One or more instances of Search service instances can be defined.

NetMail The SunPortalNetMailService includes data associated with the NetMail application primarily consisting of the user's preferences.

Rewriter The SunPortalRewriterService includes data associated with the Rewriter component, including the named rule sets that control the rewriting operation. The Rewriter API makes reference to the named rule sets that are stored in the directory.

In addition, the Portal Server also uses other DTDs to define LDAP attribute values for the display profile and the Rewriter ruleset.

The Display Profile Document Type Definition (DTD) defines how the Display Profile is structured. The underlying data format for a display profile document is XML. It is intended to define the display configuration for the Desktop. It does that by defining provider, portlet, and channel objects, and their properties. The Rewriter ruleset DTD defines the structure of the ruleset. The Rewriter includes a default ruleset.

The following two column table provides a file path for the various XML, DTD, and schema files used to define the services of the Portal Server (in the first column) and the service which uses the corresponding file (in the second column).

`/opt/SUNWam/dtd/sms.dtd`	Service Management Services Document Type Definition (DTD)
`/opt/SUNWps/export/service/psDesktop.xml`	Portal Server Desktop service definition
`/opt/SUNWps/export/service/psNetMail.xml`	Portal Server NetMail service definition
`/opt/SUNWps/export/service/psRewriter.xml`	Portal Server Rewriter service definition
`/opt/SUNWps/export/service/psSearch.xml`	Portal Server Search service definition
`/etc/opt/SUNWps/dtd/psdp.dtd`	Display Profile document type definition (DTD)
`/opt/SUNWps/web-src/WEB-INF/lib/rewriter.jar`	Rewriter Ruleset document type definition (DTD) under `resources/RuleSet.dtd`
`/opt/SUNWps/web-src/WEB-INF/lib/rewriter.jar`	Default ruleset under `resources/DefaultRuleSet.xml`
`/opt/SUNWps/export/ldif`	
`/opt/SUNWps/export/psDesktop.ldif`	Portal Server Desktop Schema
`/opt/SUNWps/export/psNetMail.ldif`	Portal Server NetMail Schema
`/opt/SUNWps/export/psSearch.ldif`	Portal Server Search Schema

Part IV

Display Profile

Chapter 21

Introduction

This chapter contains the following sections:

What is Display Profile?

The display profile is a series of XML documents describing container management and properties for providers and channels. The display profile creates the display configuration for the Desktop by defining the following items in the XML document:

Provider definition Specifies the name and the Java™ class for the provider. A provider is a template used to generate content, which is displayed in the channel. See “Provider Object” on page 185 for more information.

Channel definition Specifies the run-time configuration of an instance of the provider class. 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. See “Channel Object” on page 183 for more information.

The container channel properties include the display definition about 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 (minimized or detached).

Provider and channel property definitions Specify the values for provider and channel properties. Properties defined in a provider usually specify default values for the channels that are derived from the provider. The display configurations for the channels include properties such as the title, description, channel width, and so on. The properties defined in the channel usually specify the specific value for that channel that is different from the default value. See "Property object" on page 186 for more information.

NOTE If a property is not defined in the channel, then the default value for the property as defined in the provider is used. If a property is defined in the channel, then the value for the property defined in the provider is ignored.

The display profile does not actually define the overall layout or organization of what users see on their Desktops. The display profile exists only to provide property values for channels. However, the display profile does indirectly control some aspects of channel presentation, such as column layout for a table container or how the table container draws channels in a table.

Administering the Display Profile

You can edit the display profile and other Portal Desktop service data through the Sun Java System Identity Server software administration console and the `dpadmin` command. When you edit the display profile, you add, modify, and remove providers, containers, and channels, and edit properties. The Upload XML and Download XML links allow you to upload and download the display profile document. In addition, the Identity Server software administration console provides an Channel and Container Management link in the Portal Desktop attributes page to add channels and containers and edit existing properties. The Channel and Container Management link enables you to define properties when a new channel or container is created. You can also use the Channel and Container Management link to add, modify, and remove channels and containers. See the *Sun Java System Portal Server Administration Guide* for more information.

Chapter 22

Display Profile Document

This chapter contains the following sections:

Document Structure

This chapter describes the overall structure of the display profile documents. The underlying data format for a display profile document is XML.

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 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.

The following sections describe the display profile objects in more detail.

How are the Display Profile XML Documents Stored?

Display profile documents are stored in their entirety as a single attribute in the Sun Java System 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.

The user's display profile is a series of XML documents describing container management and properties for channels. (One display profile document is equivalent to one XML document.) The 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.

To change display profile property values, the providers use the provider APIs (PAPI) to get and set the values. When the channel values are set to the display profile, the PAPI internal implementation uses the Identity Server SDK to set the display profile document in the Identity Server software Desktop service attribute.

NOTE Though possible, you should not edit the display profile using the Identity Server SDK.

Types of Display Profile Documents

This section explains the different types of display profile documents and how to use the Identity Server software administration console to administer them.

Global Display Profile Document Defines display profile elements that are inherited by all users on the system, regardless of the organization or role to which they belong. (Although currently not enforced, you might also want to use the display profile XML document to define the common providers that will be used by everyone.)

Dynamic Display Profile Document Describes container management and properties for channels. This display profile is not ‘used’ to generate a user’s Desktop at runtime, but becomes the default for each newly created organization and role. By default, the dynamic display profile document is blank. To use the dynamic display profile, you need to first populate it.

Organization, Suborganization, or Role Display Profile Shows the display profile for the selected organization, suborganization, or role. When you create a new organization, suborganization, or role, you create a template for this entity. When you create the template for the Desktop service, the initial display profile is set to the dynamic display profile document as mentioned above. Thus, if the dynamic display profile is blank, nothing is filled in.

Most likely, you use this display profile document to customize container management and channel properties to fit the needs of different organizations and roles.

Chapter 23

Display Profile Objects

This chapter contains the following sections:

Introduction

There are three basic types of objects in the display profile: channels, providers, and properties. Every object in the display profile is associated with a DN. 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 display profile XML documents.

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 23-1 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"/>
```

Code Example 23-1 Example Channel Object XML Syntax *(Continued)*

```
        <String name="url"
value="file:///etc/opt/SUNWps/desktop/default/SampleXML/getQuotes.xml"/>
        <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.

Code Example 23-2 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">
            <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>
```

Code Example 23-2 Example Container Object XML Syntax *(Continued)*

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

</Container>
```

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”>`.

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 stores provider definitions that are available to the channel and containers to implement their content generation behavior. 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 class that implements the provider’s behavior is defined in the `provider` attribute. Channels use the `name` attribute values to refer to the provider.

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 23-3 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"/>
```

Code Example 23-3 Example Provider Object XML Syntax *(Continued)*

```
        <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>
```

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.

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.

Object Lookup

At runtime, the system never asks for properties directly from a provider. The request always goes to a channel. If a Java provider object requests a property, it searches the display profile in the following order until it finds the property, or until it reaches the top of the containment hierarchy:

1. Channel's properties
2. Channel's provider's properties
3. Channel's parent's properties
4. Channel's parent's provider's properties

5. Channel's parent's properties (and so on)
6. The global properties bag defined in the display profile root definition

Therefore, when a channel asks for the names of its properties, it gets the set of the union of all the above.

Properties that exist in a provider object are intended to have the semantics of default values for the channel. For example, for a provider XML that defines property title, all channels that are derived from provider XML inherit the title property. If the channel wants to override this property, it can set the value within its own properties.

Chapter 24

Document Priorities

This chapter contains the following sections:

Overview

At runtime, when a user logs in, the system determines the set of documents that makes up the user's display profile document set. The Desktop internal implementation of the display profile (the part that interprets the display profile) determines this set by looking at all of the LDAP nodes that the user belongs to. This can be the organization DN (`dc=sesta.com`), suborganizations, role DNs (`cn=Role1,dc=sesta.com`), and uid (`uid=user,ou=People,cn=Role1,dc=sesta.com`), as well as the global display profile. The display profile documents from each of these LDAP nodes and global display profile are then read (if it exists there), and all of the documents are put into a set. The system sorts the set according to the document priorities. A lower number represents a lower priority. For example, a 1 is a lower priority than a 2. The documents are then sorted from lower number to higher number. See "Process of Merging" on page 197 for more information on this process.

The user level document (`uid=amAdmin,ou=People,...`) is a special case referred to as the base document. Think of the base document as a priority equal to infinity. Thus, it is always the highest number (and hence highest priority). All of the mergers are associated with the base document in sorted order, and the priority setting on a user document is always the highest. The priority attribute used in the `<DisplayProfile>` tag takes the special keyword user to indicate that the current display profile is the user level display profile.

When a merge occurs, it starts at the lowest priority document (lowest number) and proceeds in increasing priority number, until it arrives at the user (base) document.

Thus, the implication of display profile document priorities is that what really matters is the priority number. For example, an organization level document can have a higher priority than a role level document, but it does not have to. It depends on how you need to prioritize these documents for your site.

Specify the display profile document priority in the XML file with the `<DisplayProfile priority= syntax>` tag. You can change the priority by directly editing the display profile XML by using the Sun Java System Identity Server administration console or by using the `dpadmin` command to load the display profile. See Chapter 12, “dpadmin” for more information on the `dpadmin` command.

NOTE Do not assign the same priority to two display profile documents. Doing so causes the Desktop to not appear properly. However, the product does not check for duplicate document priorities.

Examples

Example 1

This example uses two display profiles, one for the organization example and one for the uid bill. When Bill logs in (`uid=bill`) to the Desktop, the bookmark channel titled “Bill’s Bookmarks” is displayed with the following three bookmarks (in that order):

- ACME
- Amazon
- EBay

Code Example 24-1 Display Profile Document for the Organization (dc=acme.com)

```
<DisplayProfile version="1.0" priority="10">
...
<Channel name="Bookmark" provider="BookmarkProvider" merge="fuse">
```

Code Example 24-1 Display Profile Document for the Organization (dc=acme.com) *(Continued)*

```
    <Properties>
        <String name="title" value="My Bookmarks" merge="replace" lock="false"
propagate="true"/>
        <String name="refreshTime" value="600" merge="replace" lock="false"
propagate="true"/>
        <Collection name="targets" merge="fuse" lock="false" propagate="true">
            <String value="ACME home page|http://www.acme.com" merge="replace" lock="false"
propagate="true"/>
        </Collection>
    </Properties>
</Channel>
...
</DisplayProfile>
```

Code Example 24-2 Display Profile Document for the uid=Bill,ou=people,o=acme.com

```
<DisplayProfile version="1.0" priority="10">
...
<Channel name="Bookmark" provider="BookmarkProvider" merge="fuse">
    <Properties>
        <String name="title" value="Bill's Bookmarks" merge="replace" lock="false"
propagate="true"/>
        <Collection name="targets" merge="fuse" lock="false" propagate="true">
            <String value="Amazon|http://www.amazon.com" merge="replace" lock="false"
propagate="true"/>
            <String value="EBay|http://www.ebay.com" merge="replace" lock="false"
propagate="true"/>
        </Collection>
    </Properties>
</Channel>
...
</DisplayProfile>
```

Example 2

This example uses three display profiles, the global display profile, the display profile for the organization acme, and the display profile for the role hradmin. When the user who is assigned to the hradmin role logs in to the Desktop, the TemplateTableContainer appears with the following channels:

- UserInfo
- MailCheck

- SampleSimpleWebService

Code Example 24-3 Global Display Profile Document

```
<DisplayProfile version="1.0" priority="0">
...
<Container name="TemplateTableContainer" provider="TemplateTableContainerProvider"
merge="fuse">
    <Properties>
        ...
    </Properties>
    <Available>
        ...
    </Available>
    <Selected merge="fuse" lock="false">
        <Reference value="UserInfo"/>
    </Selected>
    <Channels/>
</Container>
...
</DisplayProfile>
```

Code Example 24-4 Display Profile Document for the Organization (dc=acme.com)

```
<DisplayProfile version="1.0" priority="0">
...
<Container name="TemplateTableContainer" provider="TemplateTableContainerProvider"
merge="fuse">
    <Properties>
        ...
    </Properties>
    <Available>
        ...
    </Available>
    <Selected merge="fuse" lock="false">
        <Reference value="UserInfo"/>
        <Reference value="Notes"/>
    </Selected>
    <Channels/>
</Container>
...
</DisplayProfile>
```

Code Example 24-5 Display Profile Document for the hradmin Role

