Tag Library for Delegated Administration

Sun Java™ System Portal Server 7 Technical Note

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

Part No: 819–5454
2006

060213@13996

Tag Library for Delegated Administration

This technical note describes the administration tag library that can be used to develop administration portlets for allowing administrators to administer portal through their portal desktop. Administration through portal desktop allow administrators to designate delegated administrators by providing access to administration tasks as portlets. A delegated administrator can be responsible for managing various tasks such as resetting passwords, creating/deleting users, tabs, and portlets, and arranging role based tabs, in a particular organization or a sub-organization. The tag library provides tags for achieving these tasks by writing portlets.

The administration tag library, associated with the portlets, can be used to modify out-of-the-box administration portlets or develop portlets with new administration functionality. The tag library supports user management, provider management, and portlet and WSRP management tasks. The tags in this library can be used for creating and administering channels based on JSPProvider. It is possible to write custom administration portlets with custom GUI using the tags in the tag library. The tag library can be used to write administrative portlets to administer any custom channel.

This technical note contain the following sections:

- "Technical Note Revision History" on page 3
- "Tags for Desktop Channel and Container Management Tasks" on page 4
- "Tags for Portlet Management Tasks" on page 13
- "Tags for User Management Tasks" on page 15
- "Tags for WSRP Management Tasks" on page 17
- "Accessing Sun Resources Online" on page 19

Technical Note Revision History

Version	Date	Description of Changes
10	January 2006	Final version.

Tags for Desktop Channel and Container Management Tasks

`obtainChannelAdmin`	All the channel administration tags must be nested inside this tag and will operate on the base distinguished node that is passed into this tag. For example: `<dtadmin:obtainChannelAdmin baseDN="dc=example,dc=siroe,dc=com"> ... <dtadmin:obtainChannelAdmin>` The `baseDN` is a required attribute and must be a valid node in the directory server. For example, `dc=red,dc=iplanet,dc=com`. Returns variables available inside the nesting with extra information for this tag: `CHANNEL_NAME_SEPARATOR`, `STRING_DP`, `INTEGER_DP`, `BOOLEAN_DP`, `COLLECTION_DP`, `UNKNOWN_DP`.
`getBaseDNs`	Gets the list of distinguished nodes the currently authenticated delegated administrator can administer. For example: `<dtadmin:getBaseDNs id="nodes"/>` . The `id` attribute is optional and takes the name of the exported scoped variable for the resulting value. The `scope` attribute is optional and takes the scope for `id`. Returns `java.util.Set`.
`getDNDisplayName`	Gets the display name for the distinguished name that is passed in. For example, `<dtadmin:getDNDisplayName id="name" dn="dc=red,dc=iplanet,dc=com"/>` . The `id` attribute is optional and takes the name of the exported scoped variable for the resulting value. The `scope` attribute is optional and takes the scope for `id`. The `dn` attribute is required and takes the distinguished name for which the display name is requested. Returns `java.lang.String`.
`getAssignableChannels`	Returns the set of channels or containers that are assignable to the available and selected list of a container. This tag returns the list of channels which includes the container's children and the container's parent's children, parent's parent's children, and traverses recursively until it reaches the display profile root. Mergers are taken into account when the channels are traversed. For example,

