SAP NetWeaver
How-To Guide

How To... Automate Content Creation via XML (XML Content and Actions)

Applicable Releases:

SAP Enhancement Package 1 for SAP NetWeaver Composition Environment 7.1

Topic Area:

User Productivity

Capability:

Portal and Collaboration

Version 1.10

December 2008

Document History

Document Version	Description
1.10	Entire guide structure revised. Updates: • How to create business objects and operations for OBN, and assign an iView or a page as an OBN target. • How to create display rules. • In SAP NetWeaver Composition Environment 7.1, in a Federated Portal Network (FPN) a portal can only function as a producer, and not as a consumer. All consumer-related documentation was removed.
1.00	First official release of this guide for SAP NetWeaver Composition Environment 7.1. (Apr 2008)

Typographic Conventions

Type Style	Description
Example Text	Words or characters quoted from the screen. These include field names, screen titles, pushbuttons labels, menu names, menu paths, and menu options. Cross-references to other documentation
Example text	Emphasized words or phrases in body text, graphic titles, and table titles
`Example text`	File and directory names and their paths, messages, names of variables and parameters, source text, and names of installation, upgrade and database tools.
`Example text`	User entry texts. These are words or characters that you enter in the system exactly as they appear in the documentation.
`<Example text>`	Variable user entry. Angle brackets indicate that you replace these words and characters with appropriate entries to make entries in the system.
`EXAMPLE TEXT`	Keys on the keyboard, for example, F2 or ENTER.

Icons

Icon	Description
	Caution
	Note or Important
	Example
	Recommendation or Tip

Table of Contents

1. XML Content and Actions Feature

Purpose

The XML Content and Actions feature enables administrators to use XML to automate the creation of portal semantic objects (such as iViews, pages and systems) and to perform actions (such as assigning roles or deleting content). The content and actions are specified in an XML file, which is imported into the portal. The portal parses the XML and generates the specified content and performs the specified actions.

The XML import process enables the creation of mass content without the need to use the portal wizards and editors. In addition, advanced users can perform batch operations and make pinpoint modifications within a large content base.

> **Note**
>
> In previous versions of the portal, the XML Content and Actions feature was known as the Generic Creator service.

The XML Content and Actions feature is not to be confused with the portal's Transport mechanism. Use the XML Content and Actions feature to create new content, whereas the Transport mechanism should be used to move content from one portal to another. The Transport mechanism also provides additional functionalities, such as multi-language support, as well as the transport of applications and not just Portal Content Directory (PCD) content.

Creating Valid XML

The imported XML file must adhere to the specifications described in this document. The XML can be coded in a number of ways, including using scripts that transform Microsoft Excel or text documents to XML. Such services are not supplied by SAP.

You can also build a template for the XML file based on existing content. The portal provides an export tool that creates an XML file from existing content. You can edit the exported XML as necessary, and create template XML files, which can then be imported into a portal to create content.

Constraints/Limitations

- The portal does not include an editor for viewing, editing, or validating the XML before it is imported.
- The XML Content and Actions feature does not support the creation (import) and export of multi-value attributes. This limitation also affects the export of Object-Based Navigation (OBN) targets: when an iView or a page is the OBN target for more than one operation, not all the operations are exported. Therefore, it is not recommended to use the XML Content and Actions feature for exporting OBN targets that implement more than one operation.

Intended Audience

The intended audience of this guide is content administrators or developers with knowledge in XML scripting.

This guide requires familiarity with portal structure, portal semantic objects and actions that can be performed in the portal. For more information, refer to portal documentation in SAP NetWeaver Library at **`help.sap.com/netweaver`**.

1.1 Architecture

This section describes the major components of the XML Content and Actions feature, as well as the internal process flow that occurs when an XML file is imported.

1.1.1 Key Components

The following are the key components involved in creating content and performing actions:

Component	Function
XML Content and Actions export tool iView	• Creates an XML file based on selected content. This file is a well-formed and valid XML file that can be imported to a portal to create the same content.
XML file (created by an administrator)	• Specifies the objects to be created, updated and deleted • Specifies the actions to be performed.
XML Content and Actions import tool iView	• Imports the XML file to the portal and passes it to the Generic Creator engine for processing
Generic Creator engine	• Parses the XML • Associates each XML block with the appropriate handler • Manages the process of creating content and performing actions, including error handling
Handlers	• Each handler is responsible for creating, modifying or deleting a specific semantic object, or for performing a specific action.

1.1.2 Process Flow

The following describes the process flow when importing an XML file via the import tool iView:

1. The XML is checked to make sure that it is well formed.

   If the XML is not well formed, the import process is aborted.

2. The XML is parsed to determine what objects need to be created and what actions need to be performed.

   A set of building blocks is written into an execution queue.