```
<DisplayProfile version="1.0" priority="0">
...
<Container name="TemplateTableContainer" provider="TemplateTableContainerProvider"
merge="fuse">
    <Properties>
        ...
    </Properties>
    <Available>
        ...
    </Available>
    <Selected merge="fuse" lock="true">
        <Reference value="MailCheck"/>
        <Reference value="SampleSimpleWebService"/>
    </Selected>
    <Channels/>
</Container>
...
</DisplayProfile>
```

Summary

A display profile document has a low or high priority depending on whether you consider the merge order or the ability to lock as the defining factor.

Without considering locking, the lower numbered display profile document has a lower priority. The lower numbered display profile document gets merged first so the value of a higher priority document overrides the value of a lower priority document. In this sense, the lower numbered document has a lower priority.

However, the lower numbered display profile document can also lock an object so it cannot be affected by a higher numbered document. In this sense, the lower numbered document has a higher priority.

Chapter 25

Merge Semantics

This chapter contains the following sections

Introduction

The display profile is composed of a hierarchy of XML documents. The software can store a display profile document for the user, each role the user belongs to, and the user's organization or suborganization. At runtime, the system merges these multiple display profile documents to deliver a particular portal desktop to the user. This process of merging display profile documents affects the final display profile by potentially changing channel, provider, and property definitions.

The display profile data format contains syntax that defines how these documents are combined. This definition is commonly known as merge semantics.

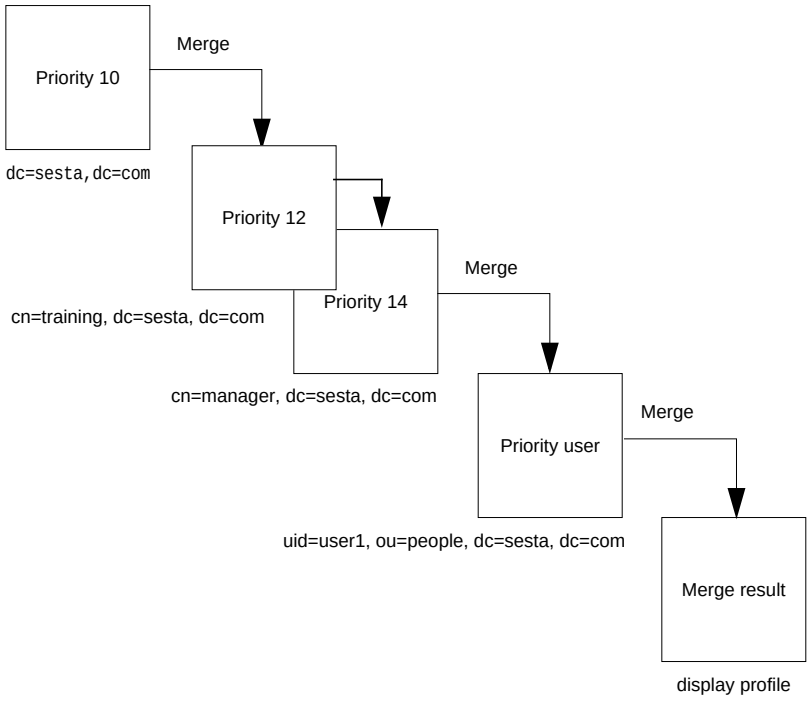
Merge semantics control how attributes are combined as display profile documents from different LDAP nodes (base DN, DN, and role DNs) which are merged to form a single representation (that is, Desktop). Merge semantics assume an ordering to display profile documents. The Sun Java System Portal Server software imposes an additional ordering on Sun Java System Identity Server software roles to simulate a hierarchical structure.

The set of display profile documents for a user consists of: the documents that exist at the user's LDAP organization and suborganization nodes; the documents that exist at each of the user's role nodes; and the document that exists at the user's entry node. Documents do not need to be defined at each of these nodes, but there must be at least one document defined at a node. The set of documents is sorted according to a priority value that the display profile document defines. See Display Profile Document Priorities for more information.

You can visualize the process of document merging as laying one display profile document on top of another. A merge happens where like named channels, providers, and properties fall on top of one another. Merging is based on the name of the display profile object, not the XML structure defined in the display profile document. Like named channels can exist in different containers within the containment hierarchy in the display profile to be merged.

For example, Figure 25-1 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 *Sun Java System Portal Server Administration Guide* for more information.

Figure 25-1 Example of Merge Process



Process of Merging

When a user logs in to the Portal Server Desktop, and after authentication takes place, the system determines the user's display profile by:

1. Locating all the display profile documents for that user by searching through the global display profile, and LDAP organization, suborganization, role, and user nodes that the user belongs to.
2. Placing the retrieved display profile documents in a temporary area, which can be visualized as a bag.

3. Sorting the display profile documents in the bag based on priority, starting at the lowest priority. (The node at which the document was retrieved does not influence the priority sorting. Also, the user display profile document always has the highest priority.)
4. Taking the documents out of the bag, lowest priority first, then placing the next higher level priority document over this document, and applying merge and lock semantics.
5. Continuing Step 4 until all the documents have been taken out of the bag so that the system returns a value to the user that is a merge of the objects found in the documents.

Types of Merge

Overview

The display profile uses the following three types of merges to determine how to combine display profile documents:

replace All the display profile objects defined in the higher priority document completely override the ones defined at the lower one. If the object does not exist in the lower priority document, it is added to the merge result (the object replaces the value in the merge results).

remove The named object is removed from the merge up to this point (the object is removed from the merge results). It no longer exists in the display profile (but it can be re-introduced by another document to be merged). It can be redefined by a higher priority document.

fuse The object from the lower priority document is combined with one from the higher priority document (the object is merged with the value in the merge results).

NOTE The exact meaning of each merge type depends on the display profile object they are applied to.

For channels and providers, fuse has special meaning. The channels themselves are not actually fused together. Rather, fuse indicates that the channel's or provider's properties should be combined. The replace semantic replaces the entire channel or provider, including all properties. The remove semantic removes the entire channel or provider from the merge up to that point.

The display profile `<DisplayProfile>` root node can also have merge semantics. The replace semantic means that all the DP objects defined in the higher priority document completely override the ones defined at the lower one. All merges up to that point are negated and the higher priority document is used as the new base for merging. The remove semantic indicates that all merge results up to the point of this document are to be discarded. The merge begins with the next display profile document found in the sorted set. As with channels and providers, the fuse semantic means that the contained objects (channels and providers) should be combined.

Atomic display profile properties (those that cannot contain other properties) cannot use the fuse semantic. This includes the String, Integer, Boolean, and Reference properties.

The set of properties for a channel consists of the channel's properties plus the channel's provider's properties plus the channel's parent's properties, and so on. You can think of this total set of properties as the channel's single document properties. An implication of document merging is that the total set of properties for a document consists of the set union of the channel's single document properties for all documents in the user's merge set.

Examples

Remove Example

This example uses the merge type remove to modify a container's selected channel list.

The following example code shows how all users' merge set can consist of an organizational level document that has the following display profile fragment.

```
<Container name="TemplateTableContainer"
provider="TemplateTableContainerProvider" merge="fuse">
    <Properties> ... </Properties>
    <Available> ... </Available>
    <Selected merge="fuse">
```

```
        <Reference value="UnixTipoftheDay"/>
    </Selected>
</Container>
```

The “unix tip of the day” describes ways to use UNIX. It is likely that users that belong to the admin role would not find this channel helpful. To remove this channel from everyone with the admin role, define the TemplateTableContainer channel in the admin role document as follows:

```
<Container name="TemplateTableContainer"
provider="TemplateTableContainerProvider" merge="fuse">
    <Properties> ... </Properties>
    <Available> ... </Available>
    <Selected merge="fuse">
        <Reference value="Outages"/>
        <Reference value="SolarisAdmin"/>
        <Reference value="AdminTipoftheDay"/>
        <Reference value="UnixTipoftheDay" merge="remove"/>
    </Selected>
</Container>
```

The preceding sample snippet causes the `<Reference value="UnixTipoftheDay">` to be removed from the admin role display profile.

Replace Example

This example uses the merge type replace to remove channel from all users’ display.

The following example shows how for a particular container, a role admin can ignore all of the channels defined in the organization level. The organization definition resembles the following:

```
<Container name=...>
    ...
    ...
    <Selected>
        <Reference name="X"/>
        <Reference name="Y"/>
```

```
        <Reference name="Z"/>
    </Selected>
</Container>
```

Because the role admin does not want any of the users under that role to have the X, Y, or Z channels, the container is defined as follows:

```
<Container name=...>
    ...
    ...
    <Selected merge="replace">
        <Reference name="A"/>
        <Reference name="B"/>
        <Reference name="C"/>
    </Selected>
</Container>
```

The selected list in the role document's container replaces the selected list in the organization document's container.

Fuse Example

This example uses the merge type fuse to create role-based channel list.

Use the fuse merge semantic to combine non-atomic display profile objects. These objects include Collection and the available or selected channel lists. Here, fuse indicates that all the properties contained in the non-atomic property should also be merged. Using fuse in this way enables the final non-atomic property presented to the user to be build up from various documents.

The following example display profile documents are for a user who belongs to the admin, employee, and movieFreak roles. The selected channels for the user appear at the end.

Code Example 25-1 Display Profile for the Admin Role

```
<Container name="TemplateTableContainer" provider="TemplateTableContainerProvider"
merge="fuse">
    <Properties> ... </Properties>
    <Available> ... </Available>
    <Selected merge="fuse">
```

Code Example 25-1 Display Profile for the Admin Role *(Continued)*

```
        <Reference value="Outages"/>
        <Reference value="SolarisAdmin"/>
        <Reference value="AdminTipoftheDay"/>
    </Selected>
</Container>
```

Code Example 25-2 Display Profile for the Employee Role

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

Code Example 25-3 Display Profile for the movieFreak Role

```
<Container name="TemplateTableContainer" provider="TemplateTableContainerProvider"
merge="fuse">
    <Properties> ... </Properties>
    <Available> ... </Available>
    <Selected merge="fuse">
        <Reference value="NewMoviesReleases"/>
        <Reference value="MovieShowTimes"/>
    </Selected>
</Container>
```

The resultant list of selected channels for the user is as follows, with the available channel list ordered in the same way that the merging was applied, from lower to higher priority:

```
<Container name="TemplateTableContainer"
provider="TemplateTableContainerProvider" merge="fuse">
    <Properties> ... </Properties>
    <Available> ... </Available>
    <Selected merge="fuse">
        <Reference value="Outages"/>
        <Reference value="SolarisAdmin"/>
        <Reference value="AdminTipoftheDay"/>
        <Reference value="Benefits"/>
        <Reference value="EmployeeNews"/>
        <Reference value="NewMoviesReleases"/>
        <Reference value="MovieShowTimes"/>
    </Selected>
</Container>
```

Merge Locking

Any display profile object that is able to be merged can also be locked. When an object is locked, it cannot be affected by merge semantics in higher priority documents. This enables low-priority documents to prevent a high-priority document from using the merge semantics to change particular aspects of the display profile.

Examples

Example 1 This example demonstrates how to use the merge lock feature to force property value for all users. The example shows how to ensure that for a particular organization, all users see the “employee news” channel. The users cannot remove this channel from their display. At the organization level document, the container channel’s selected list is defined as follows:

```
<Selected merge="fuse">
    ...
    <Reference value="EmployeeNews" lock="true"/>
    ...
</Selected>
```

Example 2 This example demonstrates how to use the merge lock feature to forcibly remove channel from all users' display. The example shows how to force the "online games" channel to be removed. In this scenario, users have added this channel to the selected channels list in their user document, so simply removing it from the organization level document's selected channel's list will not work. Instead, the employee and organization lists will be merged together resulting in the "online games" channel being present. To forcibly remove the channel from all users under the organization, the selected channels list is defined as follows:

```
<Selected merge="fuse">
    ...
    <Reference value="OnlineGames" merge="remove" lock="true"/>
    ...
</Selected>
```

Here, the remove semantic removes the channel from merged result, and lock prevents lower priority documents from merging the value back in.

Chapter 26

Display Profile Properties: Overview

This chapter contains the following sections:

Introduction

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 *PortalServer-base*/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 Chapter 25, "Merge Semantics" for a complete discussion of the semantics of the display profile merging.

Display Profile Property Types

Table 26-1 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 26-1 Display Profile General Property Types

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

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 Properties: 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 27-1 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 27-1 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 column and a brief description in the second 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.

Chapter 28

Display Profile Properties: Container Provider Properties

This chapter contains the following sections:

Introduction

This chapter 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 28-1 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.
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.

Table 28-1 Table Container Properties *(Continued)*

Property Tag	Description
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.
channelsIsDetachable	Defines a collection property to contain the isDetachable value for channels in this container.

Table 28-1 Table Container Properties *(Continued)*

Property Tag	Description
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.

Chapter 29

Display Profile Properties: Leaf Building-Block Provider Properties

This chapter 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).

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.

Chapter 30

Display Profile Properties: Service Provider Properties

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
- DiscussionsProvider
- 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.

DiscussionsProvider

The DiscussionsProvider 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 DiscussionsProvider. 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.

Chapter 31

Display Profile Properties: Content Provider Properties

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

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

AddressBookProvider, LotusNotesAddressBookProvider, and MSExchangeAddressBookProvider

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, LotusNotesAddressBookProvider, and MSExchangeAddressBookProvider. 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, LotusNotesCalendarProvider, and MSExchangeCalendarProvider

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

The following is a list of the properties specific to CalendarProvider, LotusNotesCalendarProvider, and MSExchangeCalendarProvider. 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`

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, LotusNotesMailProvider, and MSExchangeMailProvider

The MailProvider works with the Sun Java System Messaging Server software to provide simple mail functionality and single sign-on services. The LotusNotesMailProvider works with the Lotus Notes Server to provide simple mail functionality. The MSExchangeMailProvider works with the Microsoft Exchange Server to provide simple mail functionality.

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).

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="locale list"`	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.

Chapter 32

Display Profile Common Properties for Leaf Providers

Providers have required properties, as well as general properties. This chapter describes the general properties of the providers. It contains the following sections:

Introduction

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 administration console) to store it in LDAP.

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.

Common Properties

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 32-1.

Code Example 32-1 <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>
```

Chapter 33

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.

Part V

Desktop JavaServer Pages

Chapter 34

Overview

This chapter contains the following sections:

Introduction

To generate the rendered Desktop user interface (what the industry refers to as the "presentation"), the Sun Java System 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.

Installation Location

The default set of JSPs are installed in `/etc/opt/SUNWps/desktop/default` directory. The sample portal JSPs are installed in `/etc/opt/SUNWps/desktop/sampleportal` and `/etc/opt/SUNWps/desktop/anonymous` directories. Files in the `/etc/opt/SUNWps/desktop/anonymous` directory are specific to the sample portal Anonymous Desktop.

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.

File Lookup Scenario

The Portal Server software uses the lookup scenario outlined in Code Example 34-1 and Code Example 34-2 on page 253 to find the JSPs it needs. Use this order to decide the final location of your own JSPs.