	`<dtadmin:getAssignableChannelsid="AssignableChannels" container="containerName"/>` The `id` attribute is optional and takes the name of the exported scoped variable for the resulting value. The `scope` attribute is optional and takes the scope for `id`. The `container` attribute is required and takes the name of container for which assignable channels are requested. Returns `java.util.Set`.
`getExistingChannels`	Returns the set of existing channels. The channels in this list can be modifiable or deleted by the user who can administer the base distinguished node that this tag is operating on. This tag returns the list of channels which includes the channels defined at the base distinguished node, channels defined inside the containers that are defined at the base distinguished node. If *all* is set to true, all the channels from the merged list is returned. This tag returns the set of channels that match the search string if *regExp* is provided. For example, `<dtadmin:getExistingChannels id="FilteredChannels" regExp="*" all="false"/>` The `id` attribute is optional and takes the name of the exported scoped variable for the resulting value. The `scope` attribute is optional and takes the scope for `id`. The `all` attribute is required and if true, takes mergers into account. The `regExp` attribute takes a search string (if filtering resulting channels based on a regular expression) specifically of the form `foo*`, `*foo`, `foo*bar`, `foo*bar*`. Returns `java.util.Set`.
`getExistingContainers`	Returns the set of existing containers. The containers in this list can be modifiable or deleted by the user who can administer the base distinguished node that this tag is operating on. This tag returns the list of containers which includes the containers defined at the base distinguished node, containers defined inside the containers that are defined at the base distinguished node. If *all* is set to true, all the containers from the merged list is returned. This tag returns the set of containers that match the search string if `regExp` is provided. For example, `<dtadmin:getExistingContainers id="ExistingContainers" all="true"/>`

	The `id` attribute is optional and takes the name of the exported scoped variable for the resulting value. The `scope` attribute is optional and takes the scope for `id`. The `all` attribute is required and if true, takes mergers into account. The `regExp` attribute takes a search string (if filtering resulting channels based on a regular expression) specifically of the form `foo*`, `*foo`, `foo*bar`, `foo*bar*`. Returns `java.util.Set`.
`getExistingProviders`	Returns the set of existing providers. The providers in this list can be used to create channels by the user that can administer the base distinguished node that this tag is operating on. Always takes mergers into account. For example, `<dtadmin:getExistingProviders id="ExistingProviders"/>` The `id` attribute is optional and takes the name of the exported scoped variable for the resulting value. The `scope` attribute is optional and takes the scope for `id`. Returns `java.util.Set`.
`getExistingContainerProviders`	Returns the set of existing container providers. The container providers in this list can be used to create containers by the user who can administer the base distinguished node that this tag is operating on. Always takes mergers into account. For example, `<dtadmin:getExistingContainerProviders id="ExistingContainerProviders"/>`. The `id` attribute is optional and takes the name of the exported scoped variable for the resulting value. The `scope` attribute is optional and takes the scope for `id`. Returns `java.util.Set`.
`createChannel`	Creates a new channel. A new channel is created based on the named provider. To create a nested channel, supply a hierarchical channel name. For example, to create channel A inside of container X, based on provider P: `<dtadmin:createChannel channelName="A/X" providerName="P"/>`. Use the `CHANNEL_NAME_SEPARATOR` variable for constructing the hierarchical channel names. The `channelName` attribute is required and takes the name of the channel. The `providerName` attribute is required and must contain the name of the base provider.

`createContainer`	Creates a new container. A new container is created based on the named container provider. To create a nested container, supply a hierarchical channel name. For example, to create container A inside of container X, based on provider P: `<dtadmin:createContainerl channelName="A/X" providerName="P"/>` The `channelName` attribute is required and takes the name of the channel. The `providerName` attribute is required and must contain the name of the base provider.
`deleteChannel`	Deletes a channel or container. To delete a nested channel or container, supply parent name. For example, to delete channel A inside of container X, at base distinguished node `dc=iplanet,dc=com,<dtadmin:deleteChannel node="dc=iplanet,dc=com" channelName="A" parentContainer="X"/>`. The `channelName` attribute is required and takes the name of the channel. The `parentContainer` attribute is required and takes the parent container name.
`getAvailableChannels`	Gets the list of available channels in a container. For example, `<dtadmin:getAvailableChannels id="AvailableChannels" container="containerName"/>` . The `id` attribute is optional and takes the name of the exported scoped variable for the resulting value. The `scope` attribute is optional and takes the scope for `id`. The `container` attribute is required and takes the name of the container for which assignable channels are requested. Returns `java.util.List`.
`getSelectedChannels`	Gets the list of selected channels in a container. For example, `<dtadmin:getSelectedChannels id="SelectedChannels" container="containerName"/>`. The `id` attribute is optional and takes the name of the exported scoped variable for the resulting value. The `scope` attribute is optional and takes the scope for `id`. The `container` attribute is required and takes the name of the container for which assignable channels are requested. Returns `java.util.List`.
`setAvailableChannels`	Sets the list of available channels in a container. For example, `<dtadmin:setAvailableChannels`