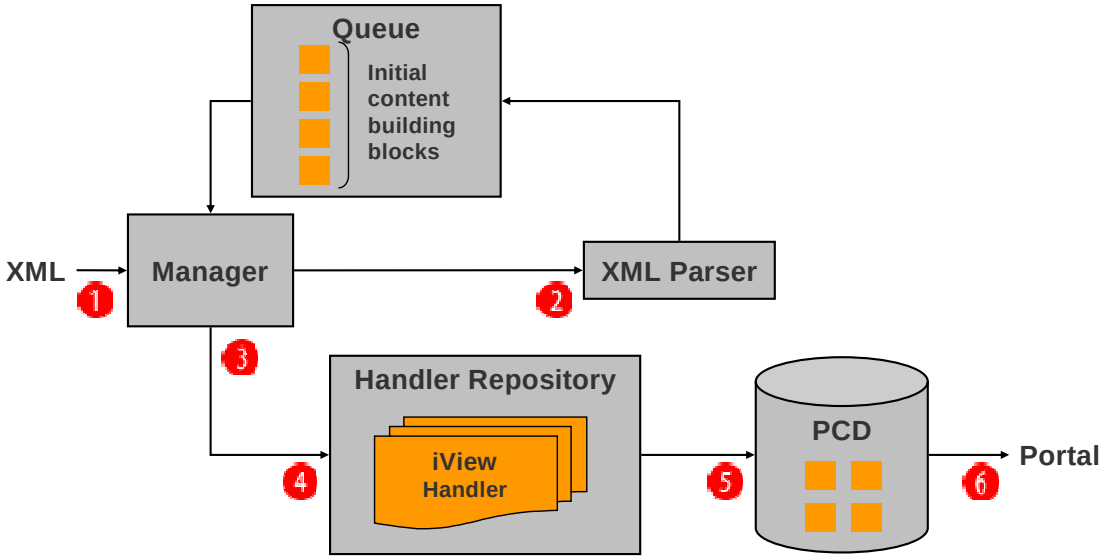
3. Required handlers are loaded.

4. Handlers check each building block to make sure it can be executed.

   If at least one object or action in the queue cannot be executed, the entire process is aborted without writing anything to the PCD.

5. Handlers write the objects to the PCD and perform the actions specified in the XML.

6. A report on the results of the import is displayed in the import tool iView.



2. Workflow for XML Content and Actions

Purpose

This section describes the typical workflow for creating content and performing actions.

Prerequisites

- You need to plan what semantic objects to create and which actions to perform.

  One method is to create a text file or Excel spreadsheet to list the required objects and actions, and then to run a script that generates XML from the text or Excel file.

  Another method is to create objects in the portal, and then export the content to XML, and use this XML as a template for the XML file to be imported.

- Before importing the XML, all portal components on which objects defined in the XML file are based, must already exist in the portal, including page layouts.

Workflow

1. Create a well-formed and valid XML file according to the general requirements of the XML parser, as described in *XML Elements and Attributes* on page 5, and the specific requirements of the handlers that you are using, as described in *Working with Semantic Objects and Actions* on page 17.

   Note

   You can also create an XML file by exporting content from the portal to XML and then editing this file. For more information on exporting content, refer to *Exporting Content* on page 45.

2. Import the XML file via the XML Content and Action import tool in the portal, as described in *Importing Content and Actions* on page 48.

   The tool is available in the portal at *System Administration* → *Transport* → *XML Content and Actions* → *Import*.

3. Review the results of the import in the user interface of the XML Content and Action import tool. The import report is described in *Importing Content and Actions* on page 48.

4. Check and test the content in the portal.

3. XML Elements and Attributes

This section describes the schema for XML files parsed by the Generic Creator engine. The XML file has the following sections:

- **General Details**: Defines general details and configuration settings for the XML import. These are defined by attributes in the `GenericCreator` root element, as described in *Defining General Details: GenericCreator Element* on page 6. This section is mandatory.
- **Global Parameters**: Defines global properties and values that can be used throughout the XML file, as described in *Defining Global Variables: Property Element* on page 8.
- **Context or Action Element Blocks**: Defines either a semantic object to create or modify (`Context` element) or an action to perform (`Action` element), as described in *Defining Semantic Objects: Context Element* on page 9 and in *Defining Actions: Action Element* on page 14.

3.1 Defining General Details: GenericCreator Element

The `GenericCreator` element must be the root element. It configures the XML file and the behavior of the XML parser.

Definition

The following is the format for the `GenericCreator` element:

```
<GenericCreator author="<author_name>"
    version="<version_and_description>" mode="<mode1>, <mode2>"
    report.level="<report_level>" ignore="<ignore_mode>"
    default.locale="<locale_ID>" createMode="<overwrite_mode>">
```

The following table describes the `GenericCreator` element attributes:

Attribute	Mandatory	Description
`author`	No	Specifies the name of the author of the XML file. The author does not have to be defined as a user in the portal. This attribute has no effect on the portal or the XML import.
`createMode`	Yes	Specifies what to do when the XML defines an object that already exists in the PCD. Valid only when the `mode` attribute is `execute`. This attribute can also be applied to `Context` elements. The following are valid values: 1: If the object exists, then **do nothing** (default). 2: If the object exists, then **replace** the entire existing object and its properties with the new one. 3: If the object exists, then **update** only the properties that are declared in the XML document. In other words: (i) if the XML defines properties that already exist for the existing object, they are updated; (ii) if the XML defines new properties, they are added to the existing object; and (iii) if the existing object contains properties that are not defined in the XML, they remain unchanged. This attribute applies to each sub-block within the block that specifies it, unless the sub-block overrides the attribute value. For example, if the value in the `GenericCreator` element is **1**, then all XML blocks are skipped if the objects they define already exist in the PCD, unless `createMode` in a specific XML block is 2 or 3.
`default.locale`	Yes	Specifies the default locale for an object if its `Context` element does not specify a locale attribute (`originalLocale`). Only meaningful if a `Context` element specifies an attribute of type `text`, for example, `Title`.