Code Example 34-1 JSP Lookup Scenario

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*

Code Example 34-1 JSP Lookup Scenario *(Continued)*

```
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:

Code Example 34-2 JSP Lookup Scenario

```
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:

`desktoptype`	For example `default` (set in the 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 `desktoptype` attribute.
`locale`	Preferred `locale` is the user's locale. For example, `en_US` (set by users through the administration console in the "User" setting)

`clientPath`	This is an optional file-path containing client-specific templates; for example, `html` (set through the administration console Client Detection service)
`channelname`	This is the name of the channel; for example, newSingleContainer (set in the display profile)
`provider`	This is the provider name; for example, JSPSingleContainerProvider (set in the display profile)
`templateroot`	This is 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.

Chapter 35

Anonymous Desktop JSPs

This chapter contains the following sections:

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`.)

Chapter 36

JSPs in the default Directory

This chapter contains the following sections:

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	This 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 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. The browseSearch.jsp and searchOnly.jsp files 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. The browseSearch.jsp and searchOnly.jsp files are included only if the user has specified a query; otherwise the category tree (no search) is displayed from browseOnly.jsp. The pageFooter.jsp file 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.

Chapter 37

JSPs in the sampleportal Directory

This chapter contains the following sections:

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.

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/PredefinedSamplesTabPanelContainerProvider 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.

Part VI

Rewriter

Chapter 38

Overview

The Sun Java System Portal Server Rewriter provides an engine for performing URL translation in markup languages and JavaScript™ code. The URLScraperProvider and the XMLProvider in the Desktop and the Sun Java System Portal Server: Secure Remote Access gateway service all use the Rewriter service.

Rewriter scans the content of web pages and identifies the URLs it finds on those web pages. It uses a collection of rules defined in a ruleset to determine the elements of a web page to rewrite. Once Rewriter identifies a URL it can rewrite the URL by:

Expanding Relative URLs to Absolute URLs

The URLScraperProvider is part of the core Portal Server product. In a non-gateway scenario, the URLScraperProvider can be used to expand relative URLs to absolute URLs. For example, if a user is trying to access the site:

```
<a href="../mypage.html">
```

The Rewriter translates this to:

```
<a href="http://www.yahoo.com/mail/mypage.html">
```

where `http://www.yahoo.com/mail/` is the base URL of the page scraped.

URLScraperProvider Limitations

The URLScraperProvider simply tries to display a designated URL in a channel. There's no way to specify parts of a document URL (document) to display. The URLScraperProvider acts much like an HTTP client, in that it makes a request for the content of the specified URL. Just like in a browser, the target URL to scrape must be network visible, or you must have a proxy configured.

The resultant URL scraper channel, however, is not a mini-browser nor is it a frame. Therefore, if you have a link in the content, it effects the whole page, not just the channel. You should not browse inside the URL scraper channel. If you select a link within the channel the browser can interpret the link and replace the currently displayed page (your portal server Desktop) with the contents of the link location.

The appearance of the scraped channel is controlled by whatever is producing the original content. The URLScraperProvider does not modify the content at all and only displays whatever is available through the URL. Since the channel is essentially a cell in an HTML table, it can only display HTML content that is legal to appear in table cells. That is, a frameset cannot be scraped using the URLScraperProvider because a <FRAMESET> tag cannot appear within a <BODY> tag. The URLScraperProvider will also not execute JavaScript code in <HEAD> tags. Because of this, the following scraping scenarios are inappropriate for the URLScraperProvider:

- When an Edit function of some sort is required so that the user can customize the channel.
- When the data comes from a non-HTML, non-web server source, that is, a database or mail server.
- When the data needs to be reformatted in some way for the channel.
- When a more efficient solution is required as the URLScraperProvider will do a request and look up for every Desktop display and user.

When cookies are sent by the origin server, they are forwarded back everytime web content is re-scraped. So the origin should get the cookies it sent as the web content scraped the first time, when portal desktop is updated or reloaded. But those cookies are not expected to be sent back when user clicks on any links in the url scraper channel.

Prefixing the Gateway URL to an Existing URL

In an implementation with a gateway such as the Secure Remote Access, the gateway acts as a proxy for the client and accesses intranet sites and returns responses to the client. The Rewriter translates URLs in downloaded pages so that they point back to the gateway rather than to the original site by prefixing the gateway URL to the existing URL.

For example, if a user tries to access an HTML page on mymachine using the following URL:

```
<a href="http://mymachine.intranet.com/mypage.html>
```

The Rewriter prefixes this URL with a reference to the gateway as follows:

```
<a
href="https://gateway.company.com/http://mymachine.intranet.com/mypage.html>
```

When a user selects a link associated with this anchor, the browser contacts the gateway. The gateway fetches the content of `mypage.html` from `mymachine.intranet.com`.

See the *Secure Remote Access Administration Guide* for more information on using the Rewriter to prefix a gateway URL to an existing URL.

Attributes

The Rewriter provides a Java™ class library for rewriting URL references in various web languages such as HTML, JavaScript, and WML, and in HTTP Location headers (redirections). The Rewriter Service does not consist of any attributes.

To implement the service, create Rewriter rules that define how rewriting is to be done and the data to be rewritten. You can create and edit Rewriter rules through the administration console.

Chapter 39

Supported URLs

Rewriter supports rewriting of all standard URLs as specified by RFC-1738. These URLs are supported whether the protocol is HTTP or HTTPS and regardless of the capitalization of the protocol. For example, hTtP, HTtp, and httP are all valid. Some sample standard URLs are listed below:

`http://www.my.sesta.com`

`http://www.example.org:8000/imaginary/test`

`http://www.example.edu/org/admin/people#andy`

`http://info.example.org/AboutUs/Index/Phonebook?dobbins`

`http://www.example.org/RDB/EMP?*%20where%20name%%3Dobbins`

`http://info.example.org/AboutUs/Phonebook`

`http://user:password@example.com`

Rewriter supports rewriting of some basic non-standard URLs. The information to convert non-standard URLs to a standard format is taken from the base URL of the page where the URL appears and can include the protocol, host name, and path. The back slash (› is supported only when it is part of a relative URL and not part of an absolute URL. For example, http:/ /sesta.com \index.html is rewritten, but http:\ \sesta.com is not.

In addition, URLs with a single slash (/) after the protocol or scheme such as http:/sesta.com are not rewritten.

Chapter 40

Defining Rewriter Rules and Rulesets

This chapter contains the following sections

Overview

The Rewriter modifies the URL portions of various elements that appear on a web page. The Rewriter comes with a default set of rules to determine the elements of a web page to rewrite. A collection of rules for various categories and subcategories is stored in a `.dtd` file and is called a ruleset. The Rewriter rulesets are defined in XML.

The DTD is located in `/opt/SUNWps/web-src/WEB-INF/lib/rewriter.jar` (resources/RuleSet.dtd). Rulesets are used to identify URLs. By default, all strings in web content starting with characters such as “`/`”, `../`, “http” and “https” are considered to be URLs and are candidates for rewriting.

To configure the Rewriter for your implementation, you create a ruleset and define rules in the Rewriter section of the Portal Server Configuration in the administration console. See Administering the Rewriter Service for details on creating and modifying rulesets. You define multiple rules based on the content type in the web pages. For example, the rule required to rewrite HTML content would be different from the rule required to rewrite JavaScript content. Rewriter rules fall into the following broad categories:

- Rules for HTML Content

- Rules for JavaScript Content
- Rules for XML Content

NOTE As Wireless Markup Language (WML) is similar to HTML, HTML rules are applied for WML content.

No rules are required for CSS content.

The ruleset is an XML document and the XML within it must be properly formed. When defining rules in a ruleset, keep the following guidelines in mind:

- All rules need to be enclosed within the <ruleset> </ruleset> tags.
- Include all rules to rewrite HTML content in the <HTML> </HTML> section of the ruleset.
- Include all rules to rewrite JavaScript content in the <JSRules> </JSRules> section of the ruleset.
- Include all rules to rewrite XML content in the <XML> </XML> section of the ruleset.

Rules for HTML Content

HTML content in web pages can be classified into attributes, JavaScript tokens, forms, and applets. Accordingly, the rules for HTML content are classified as:

- Attribute Rules for HTML Content
- JavaScript Token Rules for HTML Content
- Form Rules for HTML Content
- Applet Rules for HTML Content

Attribute Rules for HTML Content

Attribute rules identify the basic attribute tags in HTML pages to rewrite. Rewriter modifies the various occurrences of the defined tags by expanding or prefixing the existing URL. The default ruleset rewrites the following attribute tags:

- action

- background
- codebase
- code
- href
- src
- value
- imagePath
- lowsrc
- archive

The syntax for attribute rules is:

```
<Attribute name="name" [tag="tag" valuePatterns="patterns"]
```

where name specifies the attribute, tag specifies the tag to which the attribute belongs (set to * to match all tags), and patterns specifies the possible patterns to match with the attribute. The tag and valuePatterns parameters are optional.

JavaScript Token Rules for HTML Content

Web pages can contain pure JavaScript code within the JavaScript tags, or they can contain JavaScript tokens or functions. For example, a web page can contain an onClick() function that causes a jump to a different URL. In order for the page to function properly, the value of the onClick() function needs to be translated and rewritten. In most cases, the rules provided in the default ruleset are sufficient to rewrite the URLs in JavaScript tokens. The default ruleset rewrites the following JavaScript tokens:

- onAbort
- onBlur
- onChange
- onClick
- onDblClick
- onError
- onFocus

- onKeyDown
- onKeyPress
- onKeyUp
- onLoad
- onMouseDown
- onMouseMove
- onMouseOut
- onMouseOver
- onMouseUp
- onReset
- onSelect
- onSubmit
- onUnload

The syntax for JavaScript Token rules is:

```
<JSToken>javascript_function_name</JSToken>
```

where javascript_function_name is the name of the function such as onLoad or onClick.

Form Rules for HTML Content

Users can browse HTML pages that contain forms. Form elements, such as input, can take a URL as a value. The default ruleset does not rewrite any form elements. The syntax for form rules is:

```
<Form source="/source.html" name="form1" field="field1">
[valuePatterns="pattern"] />
```

where /source.html is the URL of the HTML page containing the form, form1 is the name of the form, field1 is the field of the form to be rewritten, and pattern indicates the part of the field to be rewritten. All content that follows the pattern specified is rewritten.

The valuePatterns parameter is optional.

Applet Rules for HTML Content

A single web page can contain many applets, and each applet can contain many parameters. The Rewriter rule for URLs in applets should contain pattern matching information for the following:

- source, such as filename.htm
- code, such as classname.class
- parameter name, such as servername
- parameter value, such as some_url

Rewriter matches the values specified in the rule with the content of the applet and modifies the URLs as required. This replacement is carried out at the server and not when the user is browsing the particular web page. A wildcard character (*) can also be used as part of the rule. For example, the parameter name could be *, in which case, the Rewriter does not compare the parameter name in the applet.

The default ruleset does not rewrite any applet parameters.

The syntax for applet rules is:

```
<Applet source="sourcehtml.jsp" code="class" param="parameter_name"
[valuePatterns="pattern"]
```

where /sourcehtml.jsp is the URL containing the applet, class is the name of the applet class, parameter_name is the parameter whose value needs to be rewritten, and pattern indicates the part of the field to be rewritten. All content that follows the pattern specified is rewritten. The valuePatterns parameter is optional.

Rules for JavaScript Content

URLs can occur in various portions of JavaScript code. The Rewriter cannot directly parse the JavaScript code and determine the URL portion. A special set of rules needs to be written to help the JavaScript processor translate the URL.

JavaScript elements that contain URLs are classified as follows:

- JavaScript Variables
- JavaScript Function Parameters

JavaScript Variables

JavaScript variables are again classified into five categories:

- JavaScript URL Variables
- JavaScript EXPRESSION Variables
- JavaScript DHTML Variables
- JavaScript DJS (Dynamic JavaScript) Variables
- JavaScript System Variables

JavaScript URL Variables

URL variables have a URL string on the right hand side. The default ruleset rewrites the following JavaScript URL variables:

- imgsrc
- location.href
- _fr.location
- mf.location
- parent.location
- self.location

The syntax of URL variables in JavaScript content rules is:

```
<Variable type="URL">variable_name</Variable>
```

where variable_name is the name of the variable to be rewritten.

JavaScript EXPRESSION Variables

EXPRESSION variables have an expression on the right hand side. The result of this expression is a URL. The Rewriter appends a JavaScript function for converting the expression to the HTML page as it cannot evaluate such expressions. This function takes the expression as a parameter and evaluates it at the client browser.

The default ruleset rewrites the location JavaScript EXPRESSION variable.

The syntax of EXPRESSION variables in JavaScript content rules is:

```
<Variable type="EXPRESSION">variable_exp</Variable>
```

where variable_exp is the expression variable.

JavaScript DHTML Variables

DHTML variables are JavaScript variables that hold HTML content. The default ruleset rewrites the following JavaScript DHTML variables:

- document.write
- document.writeln

The syntax of DHTML variables in JavaScript content is:

```
<Variable type="DHTML">variable</Variable>
```

where variable is the DHTML variable.

JavaScript DJS (Dynamic JavaScript) Variables

DJS (Dynamic JavaScript) variables are JavaScript variables that hold JavaScript content.

The syntax of DJS variables in JavaScript content is:

```
<Variable type="DJS">variable</Variable>
```

where variable is the DJS variable.

The JavaScript code contained in the variable needs another rule to translate it.

JavaScript System Variables

System variables are variables that are not declared by the user, but that are available as a part of the JavaScript standard.

The default ruleset rewrites the window.location.pathname JavaScript system variable.

The syntax of system variables in JavaScript content is:

```
<Variable type="SYSTEM">variable</Variable>
```

where variable is the system variable.