	`available="$available" container="JDCFrontPageTabPanel"/>`. The `container` attribute is required and takes the name of the container for which assignable channels are requested. The `available` attribute is required and takes the new available list.
`setSelectedChannels`	Sets the list of selected channels in a container. For example, `<dtadmin:setSelectedChannels selected="$selected" container="JDCFrontPageTabPanel"/>`. The `container` attribute is required and takes the name of the container for which assignable channels are requested. The `selected` attribute is required and takes the new selected list.
`getClassName`	Gets the fully classified class name for the provider class that this channel is based on. For example, `<dtadmin:getClassName id="classname" channel="channelName"/>`. The `id` attribute is optional and takes the name of the exported scoped variable for the resulting value. The `scope` attribute is optional and takes the scope for `id`. The `channel` attribute is required and takes the channel name for which the class is requested. Returns `java.lang.String`.
`getPropertyNames`	Gets the list of properties in a channel or container. This tag returns the list of basic property names at the given channel name that is passed in if *advanced* attribute is false or not set. If *advanced* attribute is set to true, it returns the advanced property names. If `regExp` is specified, returns the property names that match the search string. For example, `<dtadmin:getPropertyNames channel="SampleURLScraper"/>`. The `id` attribute is optional and takes the name of the exported scoped variable for the resulting value. The `scope` attribute is optional and takes the scope for `id`. The `channel` attribute is required and takes the channel name for which the class is requested. The `pflist` attribute is optional and takes the list of `PropertyFilter` objects. The `regExp` attribute takes a search string (if filtering the resulting set based on a regular expression.) The `advanced` attribute is optional.

	Returns `java.util.Set`.
`getPropertyType`	Returns the property type for the given property name for a given channel. For example, `<dtadmin:getPropertyType channel="JDCTab/JDCChannel" property="title" id="titleType"/>`.
	The `id` attribute is optional and takes the name of the exported scoped variable for the resulting value. The `scope` attribute is optional and takes the scope for `id`. The `channel` attribute is required and takes the channel name for which the class is requested. The `property` attribute is required and takes the property name.
	Returns `java.lang.short`. The returned value can be compared to one of the following variables available as extra info to determine the type: `STRING_DP`, `INTEGER_DP`, `BOOLEAN_DP`, `COLLECTION_DP`, `UNKNOWN_DP`
`getStringProperty`	Gets the string property value given a channel or container name and the property key. The `PropertyFilter` list is optional; if given, gets the string property based on the property filter's condition and value. If channel name is not set, then the string property value from the global properties is returned. For example, `dtadmin:getStringProperty channel="MyFrontPageTabPanelContainer" key="title"/`.
	The `id` attribute is optional and takes the name of the exported scoped variable for the resulting value. The `scope` attribute is optional and takes the scope for `id`. The `channel` attribute is required and takes the channel name for which the class is requested. The `key` attribute is required. The `pflist` attribute is optional and takes the list of `PropertyFilter` objects.
	Returns `java.lang.String`.
`getIntegerProperty`	Gets the integer property value given a channel or container name and the property key. The `PropertyFilter` list is optional; if given, gets the integer property based on the property filter's condition and value. If channel name is not set, then the integer property value from the global properties is returned. For example, `<dtadmin:getIntegerProperty channel="MyFrontPageTabPanelContainer" key="timeout"/>.`