`ignore`	Yes	Specifies whether an XML block is executed. This attribute can also be applied to `Context` and `Action` elements. This attribute applies to each sub-block within the block that specifies it, unless the sub-block overrides the attribute value. For example, if the value in the `GenericCreator` element is **`true`**, then all XML blocks are skipped, unless `ignore` in a specific XML block is **`false`**. The following are valid values: **`true`**: The block is not executed. **`false`**: The block is executed (default). See also *Executing Specific XML Blocks* on page 43.
`mode`	Yes	Specifies the mode for content creation. The following are valid values: `clean`: Objects defined within `Context` elements are removed from the PCD. Most actions defined by `Action` elements are not performed, although each handler that performs an action can define an alternative action for this mode. `execute`: Objects defined within `Context` elements are created in the PCD. Actions defined by `Action` elements are performed. The value of the `createMode` attribute (see below) determines how the `execute` mode is performed. You can define multiple values for this attribute. If you specify more than one value – separated by commas – the script is executed once in the first mode, once in the second mode, and so on. For example, this attribute is used with `clean, execute` in order to first clean previously-created content, before creating the new content.
`report.level`	Yes	Specifies which messages are reported after an XML file has been imported. The following are valid values (default report levels implemented by the standard handlers): 1. `debug` 2. `info` 3. `warning` 4. `success` 5. `fail` Results are displayed from the selected report level and higher; for example, if `warning` is defined as the report level, then results of type `warning`, `success` and `fail` will also be displayed. If `debug` is defined, then all result types will be displayed.

`version`	No	Specifies the version or a short description of the XML document. This attribute has no effect on the portal or the XML import.

Example

```
<GenericCreator author="Joe Soap"
  version="Initial Content Bank 9/3/2005 6:19PM"  mode="clean, execute"
  report.level="success" ignore="false" default.locale="en"
createMode="2">
```

3.1.1 Defining Global Variables: Property Element

The `Property` element enables you to define global variables, and reuse them as needed anywhere in the XML document. This is useful for frequent occurrences of parameters in the XML document, such as the namespace and the default locale.

Definition

Each `Property` element defines a single property name-value pair. A `Property` element must be nested within the root `GenericCreator` element. The `Property` element is defined as follows:

```
<Property name="<property_name>" value="<property_value>"/>
```

The following table describes the `Property` element attributes:

Attribute	Mandatory	Description
`name`	Yes	Specifies the name of the property variable.
`value`	Yes	Specifies the value of the property variable.

Usage

Any property name-value pair defined and nested in the `GenericCreator` element can be used elsewhere in the XML document as follows:

```
${<property_name>}
```

Example

The following example shows how to define global variables, such as `namespace` and `locale`, and use them within other elements:

```
<!—PROPERTY DEFINITION -->
<Property name="namespace" value="com.sap.portal"/>
<Property name="locale" value="en"/>
...
...
<!—PROPERTY USAGE -->
<Context name="${namespace}.urliview"
    template="par:/applications/com.sap.portal.urliviews/"
    objectClass="com.sapportals.portal.iview" create_as="0"
    domain="EP" originalLocale="${locale}" title="URL iView"/>
```

3.1.2 Defining Semantic Objects: Context Element

The `Context` element defines a semantic object to be created, deleted or updated in the PCD.

Definition

The `Context` element defines a semantic object. Certain attributes in the `GenericCreator` and `Context` elements determine which type of action is performed on the object: create, delete or update.

The `Context` element is defined as follows:

```
<Context name="<object_ID>" objectClass="<object_class>"
    template="<source_object>" create_as="<type_of_object" >
```

> **Important**
>
> Typically, the `Context` element can support any attribute and sub-element, assuming it can be parsed by the object's handler and is valid for the object type. Some attributes and sub-elements are mandatory. This guide describes only the basic and commonly-used attributes and sub-elements, including all mandatory attributes.

The following table describes the basic and commonly-used `Context` element attributes:

Attribute	Mandatory	Description
`Collection`	No	Specifies the collection setting of the object. Note that this attribute is for SAP internal use only, to support in-house translation mechanisms.