JavaScript Function Parameters

Function parameters are classified into four categories:

- JavaScript URL Parameters
- JavaScript EXPRESSION Parameters

- JavaScript DHTML Parameters
- JavaScript DJS Parameters

JavaScript URL Parameters

URL parameters are string parameters that directly contain the URL.

The default ruleset rewrites the following JavaScript URL parameters:

- openURL
- openAppURL
- openNewWindow
- parent.openNewWindo
- window.open

The syntax for URL parameters is:

```
<Function type = "URL" name = "function" [paramPatterns="y,y,"] />
```

where function is the name of the function to be evaluated and y indicates the position of the parameter(s) that need to be rewritten. Parameter positions are delimited by commas. For example, in the syntax line the first and second parameters need to be rewritten, but the third parameter should not be rewritten.

JavaScript EXPRESSION Parameters

EXPRESSION parameters are variables within a function that result in a URL when they are evaluated. The syntax for EXPRESSION parameters is

```
<Function type = "EXPRESSION" name = "function" [paramPatterns="y,y,"] />
```

where function is the name of the function to be evaluated and y indicates the position of the parameter(s) that need to be rewritten. Parameter positions are delimited by commas. For example, in the syntax line the first and second parameters need to be rewritten, but the third parameter should not be rewritten.

JavaScript DHTML Parameters

DHTML parameters are native JavaScript methods that generate an HTML page dynamically. For example, the document.write() method falls under this category.

The default ruleset rewrites the following JavaScript DHTML parameters:

- document.write
- document.writeln

The syntax for DHTML parameters is:

```
<Function type = "DHTML" name = "function" [paramPatterns="y,y,"] />
```

where function is the name of the function to be evaluated and y indicates the position of the parameter(s) that need to be rewritten. Parameter positions are delimited by commas. For example, in the syntax line the first and second parameters need to be rewritten, but not the third parameter should not be rewritten.

JavaScript DJS Parameters

Dynamic JavaScript (DJS) parameters such as Cascading Style Sheets (CSS) in HTML are also translated. There are no rules defined for this translation as the URL appears only in the url() function of the CSS. The syntax for DJS parameters is:

```
<Function type = "DJS" name = "function" [paramPatterns="y,y,"] />
```

where function is the name of the function to be evaluated and y indicates the position of the parameter(s) that need to be rewritten. Parameter positions are delimited by commas. For example, in the syntax line the first and second parameters need to be rewritten, but not the third parameter should not be rewritten.

Rules for XML Content

Web pages can contain XML content which in turn can contain URLs and Rewriter can rewrite URLS in XML content.

XML content that contains URLs is classified as follows:

- Tag Text in XML
- Attributes in XML

Tag Text in XML

Rewriter translates XML content based on the tag name.

The default ruleset rewrites the following tags in XML:

- baseroot

- img

The syntax for tag text is:

```
<TagText tag ="attribute" attributePatterns="name=src"/>
```

where attribute is the name of the tag and src is the name of the attribute.

Attributes in XML

The rules for attributes in XML are similar to the rules for attributes in HTML. See "Attribute Rules for HTML Content" on page 308 for additional information. Rewriter translates attribute values based on the attribute and tag names.

The default ruleset rewrites the following attributes in XML:

- xmlns
- href

The syntax for attributes in HTML is:

```
<Attributes>
    <Attribute name="attribute" [valuePatterns="name=src"/>
</Attributes>
```

where attribute is the name of the tag and src is the name of the attribute.

Part VII

Sample Portal

Chapter 41

Understanding the Sample Portal

This chapter contains the following sections:

NOTE 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.

Overview

Conceptually, the Desktop is split into the following three well-defined components (see Figure 41-1 also):

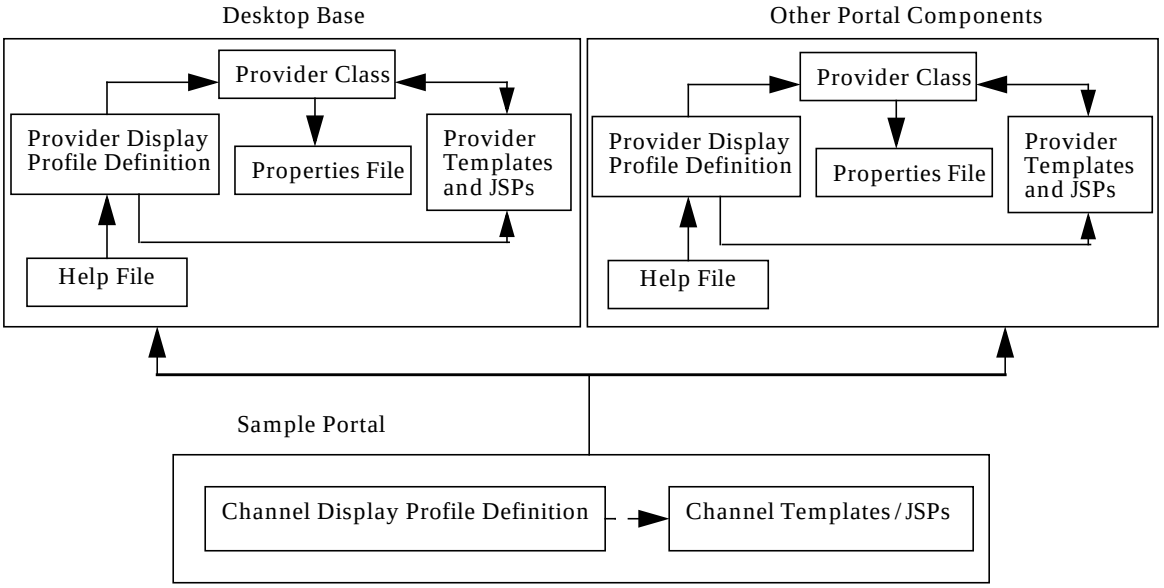
Base Desktop This includes Provider Java classes (based on the Provider API), provider display profile definitions and resource bundles (also referred to as properties files), display profile definitions of channels referenced by base Desktop XML or base Desktop JSPs and templates, default templates 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 41-1 Desktop Components



When you install the Portal Server software, you can choose to install the sample portal. The sample portal is an authentication-less (authless) desktop that consists of containers, channels, portlets, services, and templates which can be used to demonstrate what the Portal Server software is capable of. It 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.

Sample Portal Installation Directories

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

***PortalServer-base*/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.

Sample Desktops

Table 41-1 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 41-1 Sun Java System Portal Server Sample Desktop Containers

Container (Desktop) Type	Description
JSPTabContainer	Generates a JSP-based tab Desktop.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	Generates a JSP-based table Desktop. Creates a Desktop that arranges a maximum of five sub-containers into the channel arrangement. This container is JSP-based. Its provider is JSPTableContainerProvider.
TemplateTableContainer	Generates a template-based table Desktop. Creates a Desktop that arranges a maximum of five sub-containers into the channel arrangement. This container is template-based. Its provider is TemplateTableContainerProvider.
TemplateTabContainer	Generates a template-based tab Desktop. 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.
FrameTabContainer	Generates a frame-based Desktop. 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.

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 *Sun Java System Portal Server Developer's Guide* for more information on extending the providers.

Chapter 42

JSP-Based Desktop

This chapter contains the following sections:

JSPTabContainer

The JSPTabContainer provides a JSP-based tabbed Desktop.

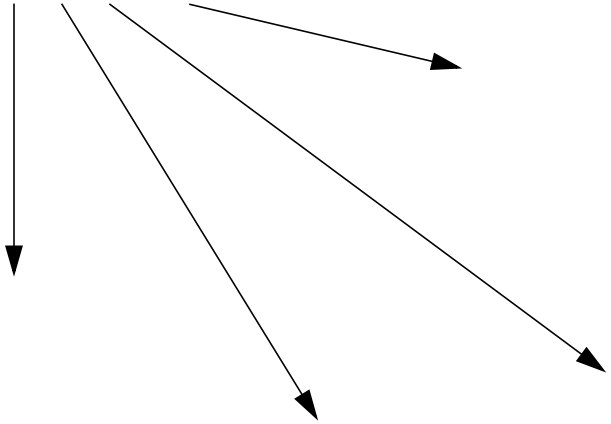
Sample Desktop

Default Layout

By default, the sample portal Desktop based on the JSPTabContainer (see Figure 42-1 on page 323) 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 42-1 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.

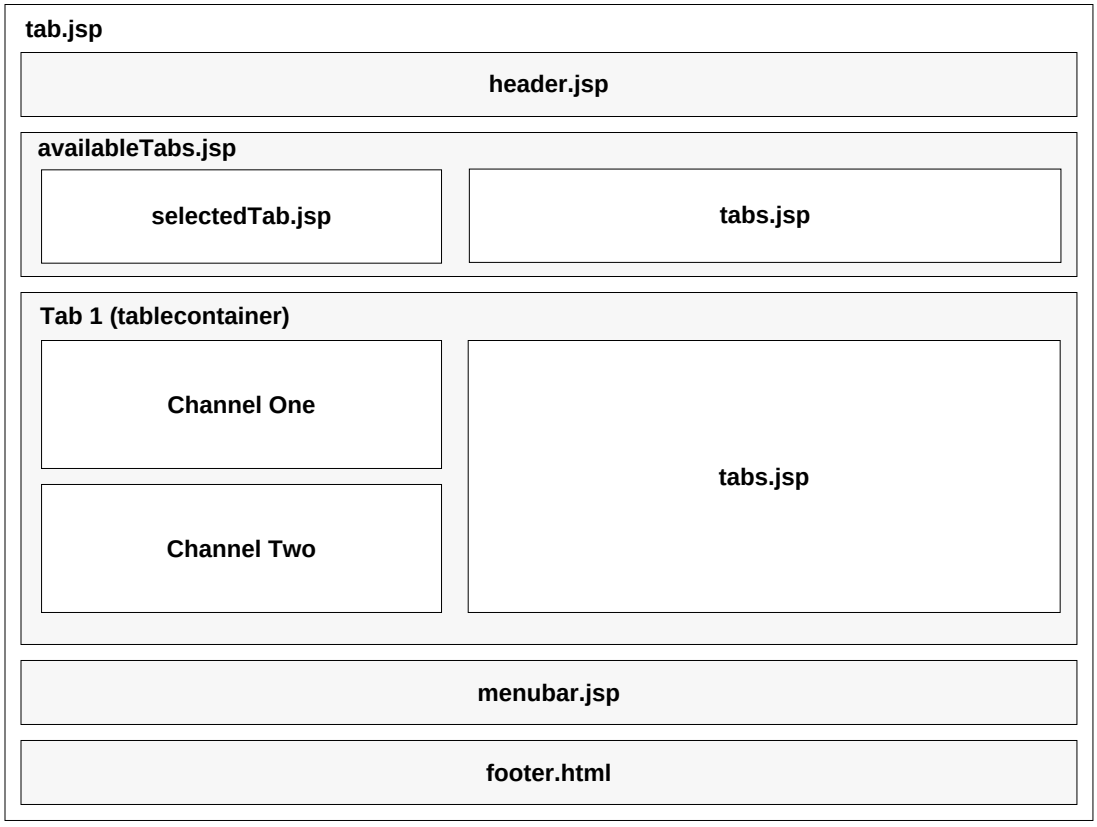
`contentPage`	Set to `tab.jsp`. This draws the Content Page for the tab container.
`editPage`	Set to `tabedit.jsp`. This displays the Edit page for the tab container where new tabs can be added, and existing tabs removed or renamed.
`startTab`	Sets the tab that opens first on the Desktop as MyFrontPageTabPanelContainer.
`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 42-2 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 42-2 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 42-3 on page 328) 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 42-3 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

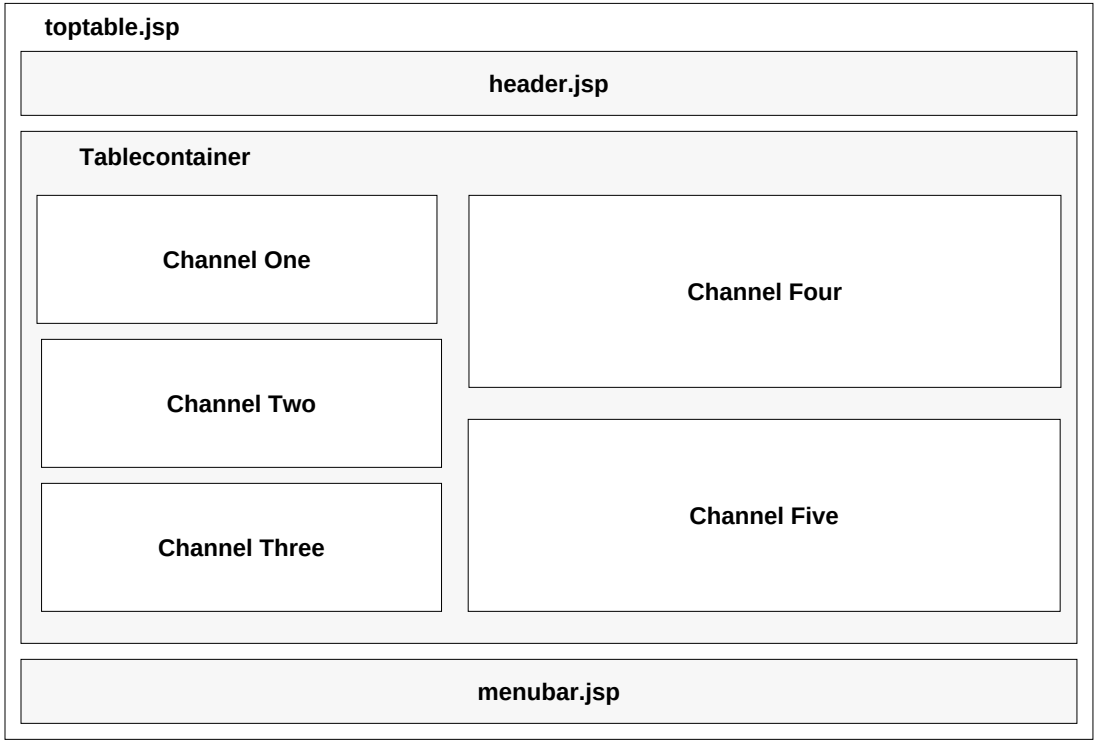
The provider responsible for generating the JSPTableContainer channel is JSPTableContainerProvider. The provider profile is the template which decides the properties for a container channel, but the container channel profile will ultimately decide the values for the container channel attributes. The 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 42-4 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 42-4 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.