	The `id` attribute is optional and takes the name of the exported scoped variable for the resulting value. The `scope` attribute is optional and takes the scope for `id`. The `channel` attribute is required and takes the channel name for which the class is requested. The `key` attribute is required. The `pflist` attribute is optional and takes the list of `PropertyFilter` objects. Returns `int`.
`getBooleanProperty`	Gets the boolean property value given a channel or container name and the property key. `PropertyFilter` list is optional; if given, gets the boolean property based on the property filter's condition and value. If channel name is not set, then the boolean property value from the global properties is returned. For example, `<dtadmin:getBooleanProperty channel="MyFrontPageTabPanelContainer" key="parallelChannelsInit"/>.` The `id` attribute is optional and takes the name of the exported scoped variable for the resulting value. The `scope` attribute is optional and takes the scope for `id`. The `channel` attribute is required and takes the channel name for which the class is requested for. The `key` attribute is required. The `pflist` attribute is optional and takes the list of `PropertyFilter` objects. Returns `boolean`.
`getListProperty`	Gets the list property value given a channel or container name and the property key. `PropertyFilter` list is optional; if given, gets the collection property based on the property filter's condition and value. If channel name is not set, then the collection property value from the global properties is returned. The value returned is a List. For example, `<dtadmin:getListProperty channel="MyFrontPageTabPanelContainer" key="categories"/>.` The `id` attribute is optional and takes the name of the exported scoped variable for the resulting value. The `scope` attribute is optional and takes the scope for `id`. The `channel` attribute is required and takes the channel name for which the class is requested for. The `key` attribute is required. The `pflist` attribute is optional and takes the list of `PropertyFilter` objects.

	Returns `java.util.List`.
`getMapProperty`	Gets the map property value given a channel or container name and the property key. `PropertyFilter` list is optional; if given, gets the collection property based on the property filter's condition and value. If channel name is not set, then the collection property value from the global properties is returned. The value returned is a Map. For example, `<dtadmin:getMapProperty channel="MyFrontPageTabPanelContainer" key="channelsRow"/>`. The `id` attribute is optional and takes the name of the exported scoped variable for the resulting value. The `scope` attribute is optional and takes the scope for `id`. The `channel` attribute is required and takes the channel name for which the class is requested for. The `key` attribute is required. The `pflist` attribute is optional and takes the list of `PropertyFilter` objects. Returns `java.util.Map`.
`setStringProperty`	Sets the string property value given a channel or container name and the property key and the value. `PropertyFilter` list is optional; if given, sets the string property based on the property filter's condition and value. If channel name is not set, then the string property value at the global properties is set. For example, `<dtadmin:setStringProperty channel="MyFrontPageTabPanelContainer" key="title" value="New Front Page title-1"/>`. The `channel` attribute is required and takes the channel name for which the class is requested for. The `key` and `value` attributes are required. The `pflist` attribute is optional and takes the list of `PropertyFilter` objects.
`setIntegerProperty`	Sets the integer property value given a channel or container name and the property key and the value. `PropertyFilter` list is optional; if given, sets the integer property based on the property filter's condition and value. If channel name is not set, then the integer property value at the global properties is set. For example, `<dtadmin:setIntegerProperty channel="MyFrontPageTabPanelContainer" key="timeout" value="80"/>`. The `channel` attribute is required and takes the channel name for which the class is requested for. The `key` attribute

	and `value` attributes are required. The `pflist` attribute is optional and takes the list of `PropertyFilter` objects.
`setBooleanProperty`	Sets the boolean property value given a channel/container name and the property key and the value. `PropertyFilter` list is optional. If given sets the boolean property based on the property filter's condition and value. If channel name is not set, then the boolean property value at the global properties is set. For example, `<dtadmin:getBooleanProperty channel="MyFrontPageTabPanelContainer" key="parallelChannelsInit"/>`. The `channel` attribute is required and takes the channel name for which the class is requested for. The `key` and `value` attributes are required. The `pflist` attribute is optional and takes the list of `PropertyFilter` objects.
`setListProperty`	Sets the list property value given a channel or container name, and the property key and the value. `PropertyFilter` list is optional; if given, sets the list property based on the property filter's condition and value. If channel name is not set, then the list property value at the global properties is set. For example, `<dtadmin:setListProperty channel="MyFrontPageTabPanelContainer" key="categories" value="$list"/>`. The `channel` attribute is required and takes the channel name for which the class is requested for. The `key` attribute is required. The `value` attribute is required. The `pflist` attribute is optional and takes the list of `PropertyFilter` objects.
`setMapProperty`	Sets the map property value given a channel/container name and the property key and the value. `PropertyFilter` list is optional. If given sets the map property based on the property filter's condition and value. If channel name is not set, then the map property value at the global properties is set. For example, `<dtadmin:setMapProperty channel="MyFrontPageTabPanelContainer" key="channelsRow" value="$map/>`. The `channel` attribute is required and takes the channel name for which the class is requested for. The `key` and `value` attributes are required. The `pflist` attribute is optional and takes the list of `PropertyFilter` objects.