`container`	Yes[1]	Specifies the container name in a page layout in which to position the iView/page. The container name must exist in the primary page layout defined for the page to which the current object is assigned.
`create_as`	Yes	Specifies the relationship of the object to the template or portal component on which it is based. This attribute is dependent on the `template` attribute. The following are valid values: **`0`**: For the following cases: To create a **new** object that is based directly on a portal component. The `template` attribute specifies the portal component. To make a **copy** of an object that already exists in the PCD. The new object and the copied object become siblings and share the same source object (via a delta link) or portal component. The `template` attribute specifies the source object. **`1`**: To create an object that is a **delta link** of an object that already exists in the PCD. The `template` attribute specifies the source object. Important This value cannot be used if the `template` attribute specifies a portal component. It must specify a semantic object. Typically, delta link objects inherit properties and values from their source objects. To assign different object properties, use the `Attribute` and `AttributeValue` elements. For code samples, refer to *Creating iViews* on page 23.
`createMode`	No	Functions in the same way as the `createMode` attribute in the root `GenericCreator` element. For more information, refer to *Defining General Details: GenericCreator Element* on page 6. If you define a value for this attribute in the `Context` element, it overrides the value defined in the `GenericCreator` element. However, if no value is defined in the `Context` element, the value defined in the `GenericCreator` element is used.
`Domain`	No	Specifies the domain of the object. Note that this attribute is for SAP internal use only, to support in-house translation mechanisms.

`ignore`	No	Functions in the same way as the `ignore` attribute in the root `GenericCreator` element. For more information, refer to *Defining General Details: GenericCreator Element* on page 6. If you define a value for this attribute in the `Context` element, it overrides the value defined in the `GenericCreator` element. However, if no value is defined in the `Context` element, the value defined in the `GenericCreator` element is used.
`name`	Yes	Specifies the object ID (technical name) of the object. Do not specify the full PCD path of the object. An object's PCD location is derived by its ID and the IDs of the `Context` elements in which it is nested. This is why nesting multiple `Context` elements is important for generating a hierarchy in the PCD.
`noTemplateNeeded`	No	Indicates to create a new object based only on the attributes nested in the `<Context>` tag, and not based on either a PCD template or an application. Valid values are **`true`** and **`false`** (default). Note that when set to `false`, it is necessary to specify a value for the `template` attribute.
`objectClass`	Yes	Specifies the type of object. For a list of `objectClass` values, see *Semantic Objects* on page 18.
`originalLocale`	Yes[2]	Specifies the original locale of the object. **Important** Note that this value must only be set for unit (standalone) objects in the PCD. A unit object is one that is **not** currently assigned to another object type. The following are **not** unit objects: an iView in a page, a workset in a role, or a page in a role.

`parent`	No	Specifies the ID and path of the parent PCD folder for the current object.
		
		Note
		This attribute can only be used in a root `Context` element (one that is not nested in another `Context` element).
		
		The attribute enables you to associate the object to an existing PCD hierarchy, while defining it in the XML as a root `Context` element. This attribute is also useful for making specific modifications to a particular object located within a complex hierarchy.
		
		Important
		If you want your content to be created in the standard *Portal Content* (`portal_content`) root folder of the Portal Catalog, you nevertheless need to define this in your XML, instead of using the `parent` attribute. It is recommended that you create a `Context` element which nests your entire content script.
		
		For example:
		
		`<Context name="portal_content" objectClass="com.sap.portal.pcd.gl.GlContext" title="Portal Content">`
		
		`<!- Define content script -->`
		
		`</Context>`
		
		See also *Creating Hierarchies Without Nested Elements* on page 43.

`PrimaryLayout`	Yes[1]	Specifies whether a page layout assigned to a page is the primary (or default) layout. A page can only have one primary page layout. If you assign more than one primary layout, the last one assigned is the primary layout. The following are valid values: **`true`**: The page layout is the primary layout. **`false`**: The page layout is not the primary layout.

`template`	Yes[3]	Specifies one of the following: • The source object to which the current object is related through a delta link or copy. Use the following syntax: **`pcd:/<PCD_path>`** Note It is possible to create a delta link to a source object that does not yet exist in the PCD. Technically, you will be generating a dangling link; however you can later create the missing object. • The portal component on which an object is based. Use the following syntax: **`gpar:/<DC_name>/<component_name>`** Only iViews, pages and page layouts can be based on portal components. Note The portal components on which objects are based must be deployed to the portal before importing an XML script. • The Web Dynpro application on which an object is based. Use the following syntax: **`gwd:/<DC_name>/`** **`<app_name>_<variant>`** Only pages and page layouts can be based on Web Dynpro applications. Note The Web Dynpro applications on which objects are based must be deployed to the portal before importing an XML script. Note Do not confuse this attribute with an object template, defined using the `IsTemplate` property.
`title`	No	Specifies the friendly name of an object. If this attribute is not defined, then the value of the `name` attribute is displayed in the Portal Catalog instead.

[1] Mandatory only for portal pages and iViews that are embedded in a portal page.

[2] To be used ONLY for standalone or unit objects in the PCD. For example, do not apply this attribute to an iView inside a page, or a workset inside a role.

[3] Mandatory only for delta link target objects and objects that link directly to a portal component.

Usage

To place an object inside another object, for example, an iView in a page, nest the `Context` element of the child object within the `Context` element of the parent object. You can also use the `parent` attribute in the child `Context` element instead of nesting elements.