Chapter 43

Frame-based Desktop

This chapter contains the following sections:

The FrameTabContainer provides a frame-based table Desktop.

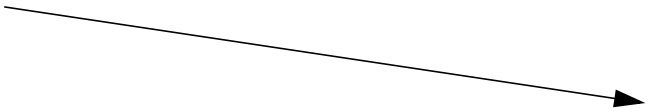
Sample Desktop

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 43-1 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.)

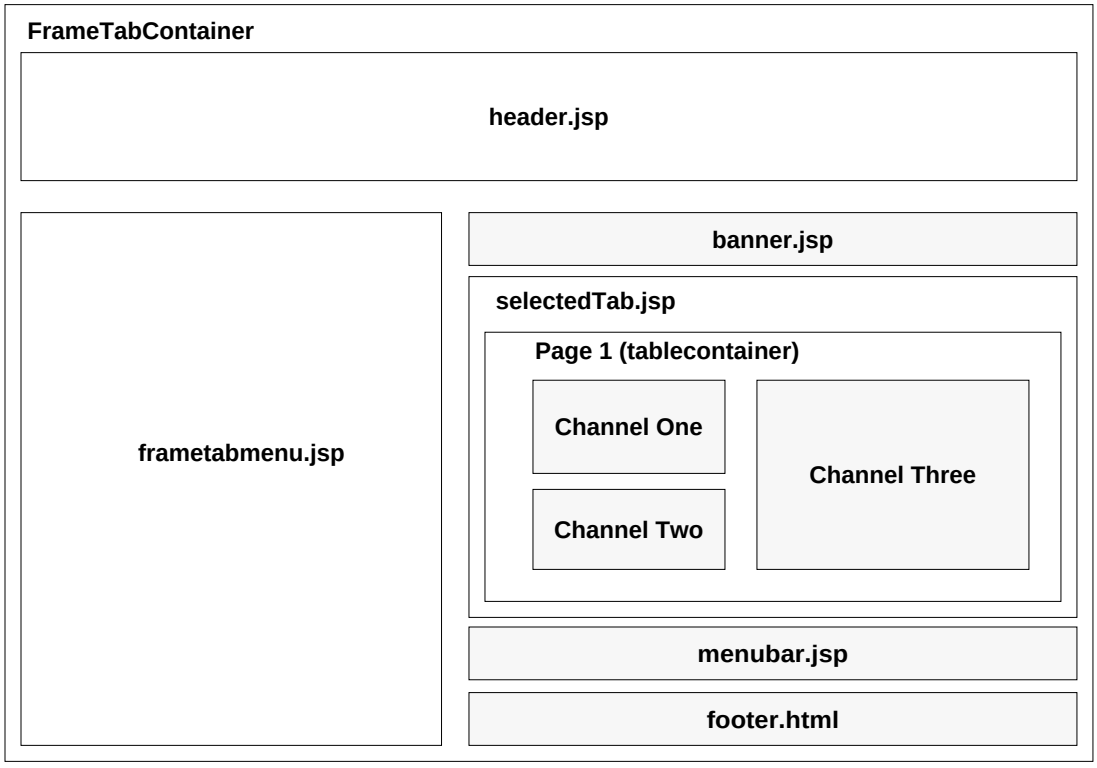
`contentChannel`	Specifies JSPContentContainer as the content channel that provides the Content page displaying channels to add to a user-created page.
`TabProperties`	This collection has entries for each of the four containers that are available in the FrameTabContainer.
`Available`	This list describes 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 43-2 on page 336 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 43-2 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 44

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 44-1 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 44-1 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.

Chapter 45

Global Themes

This chapter contains the following sections:

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.

GlobalThemes Display Profile Definition

The display profile document, *PortalServer-base*/`samples/desktop/dp-org.xml` file, contains the XML fragment for the eight default themes. See this file for the definition of these themes. See the *Sun Java System Portal Server Desktop Customization Guide* for more information on customizing the Global Themes.

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 344 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 45-1 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

Table 45-1 Glossary of Terms *(Continued)*

Term	Description
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

Part VIII

Search Engine Robot

Chapter 56, “Robot Application Functions - Shutdown Functions”

Chapter 46

Overview

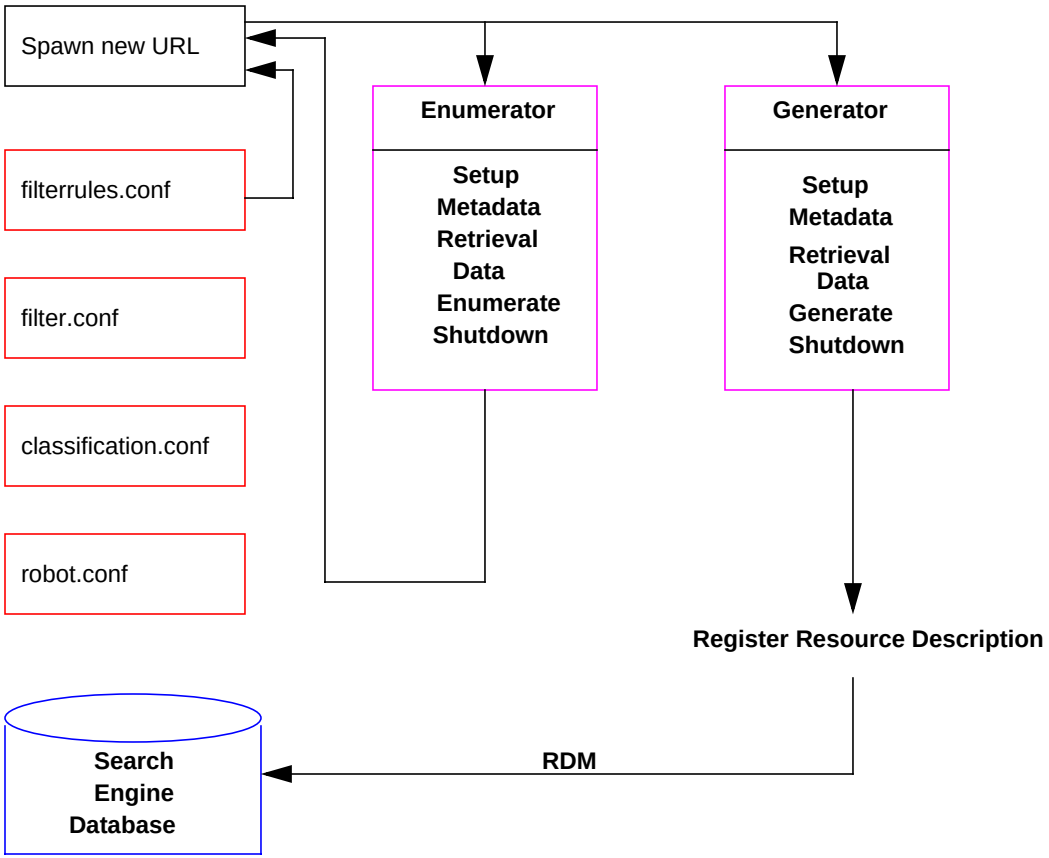
A Search Engine robot is an agent that identifies and reports on resources in its domains. It does so by using two kinds of filters: an enumerator filter and a generator filter.

The enumerator filter locates resources by using network protocols. It tests each resource, and, if it meets the proper criteria, it is enumerated. For example, the enumerator filter can extract hypertext links from an HTML file and use the links to find additional resources.

The generator filter tests each resource to determine if a resource description (RD) should be created. If the resource passes the test, the generator creates an RD which is stored in the Search Engine database.

Figure 46-1 illustrates how the Search Engine robot works. In Figure 46-1, the robot examines URLs and their associated network resources. Each resource is tested by both the enumerator and the generator. If the resource passes the enumeration test, the robot checks it for additional URLs. If the resource passes the generator test, the robot generates a resource description that is stored in the Search Engine database.

Figure 46-1 How the Robot Works



The Robot Application Functions (RAFs) in the `filter.conf` file can be used to create and modify filter definitions. The file `filter.conf` is located in the `/var/opt/SUNWps/http-hostname-domain/portal/config` directory.

The `filter.conf` file contains definitions for the enumeration and generation filters. Each of these filters invokes a set of rules which are stored in the `filterrules.conf` file. The filter definitions contain instructions that are specific to each filter while the filter rules contain the rules used by both filters.

To understand how filter rules are defined, examine the `filterrules.conf` file. Note that you typically need not manually edit this file since you can create filter rules from the administration console.

For an example of filter definitions, examine the `filter.conf` file. Edit the `filter.conf` file only to modify the filters in a way that is not accommodated in the administration console, such as instructing the robot to enumerate some resources without generating resources for them.

Chapter 47

Process Parameters

The `robot.conf` file defines many options for the robot, including pointing the robot to the appropriate filters in `filter.conf` file. For backwards-compatibility with older versions, the `robot.conf` file can also contain the seed URLs.

The administration console is used to edit the file `robot.conf`. Because you can set most parameters by using the administration console, you typically do not need to edit the `robot.conf` file. However, advanced users might manually edit this file in order to set parameters that cannot be set through the administration console.

Table 47-1 lists the user-modifiable parameters in the `robot.conf` file. The first column of the table lists the parameter, the second column provides a description of the parameter, and the third column provides an example.

Table 47-1 User Modifiable Parameters in robot.conf File

Parameter	Description	Example
auto-proxy	Specifies the proxy setting for the robot. It can be a proxy server or a JavaScript file for automatically configuring the proxy.	auto-proxy=”http://proxy_server/proxy.pac”
bindir	Specifies whether the robot will add a bind directory to the PATH environment. This is an extra PATH for users to run an external program in a robot, such as those specified by cmd-hook parameter.	bindir=path
cmd-hook	Specifies an external completion script to run after the robot completes one run. This must be a full path to the command name. The robot will execute this script from the `/var/opt/SUNWps/` directory. There is no default. There must be at least one RD registered for the command to run.	cmd-hook=”command-string”

Table 47-1 User Modifiable Parameters in robot.conf File *(Continued)*

Parameter	Description	Example
command-port	Specifies the socket that the robot listens to in order to accept commands from other programs, such as the Administration Interface or robot control panels. For security reasons, the robot can accept commands only from the local host unless remote-access is set to yes.	command-port=port_number
connect-timeout	Specifies the maximum time allowed for a network to respond to a connection request. The default is 120 seconds.	command-timeout=seconds
convert-timeout	Specifies the maximum time allowed for document conversion. The default is 600 seconds.	convert-timeout=seconds
depth	Specifies the number of links from the seed URLs (also referred to as starting point) that the robot will examine. This parameter sets the default value for any seed URLs that do not specify a depth. The default is 10. A value of negative one (depth=-1) indicates that the link depth is infinite.	depth=integer
email	Specifies the email address of the person who runs the robot. The email address is sent with the user-agent in the HTTP request header, so that Web managers can contact the people who run robots at their sites. The default is user@domain.	email=user@hostname
enable-ip	Generates an IP address for the URL for each RD that is created. The default is true.	enable-ip=[true | yes | false | no]
enable-rdm-probe	Determines if the server supports RDM, the robot decides whether to query each server it encounters by using this parameter. If the server supports RDM, the robot will not attempt to enumerate the server's resources, since that server is able to act as its own resource description server. The default is false.	enable-rdm-probe=[true | false | yes | no]
enable-robots-txt	Determines if the robot should check the robots.txt file at each site it visits, if available. The default is yes.	enable-robots-txt=[true | false | yes | no]

Table 47-1 User Modifiable Parameters in robot.conf File *(Continued)*

Parameter	Description	Example
engine-concurrent	Specifies the number of pre-created threads for the robot to use. The default is 10. This parameter cannot be set interactively through the administration console.	engine-concurrent=[1..100]
enumeration-filter	Specifies the enumeration filter that is used by the robot to determine if a resource should be enumerated. The value must be the name of a filter defined in the file filter.conf. The default is enumeration-default. This parameter cannot be set interactively through the administration console.	enumeration-filter=enumfiltername
generation-filter	Specifies the number of minutes that the robot should collect RDs before batching them for the Search Engine. If you do not specify this parameter, it is set to 256 minutes.	generation-filter=genfiltername
index-after-ngenerated	Specifies the number of minutes that the robot should collect RDs before batching them for the Search Engine. If you do not specify this parameter, it is set to 256 minutes.	index-after-ngenerated=30
loglevel	Specifies the levels of logging. The loglevel values are as follows: • Level 0: log nothing but serious errors • Level 1: also log RD generation (default) • Level 2: also log retrieval activity • Level 3: also log filtering activity • Level 4: also log spawning activity • Level 5: also log retrieval progress The default value is 1.	loglevel=[0...100]
max-connections	Specifies the maximum number of concurrent retrievals that a robot can make. The default is 8.	max-connections=[1..100]
max-filesize-kb	Specifies the maximum file size in kilobytes for files retrieved by the robot.	max-filesize-kb=1024