Tags for Portlet Management Tasks

`obtainPortletAdmin`	All the portlet administration tags should be called from within this nested tag and will be operating on the base distinguished node specified here. For example, `<dtportletadmin:obtainPortletAdmin baseDN="$baseDN">...</dtportletadmin:obtainPortletAdmin>`. The `baseDN` attribute is required and must be a valid node in Directory Server. For example, `dc=red,dc=iplanet,dc=com`.
`createPortletChannel`	Allows creation of a portlet channel based on a portlet. For example, `<dtportletadmin:createPortletChannel channelName="myPortletChannel" portletName="portletsamples.JSPPortlet"/>`. The `channelName` attribute is required and must contain the name of the channel. The `portletName` attribute is required and must contains the name of the base portlet.
`getExistingPortlets`	Gets the list of existing portlets. For example, `<dtportletadmin:getExistingPortlets id="ExistingPortlets"/>`. The `id` attribute is optional and takes the name of the exported scoped variable for the resulting value. The `scope` attribute is optional and takes the scope for `id`. Returns `java.util.Set`.
`getPortletPreferenceNames`	Gets the list of portlet preference names. For example, `<dtportletadmin:getPortletPreferenceNames id="prefNames" portletName="BookmarkPortlet"/>`. The `id` attribute is optional and takes the name of the exported scoped variable for the resulting value. The `scope` attribute is optional and takes the scope for `id`. The `portletName` attribute is required and must contain the name of the portlet channel. Returns `java.util.Set`.
`getPortletPreferenceStringValue`	Gets the portlet preference value as a string given a portlet channel name and the preference key. For example,

	`<dtportletadmin:getPortletPreferenceStringValue id="targetstring" portletName="BookmarkPortlet" prefNam e="targets"/>`. The `id` attribute is optional and takes the name of the exported scoped variable for the resulting value. The `scope` attribute is optional and takes the scope for `id`. The `portletName` attribute is required and must contain the name of the portlet channel. The *prefName* attribute is required and must contain the preference key. Returns `java.lang.String`.
`getPortletPreferenceValues`	Gets the portlet preference values given a portlet channel name and the preference key. For example, `<dtportletadmin:getPortletPreferenceValues id="targets" portletName="BookmarkPortlet" prefName="targets"/>`. The `id` attribute is optional and takes the name of the exported scoped variable for the resulting value. The `scope` attribute is optional and takes the scope for `id`. The `portletName` attribute is required and must contain the name of the portlet channel. The `prefName` attribute is required and must contain the preference key. Returns `java.util.List`.
`setPortletPreferenceStringValue`	Sets the portlet preference value as a string given a portlet channel name and the preference key and value. For example, `<dtportletadmin:setPortletPreferenceStringValue portletName="JSPPortlet" prefName="contentpage" value="content.jsp"/>`. The `portletName` attribute is required and must contain the name of the portlet channel. The `prefName` attribute is required and must contain the preference key. The `value` attribute is required and must contain a string value.
`setPortletPreferenceValues`	Sets the portlet preference values given a portlet channel name and the preference key and an array of String values. For example, `<dtportletadmin:setPortletPreferenceValues portletName="BookmarkPortlet" prefName="targets" prefValues= "$prefnamesvalues"/>` .

The `portletName` attribute is required and must contain the name of the portlet channel. The `prefName` attribute is required and must contain the preference key. The `prefvalues` attribute is required and must contain a string array of values.