Keep in mind that the Portal Catalog displays only folders and unit objects (parent objects). To access nested child objects in the portal, you need to open the parent object in its editor.

Example

```
<Context parent="portal_content" name="myFolder"
    objectClass="com.sap.portal.pcd.gl.GlContext"
    title="My Folder" originalLocale="en">
```

For additional examples, refer to *Code Samples for Semantic Objects* on page 20.

3.1.3 Defining Actions: Action Element

XML elements of type `Action` differ in concept and syntax to XML elements of type `Context`. The `Action` element generally performs an action within the portal, instead of generating or updating a semantic object in the PCD.

Actions are typically general; they tend not to be specific to a particular object type (although it is possible to develop a handler of type `Action` that operates on a particular content type).

Definition

The `Action` element is defined as follows:

```
<Action id="<handler_name>" ignore="<mode>" />
```

The following table describes the basic attributes of the `Action` element:

Attribute	Mandatory	Description
`id`	Yes	Specifies the action. For a list of values, see Actions.
`ignore`	No	Functions in the same way as the `ignore` attribute in the root `GenericCreator` element. For more information, see Defining General Details Using the GenericCreator XML Element. If you define a value for this attribute in the `Action` element, it overrides the value defined in the `GenericCreator` element. However, if no value is defined in the `Action` element, the value defined in the `GenericCreator` element is used.

There may be additional attributes specific to each handler.

Usage

- `Action` elements cannot be nested within each other, nor can they be nested within `Context` elements, or vice versa.
- `Action` elements are only executed when the script is parsed in `execute` mode. The mode is specified in the `mode` attribute specified in the `GenericCreator` root element.

3.1.4 Defining Properties: Attributes, Attribute, AttributeValue Elements

The `Attributes`, `Attribute` and `AttributeValue` elements enable you to define properties (metadata) for semantic objects in the PCD. In the portal, properties are viewed and edited within the Property Editor.

Typically, some object properties and values are defined in portal components (in PAR files). Therefore, use `Attributes`, `Attribute` and `AttributeValue` elements in the following cases:

- To assign a different value to an existing property so that it does not inherit the predefined value from a source object (in the case of a delta link) or its portal component.
- To assign values to existing properties that are not initially assigned a value.
- To define new properties

The `Attributes`, `Attribute` and `AttributeValue` elements can also be used to pass information to configure an action.

Definition

The `Attribute` element defines a property and the `AttributeValue` elements nested within an `Attribute` element define the values for that property.

All `Attribute` elements for a `Context` or Action element must be nested within an `Attributes` element, which takes no attributes.

The format for `Attribute` and `AttributeValue` elements is as follows:

```
<Attributes>
    <Attribute name="<attribute_name>" type="<attribute_type>"
        Inheritance="<attribute_inheritance>"
        <AttributeValue value="<value>" locale="<locale>" />
</Attribute>
    ... (other attributes)
</Attributes>
```

The following describes the `Attribute` element attributes:

Attribute	Mandatory	Description and Valid Values
`Inheritance`	Yes	Specifies the inheritance mode of the property. This property is currently not supported by the portal. Set this to **`NONFINAL`**.
`name`	Yes	Specifies the name of the property. Property names can be found by opening the property editor for an object in the Portal Catalog. Also, the portal API provides interfaces that define constants for property names of semantic objects, for example, `IAttriView` for properties of iViews.
`type`	Yes	Specifies the property's data type. The following are valid values: • **`string`** • **`text`** • **`integer`** • **`boolean`** • **`double`**

The following describes the `AttributeValue` element attributes:

Attribute	Mandatory	Description and Valid Values
`locale`	Yes[1]	Specifies the locale of the property's value (for text-based data, where `type="text"`)
`value`	Yes	Specifies the value of the property variable

[1] Mandatory only for properties of type `text`.

Example

```
<Context>
    ...
    <Attribute name="com.sap.portal.pcm.Description" type="text">
        <AttributeValue value="Schedule Processing" locale="en" />
    </Attribute>
    ...
</Context>
```

3.1.4.1 Meta-Attributes

To define meta-attributes, nest `Attribute` elements. The following defines the `category` meta-attribute for the `myProperty` attribute:

```
<Context>
    ...
    <Attribute name="myProperty" type="text">
        <AttributeValue value="ABC" locale="en" />
        <Attribute name="category" type="text">
             <AttributeValue value="myProperties" locale="en"/>
        </Attribute>
   </Attribute>
    ...
</Context>
```

4. Working with Semantic Objects and Actions

This section provides the following information:

- List of semantic objects
- List of actions
- Code samples for creating semantic objects
- Code samples for executing actions
- Tips and tricks

Semantic Objects

The following table lists the semantic objects that can be created or modified via an imported XML file, and the corresponding values to specify for the `objectClass` attribute in the `Context` element:

Semantic Object	Object Class
Business Objects	**`com.sap.portal.obn.businessObject`**
Desktops	**`com.sapportals.portal.desktop`**
Display Rules	**`com.sapportals.portal.resolving.rule`**
Folders (in Portal Catalog)	**`com.sap.portal.pcd.gl.GlContext`**
iViews	**`com.sapportals.portal.iview`**
Operations (OBN)	**`com.sap.portal.obn.operation`**
Page Layouts	**`com.sapportals.portal.layout`**
Pages	**`com.sapportals.portal.page`**
Role Folders	**`com.sapportals.portal.rolefolder`**
Roles	**`com.sapportals.portal.role`**
Systems	**`com.sapportals.portal.system`**
Translation Worklists	**`com.sap.portal.pcd.translation.TranslationWorklist`**
Transport Packages	**`com.sapportals.portal.transport.TransportPackage`**
Worksets	**`com.sapportals.portal.workset`**

In addition to creating semantic objects and setting attributes, you can also perform the following tasks with the `Context` element:

- Assign iViews to a Page**:** You can also assign iViews and pages to worksets, or worksets to roles.
- Create Related Items**:** You can create Related Items links or Dynamic Navigation iViews for a specific iView or page.
- Assigning an OBN Target: You can assign an iView or a Page as an OBN (object-based navigation) target using the following object class: **`com.sap.portal.obn.operationImplementation`**

Actions

The following table lists actions that can be performed via an imported XML file, and the corresponding values to specify for the `id` attribute in the `Action` element:

Action	ID
Adding/Removing System Aliases	**`com.sap.portal.alias.handler`**
Assigning Users/Groups to Roles	**`com.sap.portal.roleassignment`**
Configuring Proxy Settings	**`com.sap.portal.proxy`**
Copying Content	**`com.sap.portal.copy`**
Deleting Content	**`com.sap.portal.gc.deepCleaner`**
Mirroring Content	**`com.sap.portal.mirror`**
Running Another Script	**`com.sap.portal.script.runner`**
Setting Permissions	This tag has a special syntax, and does not use the `<Action>` tag.

4.1 Code Samples for Semantic Objects

This section contains basic XML code samples for creating and modifying semantic objects.

4.1.1 Creating Business Objects

In addition to the basic attributes required by the `Context` element, the following attributes are used to create a Business Object.

Attribute	Mandatory	Description and Valid Values
`com.sap.portal.pcm.Description`	No	A description of the business object, displayed when hovering over it with the mouse.
`com.sap.portal.pcm.Title`	Yes	The (friendly) name of the business object, as it appears in the Portal Catalog.
`SystemAlias`	No You must provide an alias when the business object `Type` is `SYSTEM_OBJECT`.	Either the alias of an existing system, or any string that can be used to uniquely identify a business object, such as a namespace. The PCD ID of a business object is `<SystemAlias>.<TechnicalName>`
`TechnicalName`	Yes	The technical name of the business object.
`Type`	Yes	Either `SYSTEM_OBJECT` or `CONTEXT_OBJECT`.

The following creates the business object `alias1.businessObject1` in the folder `myFolder` under the `Business_Objects` folder in the PCD.

```
<Context parent="pcd:Business_Objects" name="OBN_SYSTEM_BO"
objectClass="com.sap.portal.obn.businessObject">
<Attributes>
  <Attribute name="Type" type="string">
      <AttributeValue value="SYSTEM_OBJECT" />
  </Attribute>
  <Attribute name="SystemAlias" type="string">
    <AttributeValue value="alias1" />
  </Attribute>
  <Attribute name="TechnicalName" type="string">
    <AttributeValue value="businessObject1" />
  </Attribute>
  <Attribute name="com.sap.portal.pcm.Title" type="string">
    <AttributeValue value="myFirstBusinessObject" />
  </Attribute>
</Attributes>
</Context>
```

4.1.2 Creating Desktops

The following creates a portal desktop named `Default Portal Desktop`.

The list of themes, specified for the attribute `com.sappportals.portal.desktop.allThemes`, is written in StringList format and is composed of the following strings: the name of the theme and the PCD location of the theme. (The number preceding each string indicates its length.)

```
<Context name="${namespace}.defaultDesktop"
objectClass="com.sapportals.portal.desktop" create_as="0" asUnit="true"
collection="${collection}" domain="EP" originalLocale="${locale}"
defaultTheme="sap_standard" defaultFwPage="${namespace}.frameworkpage"
title="Default Portal Desktop">
    <Attributes>
        <Attribute
        name="com.sappportals.portal.desktop.defaultFwPage"
        type="string">
            <AttributeValue value="${namespace}.frameworkpage"/>
        </Attribute>
        <Attribute
        name="com.sappportals.portal.desktop.defaultTheme"
        type="string">
            <AttributeValue value="sap_tradeshow"/>
        </Attribute>
        <Attribute name="com.sappportals.portal.desktop.allThemes"
        type="string">
            <AttributeValue value="13:sap_tradeshow39:pcd:portal_co
ntent/themes/sap_tradeshow"/>
            <AttributeValue value="12:sap_standard038:pcd:portal_co
ntent/themes/sap_standard"/>
            <AttributeValue value="10:sap_chrome00036:pcd:portal_co
ntent/themes/sap_chrome"/>
            <AttributeValue value="07:sap_hcb00000033:pcd:portal_co
ntent/themes/sap_hcb"/>
            <AttributeValue value="12:sap_highcont038:pcd:portal_co
ntent/themes/sap_highcont"/>
        </Attribute>
    </Attributes>
    <Context name="frameworkPages"
    objectClass="com.sap.portal.pcd.gl.GlContext">
        <Context name="${namespace}.frameworkpage"
        template="portal_content/com.sap.pct/every_user/general
        /${namespace}.frameworkpage" create_as="1"
        objectClass="com.sapportals.portal.page"/>
    </Context>
</Context>
```