Table 47-1 User Modifiable Parameters in robot.conf File *(Continued)*

Parameter	Description	Example
max-memory-per-url / max-memory	Specifies the maximum memory in bytes used by each URL. If the URL needs more memory, the RD is saved to disk. The default is 1. This parameter cannot be set interactively through the administration console.	max-memory-per-url=n_bytes
max-working	Specifies the size of the robot working set, which is the maximum number of URLs the robot can work on at one time. This parameter cannot be set interactively through the administration console.	max-working=1024
onCompletion	Determines what the robot does after it has completed a run. The robot can either go into idle mode, loop back and start again, or quit. The default is idle. This parameter works with the cmd-hook parameter. When the robot is done, it will do the action of onCompletion and then run the cmd-hook program.	OnCompletion=[idle | loop | quit]
password	Specifies the password is used for httpd authentication and ftp connection.	password=string
referer	Specifies the parameter sent in the HTTP request if it is set to identify the robot as the referer when accessing Web pages	referer=string
register-user and register-password	Specifies the user name used to register RDs to the Search Engine database. This parameter cannot be set interactively through the Search Engine Administration Interface.	register-user=string
register-password	Specifies the password used to register RDs to the Search Engine database. This parameter cannot be set interactively through the administration console.	register-password=string
remote-access	This parameter determines if the robot can accept commands from remote hosts. The default is false.	remote-access=[true | false | yes | no]
robot-state-dir	Specifies the directory where the robot saves its state. In this working directory, the robot can record the number of collected RDs and so on.	robot-state-dir="/var/opt/SUNWps/instance/portal/robot”

Table 47-1 User Modifiable Parameters in robot.conf File *(Continued)*

Parameter	Description	Example
server-delay	Specifies the time period between two visits to the same web site, thus preventing the robot from accessing the same site too frequently.	server-delay=delay_in_seconds
site-max-connections	Indicates the maximum number of concurrent connections that a robot can make to any one site. The default is 2.	site-max-connections=[1..100]
smart-host-heuristics	Enables the robot to change sites that are rotating their DNS canonical host names. For example, www123.siroe.com is changed to www.siroe.com. The default is false.	smart-host-heuristics=[true | false]
tmpdir	Specifies a place for the robot to create temporary files. Use this value to set the environment variable TMPDIR.	tmpdir=path
user-agent	Specifies the parameter sent with the email address in the http-request to the server.	user-agent=iPlanetRobot/4.0
username	Specifies the user name of the user who runs the robot and is used for httpd authentication and ftp connection. The default is anonymous.	username=string

The most important parameters are `enumeration-filter` and `generation-filter`, which determine the filters the robot uses for enumeration and generation. The default values for these are enumeration-default and generation-default, which are the names of the filters provided by default in the `filter.conf` file.

All filters must be defined in the file `filter.conf` file. If you define your own filters in `filter.conf` file, you must add any necessary parameters to `robot.conf` file.

For example, if you define a new enumeration filter named my-enumerator, you would add the following parameter to `robot.conf` file:

```
enumeration-filter=my-enumerator
```

Chapter 48

The Filtering Process

This chapter contains the following sections

Overview

The robot uses filters to determine which resources to process and how to process them. When the robot discovers references to resources as well as the resources themselves, it applies filters to each resource in order to enumerate it and to determine whether or not to generate a resource description to store in the Search Engine database.

The robot examines one or more seed URLs, applies the filters, and then applies the filters to the URLs spawned by enumerating the seed URLs, and so on. The seed URLs are defined in the `filterrules.conf` file.

A filter performs any required initialization operations and applies comparison tests to the current resource. The goal of each test is to either allow or deny the resource. A filter also has a shutdown phase during which it performs any required cleanup operations.

If a resource is allowed, that means that it is allowed to continue passage through the filter. If a resource is denied, then the resource is rejected. No further action is taken by the filter for resources that are denied. If a resource is not denied, the robot will eventually enumerate it, attempting to discover further resources. The generator might also create a resource description for it.

These operations are not necessarily linked. Some resources result in enumeration; others result in RD generation. Many resources result in both enumeration and RD generation. For example, if the resource is an FTP directory, the resource typically will not have an RD generated for it. However, the robot might enumerate the individual files in the FTP directory. An HTML document that contains links to other documents can receive an RD and can lead to enumeration of the linked documents as well.

Stages in the Filter Process

Both enumerator and generator filters have five phases in the filtering process. They both have four common phases: Setup, Metadata, Data, and Shutdown. If the resource makes it past the Data phase, it is either in the Enumerate or Generate phase, depending on whether the filter is an enumerator or a generator.

The phases are as follows:

Setup Performs initialization operations. Occurs only once in the life of the robot.

Metadata Filters the resource based on metadata that is available about the resource. Metadata filtering occurs once per resource before the resource is retrieved over the network. Table 48-1 lists examples of common metadata types. The table contains three columns. The first column lists the metadata type, the second column provides a description, and the third column provides an example.

Table 48-1 Common Metadata Types

Metadata	Description	Example
Complete URL	The location of a resource	http://home.siroe.com/
Protocol	The access portion of the URL	http, ftp, file
Host	The address portion of the URL	www.siroe.com
IP address	Numeric version of the host	198.95.249.6
PATH	The path portion of the URL	/index.html
Depth	Number of links from the seed URL	5

Data Filters the resource based on its data. Data filtering is done once per resource after it is retrieved over the network. Data that can be used for filtering include:

- content-type
- content-length
- content-encoding
- content-charset
- last-modified
- expires

Enumerate Enumerates the current resource in order to determine if it points to other resources to be examined.

Generate Generates a resource description (RD) for the resource and saves it in the Search Engine database.

Shutdown Performs any needed termination operations. Occurs once in the life of the robot.

Filter Syntax

The `filter.conf` file contains definitions for enumeration and generation filters. This file can contain multiple filters for both enumeration and generation. Note that the robot can determine which filters to use because they are specified by the `enumeration-filter` and `generation-filter` parameters in the `robot.conf` file.

Filter definitions have a well-defined structure: a header, a body, and an end. The header identifies the beginning of the filter and declares its name; for example:

```
<Filter name="myFilter">
```

The body consists of a series of filter directives that define the filter's behavior during setup, testing, enumeration or generation, and shutdown. Each directive specifies a function, and if applicable, parameters for the function.

The end is marked by `</Filter>`.

Code Example 48-1 on page 362 shows a filter named enumeration1.

Code Example 48-1 Enumeration File Syntax

```
<Filter name="enumeration1>
Setup fn=filterrules-setup config=./config/filterrules.conf
# Process the rules
MetaData fn=filterrules-process
# Filter by type and process rules again
Data fn=assign-source dst=type src=content-type
Data fn=filterrules-process
# Perform the enumeration on HTML only
Enumerate enable=true fn=enumerate-urls max=1024 type=text/html
# Cleanup
Shutdown fn=filterrules-shutdown
</Filter>
```

Filter Directives

Filter directives use Robot Application Functions (RAFs) to perform operations. Their use and flow of execution is similar to that of NSAPI directives and Server Application Functions (SAFs) in the file obj.conf. Like NSAPI and SAF, data are stored and transferred using parameter blocks, also called pblocks.

There are six robot directives, or RAF classes, corresponding to the filtering phases and operations listed below. See "Stages in the Filter Process" on page 360 for more information on these phases.

- Setup
- Metadata
- Data
- Enumerate
- Generate
- Shutdown

Each directive has its own robot application functions. For example, use filtering functions with the Metadata and Data directives, enumeration functions with the Enumerate directive, generation functions with the Generate directive, and so on.

The built-in robot application functions and instructions for writing your own robot application functions are explained in the *Portal Server Developer's Guide*.

Writing or Modifying a Filter

In most cases, you should not need to write filters from scratch. You can create most of your filters using the administration console. You can then modify the filter.conf and filterrules.conf files to make any desired changes. These files reside in the directory `/var/opt/SUNWps/http-`*hostname-domain*`/portal`.

However, if you want to create a more complex set of parameters, you will need to edit the configuration files used by the robot.

Note the following points when writing or modifying a filter:

- The order of execution of directives (especially the available information at each phase)
- The order of rules

For a discussion of the parameters you can modify in the `robot.conf` file, the robot application functions that you can use in the `filter.conf` file, and how to create your own robot application functions, see the *Portal Server Developer's Guide*.

Chapter 49

Robot Application Functions - Sources and Destinations

This chapter contains the following sections:

Introduction

Most of the Robot Application Functions (RAFs) require sources of information and generate data that goes to destinations. The sources are defined within the robot itself and are not necessarily related to the fields in the resource description it ultimately generates. Destinations, on the other hand, are generally the names of fields in the resource description, as defined by the resource description server's schema.

For details on using the administration console to determine the database schema, see *Sun Java System Portal Server Administration Guide*.

The following sections describe the different stages of the filtering process, and the sources available at those stages.

Setup Stage

At the Setup stage, the filter is set up and cannot yet get information about the resource's URL or content.

MetaData Filtering Stage

At the MetaData stage, the robot encounters a URL for a resource, but it has not downloaded the resource's content, thus information is available about the URL as well as data that is derived from other sources such as the `filter.conf` file. At this stage, however, information is not available about the content of the resource.

Table 49-1 lists the sources available to the RAFs at the MetaData phase. The table contains three columns. The first column lists the source, the second column provides a description, and the third column provides an example.

Table 49-1 Sources Available to the RAFs at the MetaData Phase

Source	Description	Example
csid	Catalog Server ID	x-catalog//budgie.siroe.com:8086/alexandria
depth	Number of links traversed from starting point	10
enumeration filter	Name of Enumeration filter	enumeration1
generation filter	Name of Generation filter	generation1
host	Host portion of URL	home.siroe.com
IP	Numeric version of host	198.95.249.6
protocol	Access portion of the URL	http, https, ftp, file
path	Path portion of the URL	/, /index.html, /documents/listing.html
URL	Complete URL	http://developer.siroe.com/docs/manuals/

Data Stage

At the Data stage, the robot has downloaded the content of the resource at the URL, and can access data about the content, such as the description, the author, and so on.

If the resource is an HTML file, the Robot parses the <META> tags in the HTML headers. Consequently, any data contained in <META> tags is available at the Data stage.

During the data phase, the following sources (shown in Table 49-2) are available to RAFs, in addition to those available during the MetaData phase. The table contains three columns. The first column lists the source, the second column provides a description, and the third column provides an example.

Table 49-2 Sources Available to the RAFs at the Data Phase

Source	Description	Example
content-charset	Character set used by the resource	
content-encoding	Any form of encoding	
content-length	Size of the resource in bytes	
content-type	MIME type of the resource	text/html, image/jpeg
expires	Date the resource itself expires	
last-modified	Date the resource was last modified	
data in <META> tags	Any data that is provided in <META> tags in the header of HTML resources	Author Description Keywords

Enumeration, Generation, and Shutdown Stages

At the Enumeration and Generation stages, the same data sources are available as the Data stage.

At the Shutdown stage, the filter completes its filtering and is shuts down. Although functions written for this stage can use the same data sources as those available at the Data stage, the shutdown functions typically restrict their operations to shutdown and cleanup activities.

Robot Application Functions - Enable Parameter

Each function can have an `enable` parameter. The values can be `true`, `false`, `on`, or `off`. The administration console uses these parameters to turn certain directives on or off.

The following example enables enumeration for text/html and disables enumeration for text/plain:

```
# Perform the enumeration on HTML only
Enumerate enable=true fn=enumerate-urls max=1024 type=text/html
Enumerate enable=false fn=enumerate-urls-from-text max=1024 type=text/plain
```

Adding an `enable=false` parameter or an `enable=off` parameter has the same effect as commenting the line. Because the administration console does not write comments, it writes an `enable` parameter instead.

Chapter 51

Robot Application Functions - Setup Functions

This section describes the functions that are used during the setup phase by both enumeration and generation filters. The following functions are described:

- filterrules-setup
- setup-regex-cache
- setup-type-by-extension

filterrules-setup

When you use the `filterrules-setup` function, `logtype` is the type of log file to use. The value can be `verbose`, `normal`, or `terse`.

Parameters

The following table lists the parameter used with the `filterrules-setup` function. The table contains two columns. The first column lists the parameter, and the second column provides a description.

`config`	Path name to the file containing the filter rules to be used by this filter.

Example

```
Setup fn=filterrules-setup config=./config/filterrules.conf logtype=normal
```

setup-regex-cache

The `setup-regex-cache` function initializes the cache size for the `filter-by-regex` and `generate-by-regex` functions. Use this function to specify a number other than the default of 32.

Parameters

The following table lists the parameter used with the `setup-regex-cache` function. The table contains three columns. The first column lists the parameter, the second column provides a description, and the third column provides an example.

`cache-size`	Maximum number of compiled regular expressions to be kept in the regex cache.

Example

```
Setup fn=setup-regex-cache cache-size=28
```

setup-type-by-extension

The `setup-type-by-extension` function configures the filter to recognize file name extensions. It must be called before the `assign-type-by-extension` function can be used. The file specified as a parameter must contain mappings between standard MIME content types and file extension strings.

Parameters

The following table lists the parameter used with the `setup-type-by-extension` function. The table contains two columns. The first column lists the parameter, and the second column provides a description.

`file`	Name of the MIME types configuration file.

Example

```
Setup fn=setup-type-by-extension file=./config/mime.types
```

Chapter 52

Robot Application Functions - Filtering Functions

This chapter contains the following sections:

Introduction

The functions discussed in this chapter operate at the Metadata and Data stages to allow or deny resources based on specific criteria specified by the function and its parameters.

These functions can be used in both Enumeration and Generation filters in the `filter.conf` file.