Tags for User Management Tasks

`obtainUserAdmin` — All the user administration tags should be called from within this nested tag. For example,
`<dtuseradmin:obtainUserAdmin>...</dtuseradmin:obtainUserAdmin>`

`setUserStatus` — Activates or deactivates a user. For example,

```
<dtuseradmin:setUserStatus
userDN="uid=jdcuser,ou=people,dc=red,dc=iplanet,dc=com"
activate="true"/>
```

The `id` attribute is optional and takes the name of the exported scoped variable for the resulting value. The `scope` attribute is optional and takes the scope for `id`. The `userdn` is a required attribute. The `activate` attribute is required and takes a boolean to specify activate or deactivate.

Returns boolean specifying the status of user.

`resetPassword` — Resets the user's password. For example, `<dtuseradmin:resetPassword userDN="uid=jdcuser,ou=people,dc=red,dc=iplanet,dc=com" newPasswd="jdcuser12"/>`.

The `id` attribute is optional and takes the name of the exported scoped variable for the resulting value. The `scope` attribute is optional and takes the scope for `id`. The `userdn` and `newPasswd` attributes are required.

Returns a boolean specifying if the reset operation was a success or failure.

`searchUsers` — Allows to search for a user. For example, `<dtuseradmin:searchUsers wildcard="j*" id="users"/>`.

The `id` attribute is optional and takes the name of the exported scoped variable for the resulting value. The `scope` attribute is optional and takes the scope for `id`. The `wildcard` attribute is required and takes a search string (if filtering the resulting set based on a regular expression) specifically of the form `foo*`, `*foo`, `foo*bar`, `foo*bar*`.

Returns `java.util.Set`.

`getAssignableRoles`	Gets the list of roles that the currently authenticated user can assign/remove. For example, `<dtuseradmin:getAssignableRoles id="nodes"/>`. The `id` attribute is optional and takes the name of the exported scoped variable for the resulting value. The `scope` attribute is optional and takes the scope for `id`. Returns `java.util.Set`.
`assignRole`	Assigns a particular role to a user. For example, `<dtuseradmin:assignRole userDN="uid=jdcuser,ou=people,dc=red,dc=iplanet,dc=com" roleDN="cn=JDC,dc=red,dc=iplanet,dc=com"/>`. The `id` attribute is optional and takes the name of the exported scoped variable for the resulting value. The `scope` attribute is optional and takes the scope for `id`. The `roleDN` attribute is required and takes the role to be assigned. The `userDN` attribute is required and takes the distinguished name of the user for whom the role is to be assigned. Returns a boolean specifying if the assign operation is success or failure.
`removeRole`	Removes the assigned role for a user. For example, `<dtuseradmin:removeRole userDN="uid=jdcuser,ou=people,dc=red,dc=iplanet,dc=com" roleDN="cn=JDC,dc=red,dc=iplanet,dc=com"/>`. The `id` attribute is optional and takes the name of the exported scoped variable for the resulting value. The `scope` attribute is optional and takes the scope for `id`. The `userDN` attribute is required and takes the distinguished name of the user for whom the role has to be removed. The `roleDN` attribute is required and takes the role to be removed. Returns a boolean specifying if the assign operation is success or failure.
`getUsers`	Gets the list of user distinguished names in the currently logged in administrator's people container. For example, `<dtuseradmin:getUsers id="ExistingUsers"/>`. The `id` attribute is optional and takes the name of the exported scoped variable for the resulting value. The `scope` attribute is optional and takes the scope for `id`. Returns `java.util.Set`.
`createUser`	Creates a user given all the required user attributes. For example, `<dtuseradmin:createUser uid="createtest" firstname="create" lastname="test" fullname="test user" password="createtest"/>`.