4.1.3 Creating Display Rules

The following creates a display rule named `MyRule`.

```
<Context name="MyRule" objectClass="com.sapportals.portal.resolving.rule"
create_as="0" parent="portal_content/myrulesfolder">
    <Attributes>
        <Attribute name="com.sapportals.portal.resolving.rules.xml"
         type="string">
            <AttributeValue value="&lt;CONDITIONS
             version=&quot;1.0&quot;&gt;&lt;IF
             value=&quot;UrlAlias==portal/new&quot;&gt;&lt;RETURN
             name=&quot;PORTAL_NEW&quot; value=&quot;
             pcd:portal_content/newDesktop&quot;/&gt;
             &lt;/IF&gt;&lt;IF value=&quot;User==*&quot;&gt;
             &lt;RETURN name=&quot;PORTAL_DESKTOP&quot;
             value=&quot;pcd:portal_content/defaultDesktop&quot;
             /&gt;&lt;/IF&gt;&lt;/CONDITIONS&gt;"/>
        </Attribute>
    </Attributes>
</Context>
```

Note that angle brackets (< >) are reserved for XML tags. In other cases, **`&lt`** and **`&gt`** are used instead.

4.1.4 Creating Folders in the Portal Catalog

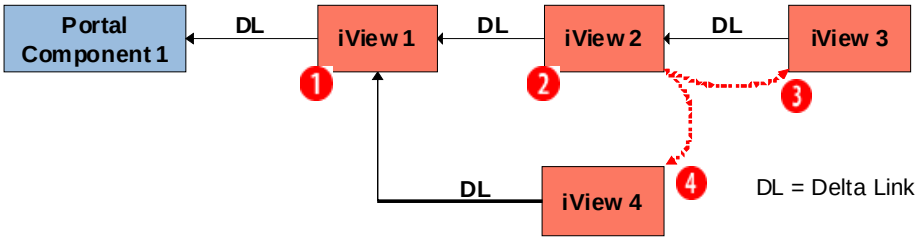
The following creates a folder named `Migrated Content` in the root `Portal Content` folder.

```
<Context name="portal_content"
objectClass="com.sap.portal.pcd.gl.GlContext"
title="Portal Content">
    <Context name="com.sap.portal.migrated"
    objectClass="com.sap.portal.pcd.gl.GlContext"
    title="Migrated Content"/>
</Context>
```

4.1.5 Creating iViews

iViews, pages and systems can be created directly from portal components or as copies or delta links of other portal objects. The method for creating iViews, pages and systems can be specified in the XML, generally with the attributes `template` and `create_as`.

The following shows several ways to create iViews with varying dependencies. In this example, iView 1, 2, 3, and 4 are all based on the same portal component. The legend describes the `template` and `create_as` attributes for defining each iView.



1. template="par:/applications/portalcomponent1"; create_as="0"
2. template="pcd:/portal_content/myFolder/iView1"; create_as="1"
3. template="pcd:/portal_content/myFolder/iView2"; create_as="1"
4. template="pcd:/portal_content/myFolder/iView2"; create_as="0"

The following examples show how to create an iView based on a portal component, or based on a copy or delta link of another portal object.

More information: Assigning an IView or a Page as an OBN Target

4.1.5.1 Based on a Portal Component

The following creates an iView from a PAR file and based on the portal component `com.sap.portal.ivs.alias_editor.AliasEditor`. See iView 1 in the figure above.

```
<Context name="${namespace}.aliasEditor"
title="System Alias Editor"
template="par:/applications/com.sap.portal.ivs.alias_editor/components/Al
iasEditor" objectClass="com.sapportals.portal.iview" create_as="0"
collection="${collection}" domain="EP" originalLocale="${locale}">
    <Attributes>
        <Attribute
        name="com.sap.portal.reserved.iview.IsolationMode"
        type="string">
            <AttributeValue value="PUMPED"/>
        </Attribute>
        <Attribute name="com.sap.portal.iview.HeightType"
        type="string">
            <AttributeValue value="FULL_PAGE"/>
        </Attribute>
        <Attribute name="com.sap.portal.iview.ShowTray"
        type="string">
            <AttributeValue value="false"/>
        </Attribute>
        <Attribute name="com.sap.portal.reserved.iview.ParamList"
        type="string">
            <AttributeValue value="*"/>
        </Attribute>
    </Attributes>
</Context>
```