Each "`filter-by`" function performs a comparison, then either allows or denies the resource. Allowing the resource means that processing continues to the next filtering step. Denying the resource means that processing should stop, because the resource does not meet the criteria for further enumeration or generation.

filter-by-exact

The filter-by-exact function allows or denies the resource if the allow / deny string matches the source of information exactly. The keyword all matches any string.

Parameters

The following table lists the parameters used with the `filter-by-exact` function. The table contains two columns. The first column lists the parameter, and the second column provides a description.

`src`	Source of information.
`allow/deny`	Contains a string.

Example

The following example filters out all resources whose content-type is text / plain. It allows all other resources to proceed:

```
Data fn=filter-by-exact src=type deny=text/plain
```

filter-by-max

The `filter-by-max` function allows the resource if the specified information source is less than or equal to the given value. It denies the resource if the information source is greater than the specified value.

This function can be called no more than once per filter.

Parameters

The following table lists the parameters used with the `filter-by-max` function. The table contains two columns. The first column lists the parameter, and the second column provides a description.

`src`	Source of information. It must be one of the following: hosts, objects, or depth.

`value`	Specifies a value for comparison.

Example

This example allows resources whose content-length is less than 1024 K:

```
MetaData fn-filter-by-max src=content-length value=1024
```

filter-by-md5

The `filter-by-md5` function only allows the first resource with a given MD5 checksum value. If the current resource's MD5 has been seen in an earlier resource by this robot, the current resource is denied. As a result, duplication of identical resources or single resources with multiple URLs is prevented.

You can only call this function at the Data stage or later. It can be called no more than once per filter. The filter must invoke the `generate-md5` function to generate an MD5 checksum before invoking `filter-by-md5` function.

Parameters

none

Example

The following example shows the typical method of handling MD5 checksums by first generating the checksum and then filtering based on it:

```
Data fn=generate-md5
```

```
Data fn=filter-by-md5
```

filter-by-prefix

The `filter-by-prefix` function allows or denies the resource if the given information source begins with the specified prefix string. The resource doesn't have to match completely. The keyword all matches any string.

Parameters

The following table lists the parameters used with the `filter-by-prefix` function. The table contains two columns. The first column lists the parameter, and the second column provides a description.

`src`	Source of information.
`allow/deny`	Contains a string for prefix comparison.

Example

The following example allows resources whose content-type is any kind of text, including text / html and text / plain:

```
MetaData fn=filter-by-prefix src=type allow=text
```

filter-by-regex

The `filter-by-regex` function supports regular expression pattern matching. It allows resources that match the given regular expression. The supported regular expression syntax is defined by the POSIX.1 specification. The regular expression * matches anything.

Parameters

The following table lists the parameters used with the `filter-by-regex` function. The table contains two columns. The first column lists the parameter, and the second column provides a description.

`src`	Source of information.
`allow/deny`	Contains a string for prefix comparison.

Example

The following example denies all resources from sites in the government domain:

```
MetaData fn=filter-by-regex src=host deny=\\*.gov
```

filterrules-process

The `filterrules-process` function handles in the rules in the `filterrules.conf` file.

Parameters

none

Example

```
MetaData fn=filterrules-process
```

Chapter 53

Robot Application Functions - Filtering Support Functions

This chapter contains the following sections:

Introduction

The functions discussed in this chapter are used during filtering to manipulate or generate information on the resource. The robot can then process the resource by calling filtering functions. These functions can be used in Enumeration and Generation filters in the `filter.conf` file.

assign-source

The assign-source function assigns a new value to a given information source. This permits editing during the filtering process. The function can assign an explicit new value, or it can copy a value from another information source.

Parameters

The following table lists the parameters used with the assign-source function. The table contains two columns. The first column lists the parameter, and the second column provides a description.

dst	Name of the source whose value is to be changed.
value	Specifies an explicit value.
src	Information source to copy to dst

You must specify either a value parameter or a src parameter, but not both.

Example

```
Data fn=assign-source dst=type src=content-type
```

assign-type-by-extension

The assign-type-by-extension function uses the resource's file name to determine its type and assigns this type to the resource for further processing.

The setup-type-by-extension function must be called during setup before assign-type-by-extension function can be used.

Parameters

The following table lists the parameter used with the assign-type-by-extension function. The table contains two columns. The first column lists the parameter, and the second column provides a description.

src	Source of file name to compare. If you do not specify a source, the default is the resource's path.

Example

```
MetaData fn=assign-type-by-extclear-source
```

clear-source

The clear-source function deletes the specified data source. You typically do not need to perform this function. You can create or replace a source by using the assign-source function.

Parameters

The following table lists the parameter used with the clear-source function. The table contains two columns. The first column lists the parameter, and the second column provides a description.

src	Name of source to delete.

Example

The following example deletes the path source:

```
MetaData fn=clear-source src=path
```

convert-to-html

The convert-to-html function converts the current resource into an HTML file for further processing, if its type matches a specified MIME type. The conversion filter automatically detects the type of the file it is converting.

Parameters

The following table lists the parameter used with the `convert-to-html` function. The table contains two columns. The first column lists the parameter, and the second column provides a description.

`type`	MIME type from which to convert.

Example

The following sequence of function calls causes the filter to convert all Adobe Acrobat PDF files, Microsoft RTF files, and FrameMaker MIF files to HTML, as well as any files whose type was not specified by the server that delivered it.

```
Data fn=convert-to-html type=application/pdf
Data fn=convert-to-html type=application/rtf
Data fn=convert-to-html type=application/x-mif
Data fn=convert-to-html type=unknown
```

copy-attribute

The `copy-attribute` function copies the value from one field in the resource description into another.

Parameters

The following table lists the parameters used with the `copy-attribute` function. The table contains two columns. The first column lists the parameter, and the second column provides a description.

`src`	Field in the resource description from which to copy.
`dst`	Item in the resource description into which to copy the source.

truncate	Maximum length of the source to copy.
clean	Boolean parameter indicating whether to fix truncated text (such as not leaving partial words). This parameter is false by default.

Example

```
Generate fn=copy-attribute \
src=partial-text dst=description truncate=200 clean=true
```

generate-by-exact

The generate-by-exact function generates a source with a specified value, but only if an existing source exactly matches another value.

Parameters

The following table lists the parameters used with the generate-by-exact function. The table contains two columns. The first column lists the parameter, and the second column provides a description.

dst	Name of source to generate.
value	Value to assign dst.
src	Source against which to match.

Example

The following example sets the classification to Siroe if the host is www.siroe.com.

```
Generate fn="generate-by-exact" match="www.siroe.com:80" src="host"
value="Siroe" dst="classification"
```

generate-by-prefix

This `generate-by-prefix` function generates a source with a specified value, but only if the prefix of an existing source matches another value.

Parameters

The following table lists the parameters used with the `generate-by-prefix` function. The table contains two columns. The first column lists the parameter, and the second column provides a description.

`dst`	Name of the source to generate.
`value`	Value to assign to dst.
`src`	Source against which to match.
`match`	Value to compare to src.

Example

The following example sets the classification to Compass if the protocol prefix is HTTP:

```
Generate fn="generate-by-prefix" match="http" src="protocol" value="World
Wide Web" dst="classification"
```

generate-by-regex

The `generate-by-regex` function generates a source with a specified value, but only if an existing source matches a regular expression.

Parameters

The following table lists the parameters used with the `generate-by-regex` function. The table contains two columns. The first column lists the parameter, and the second column provides a description.

`dst`	Name of the source to generate.

value	Value to assign to dst.
src	Source against which to match.
match	Regular expression string to compare to src.

Example

The following example sets the classification to Siroe if the host name matches the regular expression *.siroe.com. For example, resources at both developer.siroe.com and home.siroe.com will be classified as Siroe:

```
Generate fn="generate-by-regex" match="\\*.siroe.com" src="host"
value="Siroe" dst="classification"
```

generate-md5

The generate-md5 function generates an MD5 checksum and adds it to the resource. You can then use the filter-by-md5 function to deny resources with duplicate MD5 checksums.

Parameters

none

Example

```
Data fn=generate-md5
```

generate-rd-expires

The generate-rd-expires function generates an expiration date and adds it to the specified source. The function uses metadata such as the HTTP header and HTML <META> tags to obtain any expiration data from the resource. If none exists, it generates an expiration date three months from the current date.

Parameters

The following table lists the parameter used with the `generate-rd-expires` function. The table contains two columns. The first column lists the parameter, and the second column provides a description.

`dst`	Name of the source. If you omit it, it defaults to rd-expires.

Example

```
Generate fn=generate-rd-expires
```

generate-rd-last-modified

The `generate-rd-last-modified` function adds the current time to the specified source.

Parameters

The following table lists the parameter used with the `generate-rd-last-modified` function. The table contains two columns. The first column lists the parameter, and the second column provides a description.

`dst`	Name of the source. If you omit it, it defaults to rd-last-modified.

Example

```
Generate fn=generate-last-modified
```

rename-attribute

The `rename-attribute` function changes the name of a field in the resource description. It is most useful in cases where, for example, `extract-html-meta` copies information from a `<META>` tag into a field, and you want to change the name of the field.

Parameters

The following table lists the parameter used with the `generate-rd-last-modified` function. The table contains two columns. The first column lists the parameter, and the second column provides a description.

`src`	String containing a mapping from one name to another.

Example

The following example renames an attribute from author to author-name:

```
Generate fn=rename-attribute src="author->author-name"
```

Chapter 54

Robot Application Functions - Enumeration Functions

This chapter contains the following sections

Introduction

The functions discussed in this chapter operate at the Enumerate stage. These functions control if and how a robot gathers links from a given resource in order to use as starting points for further resource discovery.

enumerate-urls

The `enumerate-urls` function scans the resource and enumerates all URLs found in hypertext links. The results are used to spawn further resource discovery. You can specify a content-type to restrict the kind of URLs enumerated.

Parameters

The following table lists the parameters used with the `enumerate-urls` function. The table contains two columns. The first column lists the parameter, and the second column provides a description.

max	The maximum number of URLs to spawn from a given resource. The default, if max is omitted, is 1024.
type	Content-type that restricts enumeration to those URLs that have the specified content-type. type is an optional parameter. If omitted, it will enumerate all URLs.

Example

The following example enumerates HTML URLs only, up to a maximum of 1024:

```
Enumerate fn=enumerate-urls type=text/html
```

enumerate-urls-from-text

The enumerate-urls-from-text function scans text resources, looking for strings matching this regular expression: URL:.*. It spawns robots to enumerate the URLs from these strings and generate further resource descriptions.

Parameters

The following table lists the parameter used with the enumerate-urls-from-text function. The table contains two columns. The first column lists the parameter, and the second column provides a description.

max	The maximum number of URLs to spawn from a given resource. The default, if max is omitted, is 1024.

Example

```
Enumerate fn=enumerate-urls-from-text
```

Chapter 55

Robot Application Functions - Generation Functions

This chapter contains the following functions:

Introduction

The following functions are used in the Generate stage of filtering. Generation functions can generate information that goes into a resource description. In general, they either extract information from the body of the resource itself or copy information from the resource's metadata.

extract-full-text

The `extract-full-text` function extracts the complete text of the resource and adds it to the resource description.

NOTE The extract-full-text function should be used with caution, because it can significantly increase the size of the resource description, thus causing database bloat and overall negative impact on network bandwidth.

Parameters

The following table lists the parameters used with the `extract-full-text` function. The table contains two columns. The first column lists the parameter, and the second column provides a description.

`truncate`	The maximum number of characters to extract from the resource.
`dst`	Name of the schema item that will receive the full text.

Example

```
Generate fn=extract-full-text
```

extract-html-meta

The `extract-html-meta` function extracts any `<META>` or `<TITLE>` information from an HTML file and adds it to the resource description. A content-type may be specified to restrict the kind of URLs that are generated.

Parameters

The following table lists the parameters used with the `extract-html-meta` function. The table contains two columns. The first column lists the parameter, and the second column provides a description.

`truncate`	The maximum number of bytes to extract.
`type`	Optional parameter. If omitted, it will generate all URLs.

Example

```
Generate fn=extract-html-meta truncate=255 type=text/html
```

extract-html-text

The `extract-html-text` function extracts the first few characters of text from an HTML file, excluding the HTML tags, and adds the text to the resource description. This permits the first part of a document's text to be included in the RD. A content-type may be specified to restrict the kind of URLs that are generated.

Parameters

The following table lists the parameters usedwith the `extract-html-text` function. The table contains two columns. The first column lists the parameter, and the second column provides a description.

`truncate`	The maximum number of bytes to extract.
`skip-headings`	Set to true to ignore any HTML headers that occur in the document.
`type`	Optional parameter. If omitted, it will generate all URLs.

Example

```
Generate fn=extract-html-text truncate=255 type=text/html
skip-headings=true
```

extract-html-toc

The `extract-html-toc` function extracts the table-of-contents from the HTML headers and add it to the resource description.

Parameters

The following table lists the parameters used with the `extract-html-toc` function. The table contains two columns. The first column lists the parameter, and the second column provides a description.

`truncate`	The maximum number of bytes to extract.

`level`	Maximum HTML header level to extract. This parameter controls the depth of the table of contents.

Example

```
Generate fn=extract-html-toc truncate=255 level=3
```

extract-source

The `extract-source` function extracts the specified values from the given sources and adds them to the resource description.

Parameters

The following table lists the parameter used with the `extract-source` function. The table contains two columns. The first column lists the parameter, and the second column provides a description.

`src`	List of source names; you can use the -> operator to define a new name for the RD attribute, for example, type->content-type would take the value of the source named type and save it in the RD under the attribute named content-type.

Example

```
Generate fn=extract-source src="md5,depth,rd-expires,rd-last-modified"
```

harvest-summarizer

The `harvest-summarizer` function runs a Harvest summarizer on the resource and adds the result to the resource description.