	The `uid`, `password`, `fullname`, `firstname`, and `lastname` attributes are required.
`deleteUsers`	Deletes a list of users based on the user distinguished names passed in. For example, `<dtuseradmin:deleteUsers userDNs="$userDNs"/>` where `$userDNs` is a `java.util.Set` of use distinguished names. The *userDNs* attribute is required and takes `java.util.Set` of user DNs to be deleted.
`getUserRoles`	Gets the list of role distinguished nodes the user has already been assigned to. For example, `<dtuseradmin:getUserRoles id="alreadyAssignedUserRoleSet" userDN="uid=jdcuser,ou=people,dc=red,dc=iplanet,dc=com"/>`. The `id` attribute is optional and takes the name of the exported scoped variable for the resulting value. The `scope` attribute is optional and takes the scope for `id`. Returns `java.util.Set`.

Tags for WSRP Management Tasks

`obtainWSRPAdmin`	All the WSRP administration tags should be called from within this nested tag and will be operating on the `baseDN` that is passed in. For example, `<wsrpadmin:obtainWSRPAdmin baseDN="$baseDN">...</wsrpadmin:obtainWSRPAdmin>` . The `baseDN` attribute is required and must be a valid node in Directory Server. For example: `-dc=red,dc=iplanet,dc=com`
`getExistingProducerIds`	Gets the set of producer IDs that are available in the organization of the `baseDN` that this tag is operating on. The producers in this list can be used to get the portlets list. For example, `<wsrpadmin:getExistingProducerIds id="ExistingProducerIds"/>`. The `id` attribute is optional and takes the name of the exported scoped variable for the resulting value. The `scope` attribute is optional and takes the scope for `id`. Returns `java.util.Set`.
`getWSRPPortletHandlers`	Returns the list of portlet ids for the given `producerID` . For example, `<wsrpadmin:getWSRPPortletHandlers id="PortletHandlersList" producerID="EO5DM1IAAAAAJAM42KBAAAA"/>`.

	The `id` attribute is optional and takes the name of the exported scoped variable for the resulting value. The `scope` attribute is optional and takes the scope for `id`. The `producerID` attribute is required and takes the id of the producer. Returns `java.util.Set`.
`createWSRPChannel`	Creates a new WSRP portlet channel given the `channelName`, `portletID`, `producerID`. For example: `<wsrpadmin:createWSRPChannel channelName="MyFrontPageTabPanelContainer/jspPortlet_wsrp2" producerId="EO5DM1IAAAAAJAM42KBAAAA" portletId="JSPPortlet"/>`. The `channelName`, `producerID`, and `portletID` attributes are required.
`getProducerName`	Returns the name of producer for the given `producerID`. For example: `<wsrpadmin:getProducerName id="producerName" producerID="EO5DM1IAAAAAJAM42KBAAAA"/>` . The `id` attribute is optional and takes the name of the exported scoped variable for the resulting value. The `scope` attribute is optional and takes the scope for `id`. The `producerID` attribute is required and takes the id of the producer. Returns `java.lang.String`.
`getPortletName`	Returns the name of portlet for the given `producerID` and `portletID`. For example, `<wsrpadmin:getPortletName id="portletName" producerID="EO5DM1IAAAAAJAM42KBAAAA" portletID="JSPPortlet"/>`. The `id` attribute is optional and takes the name of the exported scoped variable for the resulting value. The `scope` attribute is optional and takes the scope for `id`. The `producerID` attribute is required and takes the id of the producer. The `portletID` attribute is required. Returns `java.lang.String`.

Accessing Sun Resources Online

The docs.sun.com^{SM} web site enables you to access Sun technical documentation online. You can browse the docs.sun.com archive or search for a specific book title or subject. Books are available as online files in PDF and HTML formats. Both formats are readable by assistive technologies for users with disabilities.

To access the following Sun resources, go to `http://www.sun.com`:

- Downloads of Sun products
- Services and solutions
- Support (including patches and updates)
- Training
- Research
- Communities (for example, Sun Developer Network)

Third-Party Web Site References

Third-party URLs are referenced in this document and provide additional, related information.

Note – 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 full document title and part number. The part number is a 7-digit or 9-digit number that can be found on the book's title page or in the document's URL. For example, the part number of this book is 819-5454-10.