To run Harvest summarizers, you must have $HARVEST_HOME/lib/gatherer in your path before you run the robot.

Parameters

The following table lists the parameter used with the `harvest-summarizer` function. The table contains two columns. The first column lists the parameter, and the second column provides a description.

`summarizer`	Name of the summarizer program.

Example

```
Generate fn-harvest-summarizer summarizer=HTML.sum
```

Chapter 56

Robot Application Functions - Shutdown Functions

This chapter contains the following sections:

Introduction

The function in this chapter can be used during the shutdown phase by both enumeration and generation functions.

filterrules-shutdown

After the rules are run, the `filterrules-shutdown` function performs clean up and shutdown responsibilities.

Parameters

none

Example

```
Shutdown fn=filterrules-shutdown
```

Part IX

Desktop Tag Library

Chapter 58

Overview

This chapter contains the following sections:

Introduction

Sun Java System Portal Server uses two types of tags: JSP tags and Desktop template tags. JSP tags are used in the JavaServer Pages™ (JSPs™) and template tags are used in the HTML pages of the Desktop.

Types of Tags

The 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.

Desktop Template Tags

This section describes 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 Templates

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/imag es/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`

JavaServer Pages Tags

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.

Purpose of Tag Library

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 tags in the JSP 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).

Validator tags These tags validate that the provider in context can legally use the tags in the TLD.

Normal tags These tags serve as wrappers of PAPI, ProviderContext, ContainerProviderContext, and the specific containers in the Sun Java System Portal Server software.

Tag Attributes, Return Values, and Exceptions

Attributes

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).

Return Values

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.

Exceptions

The tag library provides five types of exceptions.

Invalid Tag Sequence Occurs when obtain tags are nested in an invalid sequence. See Using the Desktop Tag Library in Your Application for the sequencing of the obtain tags. The name of the first tag that violates the sequence is provided.

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.

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.

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.

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.

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.

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.

Types of Tags

The `search.tld` file 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.

Tag Attributes

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.

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

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`.

Tag Library Descriptors

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.

Chapter 59

Context Setup Tags

The context setup 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 59-1 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 59-1 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		desktopContainerProviderContext.tld	channel (required) - the name of the channel
		desktopSingle.tld	none
obtainSelectedChannel			
obtainTab		desktopTab.tld	tab (required) - the name of the tab

Table 59-1 Context Setup Tags *(Continued)*

Tag Name	Description	in TLD File	Attributes/Descriptions
obtainTabByName	Gets the tab name.	desktopTab.tld	name (required) - the name of the tab

Chapter 60

Validator Tags

The validator 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`.

Chapter 61

Normal Tags in desktop.tld

All of the normal tags in the `desktop.tld` file have a bodycontent of empty.

Table 61-1 desktop.tld Normal Tags with Attributes

Tag Name	Description	Attributes
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)
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)

Table 61-1 desktop.tld Normal Tags with Attributes *(Continued)*

Tag Name	Description	Attributes
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)
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)

Table 61-1 desktop.tld Normal Tags with Attributes *(Continued)*

Tag Name	Description	Attributes
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

Chapter 62

Normal Tags in desktopProviderContext.tld

All of the normal tags in the `desktopProviderContext.tld` file have a bodycontent of empty.

Table 62-1 desktopProviderContext.tld Normal Tags with Attributes

Tag Name	Description	Attributes
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)

Table 62-1 desktopProviderContext.tld Normal Tags with Attributes *(Continued)*

Tag Name	Description	Attributes
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)
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)

Table 62-1 desktopProviderContext.tld Normal Tags with Attributes *(Continued)*

Tag Name	Description	Attributes
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 (optional)
		scope (optional)
getClientPropertiesFilters		id (optional)
		scope (optional)
getClientAndLocalePropertiesFilters		id (optional)
		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)
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)

Table 62-1 desktopProviderContext.tld Normal Tags with Attributes *(Continued)*

Tag Name	Description	Attributes
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
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)

Table 62-1 desktopProviderContext.tld Normal Tags with Attributes *(Continued)*

Tag Name	Description	Attributes
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)
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)

Table 62-1 desktopProviderContext.tld Normal Tags with Attributes *(Continued)*

Tag Name	Description	Attributes
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)

Chapter 63

Normal Tags in desktopContainerProviderContext.tld

All of the normal tags in the `desktopProviderContext.tld` file have a bodycontent of empty.

Table 63-1 desktopContainerProviderContext.tld Normal Tags with Attributes

Tag Name	Description	Attributes
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)

Chapter 64

Normal Tags in desktopTab.tld

All of the normal tags in the `desktopTab.tld` file have a bodycontent of empty.

Table 64-1 desktopTab.tld Normal Tags with Attributes

Tag Name	Description	Attributes
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 64-1 desktopTab.tld Normal Tags with Attributes *(Continued)*

Tag Name	Description	Attributes
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)

Normal Tags in desktopTable.tld

All of the normal tags in the `desktopTable.tld` file have a bodycontent of empty.

Table 65-1 desktopTable.tld Normal Tags with Attributes

Tag Name	Description	Attributes
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 (required)
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)
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 (optional) scope (optional)

Table 65-1 desktopTable.tld Normal Tags with Attributes *(Continued)*

Tag Name	Description	Attributes
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)

Normal Tags in desktopTheme.tld

All of the normal tags in the `desktopTheme.tld` file have a bodycontent of empty.

Table 66-1 desktopTheme.tld Normal Tags with Attributes

Tag Name	Description	Attributes
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)
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 (required) 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

Table 66-1 desktopTheme.tld Normal Tags with Attributes *(Continued)*

Tag Name	Description	Attributes
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.

Table 66-1 desktopTheme.tld Normal Tags with Attributes *(Continued)*

Tag Name	Description	Attributes
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

Chapter 67

Normal Tags for searchContext

The following table describes the normal tags for the searchContext Java API in the Search tag library.

Table 67-1 search.tld Normal Tags with Attributes for the searchContext

Tag Name	Description	Attributes
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

Chapter 68

Pre-Search Tags in search.tld

The tags in the following table are normally used before executing a search. Before a successful search can be executed these tags must set a value for:

- setRDMServer
- setRDMType
- setQuery or setCriteria

Table 68-1 search.tld Tags for Pre-Search

Tag Name	Description	Attributes
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.
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

Table 68-1 search.tld Tags for Pre-Search *(Continued)*

Tag Name	Description	Attributes
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
setQuery	Query string. Either this value has to be set or else setCriteria has to be set.	query (required) - string - dynamic attribute

Table 68-1 search.tld Tags for Pre-Search *(Continued)*

Tag Name	Description	Attributes
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

Execute Search tag in search.tld

The `search.tld` includes the tag that executes the search.

Table 69-1 desktop.tld Normal Tags with Attributes

Tag Name	Description	Attributes
executeSearch	Tag for the execute method in SearchContext API. Executes search after doing some validation of the search parameters and query string.	none

Post Search Tags in search.tld

The tags in the following table are related to search results and are used after a search is executed. They provide various counts and help display the search results.

Table 70-1 desktop.tld Normal Tags with Attributes

Tag Name	Description	Attributes
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)

Table 70-1 desktop.tld Normal Tags with Attributes *(Continued)*

Tag Name	Description	Attributes
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)
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)

Miscellaneous Tags in search.tld

All of the following normal tags in the `search.tld` file have a bodycontent of empty.

Table 71-1 search.tld Normal Tags with Attributes

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)

Chapter 72

Desktop Template Common Tags

This chapter contains a listing of all the common tags in the Desktop template files.

[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
[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
[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
[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&thecontainername.channelAction=remove&thecontainername.targetProvider=providername
[tag:url]	URL of new location
[tag:editURL]	URL of Edit page for this channel
	?action=edit&provider=theeditcontainername&targetprovider=providername&containerName=thecontainername
[tag:detachAttachURL]	URL of the channel to be detached
	action=process&provider=thecontainername&thecontainername.channelAction=attach&thecontainername.targetProvider=providename
[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&thecontainername.channelAction=maxmize&thecontainername.targetProvider=providername
	?action=process&provider=thecontainername&thecontainername.channelAction=minimize&thecontainername.targetProvider=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
[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

Chapter 73

Provider-Specific Desktop Template Tags

This chapter contains the following sections:

- AddressBookProvider, CalendarProvider, MailCheckProvider, and MailProvider
- AppProvider, LoginProvider, and UserInfoProvider
- TemplateTabContainerProvider and TemplateTableContainerProvider

AddressBookProvider

The tags described below are used by AddressBookProvider and providers who extend this provider (such as LotusNotesAddressBookProvider and MSExchangeAddressBookProvider).

`[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.

[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 73-1

[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

Table 73-1

[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
[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

Table 73-1

[tag:calendar-display-dayVie w-overdueTask-description]	
[tag:calendar-display-dayVie w-task-d ueHour1]	hour of task for 12-hour clock
[tag:calendar-display-dayVie w-task-d ueMonth]	month the task was due
[tag:calendar-display-dayVie w-task-dueDay]	day of the month the task was due
[tag:calendar-display-dayVie w-task-dueYear]	year the task was due
[tag:calendar-display-dayVie w-overdueTask-summary]	overdue task summary
[tag:calendar-display-dayVie w-task-pendMinute]	minutes of the task
[tag:calendar-display-dayVie w-task-location]	task location
[tag:calendar-display-dayVie w-task-complete-start]	start time of completed task
[tag:calendar-display-dayVie w-dayOfWeek]	today’s day of week
[tag:calendar-display-dayVie w-month]	today’s month
[tag:calendar-display-dayVie w-day]	today’s day in month
[tag:calendar-display-dayVie w-year]	today’s year
[tag:calendar-display-dayVie w-taskList]	task list content
[tag:calendar-display-dayVie w-eventList]	event list content
[tag:calendar-display-dayVie w-otherTaskList]	other tasks list content
[tag:calendar-display-dayVie w-overdueTaskNum]	number of overdue tasks for today

Table 73-1

[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-Hea der]	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
[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

Table 73-1

[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
[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

Table 73-1

[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
[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

Table 73-1

[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
[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

Table 73-1

[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
[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

Table 73-1

[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
[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

Table 73-1

[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
[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.

[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?metaAli as=www.siroe.com&goto=http://www.siroe.com:80/por tal/dt

MailCheckProvider

The tags described below are used by 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.
[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).

[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
[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.

[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
[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.

[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
[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.

[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
[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

Chapter 74

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 74-1 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)

Table 74-1 Tags in im.tld *(Continued)*

Tag Name	Description	Attributes
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)
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)

NOTE 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.

Part X

Desktop Templates

Chapter 75

Overview

This chapter contains the following sections:

Introduction

To generate the rendered Desktop user interface (what the industry refers to as the "presentation"), the Sun Java System Portal Server software makes use of either JavaServer Pages (JSP™) or template files.

Installation Location

The default set of template files are installed in `/etc/opt/SUNWps/desktop/default` directory. The sample portal template files are installed in `/etc/opt/SUNWps/desktop/sampleportal` and `/etc/opt/SUNWps/desktop/anonymous` directories. Files in the `/etc/opt/SUNWps/desktop/anonymous` directory are specific to the sample portal Anonymous Desktop.

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.

File Lookup Scenario

The Portal Server software uses the lookup scenario outlined in Code Example 75-1 and Code Example 75-2 on page 477 to find the JSPs it needs. Use this order to decide the final location of your own JSPs.

Code Example 75-1 Template File Lookup Scenario

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*

Code Example 75-1 Template File Lookup Scenario *(Continued)*

```
default/channelname
default/provider
default/clientPath
default
templateroot
```

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

Code Example 75-2 Template File Lookup Scenario

```
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:

`desktoptype`	For example `default` (set in the 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 `desktoptype` attribute.
`locale`	Preferred `locale` is the user's locale. For example, `en_US` (set by users through the administration console in the "User" setting)
`clientPath`	This is an optional file-path containing client-specific templates; for example, `html` (set through the administration console Client Detection service)

`channelname`	This is the name of the channel; for example, newSingleContainer (set in the display profile)
`provider`	This is the provider name; for example, JSPSingleContainerProvider (set in the display profile)
`templateroot`	This is 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.

Chapter 76

Desktop Templates in the default Directory

This chapter contains the following sections

AddressBookProvider

The following table lists the templates in the `html` subdirectory of the AddressBookProvider, LotusNotesAddressBookProvider, and MSExchangeAddressBookProvider 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.

`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
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

The following table 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.

display.template	Contains the JavaScript code that launches the windows that the HTML applications show up in.

BookmarkProvider

The following table 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.

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

The following table lists the templates in the html subdirectory of the CalendarProvider, LotusNotesCalendarProvider, and MSExchangeCalendarProvider 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.

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
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
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

The following table 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.

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.
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

The following table 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.

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.

MailCheckProvider

The following table 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.

display.template	Contains MailCheck provider layout content.
edit.template	Contains the formatting for the Edit page for the MailCheck provider.

MailProvider

The following table lists the templates in the html subdirectory of the MailProvider, LotusNotesMailProvider, and 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.

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
edit-string.template	Used for creating edit page text boxes
url.template	Used for creating hyperlinks

TemplateTabContainerProvider

The following table 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.

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

The following table 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.

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.
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

The following table 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.

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

The following table 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.

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.
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.

default

Chapter 77

Desktop Templates in the sampleportal Directory

This chapter contains the following sections:

MyFrontPageTemplatePanelContainer

The following table 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.

`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

The following table 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.

`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.
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

The following table 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.

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 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.
selectAll.js	JavaScript code used on the Layout page.
switchColumns.js	JavaScript code used on the Layout page.

SamplesTemplatePanelContainer

The following table 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.

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

The following table 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.

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

The following table 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.

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

The following table 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.

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

The following table 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.

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.

sampleportal

Index

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

R

S

T

U

V

X