4.1.5.2 Based on iView (Delta Link)

The following creates an iView based on a delta link from the iView located at `portal_content/com.sap.pct/admin.templates/iviews/${namespace}.contentCatalog`. See iView 2 in the figure above.

```
<Context name="${namespace}.contentCatalog"
template="portal_content/com.sap.pct/admin.templates/iviews/${namespace}.
contentCatalog" objectClass="com.sapportals.portal.iview" create_as="1"
container="com.sap.portal.reserved.layout.Cont2">
    <Attributes>
        <Attribute
        name="com.sap.portal.reserved.iview.IsolationMode"
        type="string">
            <AttributeValue value="URL"/>
        </Attribute>
        <Attribute name="com.sap.portal.iview.HeightType"
        type="string">
            <AttributeValue value="FULL_PAGE"/>
        </Attribute>
        <Attribute name="com.sap.portal.iview.ShowTray"
        type="string">
            <AttributeValue value="false"/>
        </Attribute>
    </Attributes>
</Context>
```

4.1.5.3 Based on iView (Copy)

The following creates an iView by copying the iView located at `portal_content/com.sap.pct/admin.templates/iviews/${namespace}.contentCatalog`. See iView 3 in the figure above.

```
<Context name="${namespace}.contentCatalog"
template="portal_content/com.sap.pct/admin.templates/iviews/${namespace}.
contentCatalog" objectClass="com.sapportals.portal.iview" create_as="0"
container="com.sap.portal.reserved.layout.Cont2">
    <Attributes>
        <Attribute
        name="com.sap.portal.reserved.iview.IsolationMode"
        type="string">
            <AttributeValue value="URL"/>
        </Attribute>
        <Attribute name="com.sap.portal.iview.HeightType"
        type="string">
              <AttributeValue value="FULL_PAGE"/>
        </Attribute>
        <Attribute name="com.sap.portal.iview.ShowTray"
        type="string">
            <AttributeValue value="false"/>
        </Attribute>
    </Attributes>
</Context>
```

4.1.6 Creating Operations

In addition to the basic attributes required by the `Context` element, the following attributes are used to create an operation to attach to a Business Object, for OBN (Object-Based Navigation).

Attribute	Mandatory	Description and Valid Values
`BOID`	Yes	The (technical) name of the business object under which the operation is created.
`com.sap.portal.pcm.Description`	No	A description of the operation.
`com.sap.portal.pcm.Title`	Yes	The (friendly) name of the operation, as it appears in the Portal Catalog.
`OperationID`	Yes	The technical name of the operation.
`com.sap.portal.unification.semanticlayer.usloperation.priority`	No	The priority of the operation. When this attribute is not used, the default priority is zero.
`SystemAlias`	Yes	Either the alias of an existing system, or any string that can be used to uniquely identify a business object, such as namespace.

The following creates for the business object `alias1.businessObject1` an operation named `ShowDetails`, which displays customer details.

```
<Context name="OBN_OPERATION_SYSTEM_BO_XML"
objectClass="com.sap.portal.obn.operation" originalLocale="${locale}">
  <Attributes>
    <Attribute name="BOID" type="string">
      <AttributeValue value="businessObject1" />
    </Attribute>
    <Attribute name="SystemAlias" type="string">
      <AttributeValue value="alias1" />
    </Attribute>
    <Attribute name="OperationID" type="string">
      <AttributeValue value="ShowDetails" />
    </Attribute>
    <Attribute name="com.sap.portal.pcm.Title" type="text">
      <AttributeValue value="Customer Details" locale="en" />
    </Attribute>
    <Attribute name="com.sap.portal.pcm.Description" type="text">
      <AttributeValue value="Show customer details" locale="en" />
    </Attribute>
    <Attribute name="com.sap.portal.unification.semanticlayer.
                              usloperation.priority" type="string">
      <AttributeValue value="10"/>
    </Attribute>
  </Attributes>
</Context>
```

4.1.7 Creating Page Layouts

The following creates a page layout called `fullWidth` and assigns it to the page `Portal Information`.

```
<Context name="${namespace}.portal_information"
template="portal_content/com.sap.pct/admin.templates/pages/${namespace}.p
ortalpagetemplate" objectClass="com.sapportals.portal.page" create_as="1"
title="Portal Information">

    <Context name="${namespace}.fullWidth"
    template="portal_content/com.sap.pct/admin.templates/
    layouts/${namespace}.fullWidth"
    objectClass="com.sapportals.portal.layout" create_as="1"
    PrimaryLayout="true"/>
    </Context>

</Context>
```

4.1.8 Creating Pages

The following creates a page named `Portal Information`.

```
<Context name="${namespace}.portal_information"
template="portal_content/com.sap.pct/admin.templates/pages/${namespace}.p
ortalpagetemplate" objectClass="com.sapportals.portal.page" create_as="1"
title="Portal Information"/>
```

4.1.8.1 Assigning iViews to a Portal Page

The following creates an iView named `User Data Import` in a page named `User Data`.

```
<Context name="${namespace}.batchUpload"
template="portal_content/com.sap.pct/admin.templates/pages/${namespace}.p
ortalpagetemplate" objectClass="com.sapportals.portal.page" create_as="1"
title="User Data">

    <Context name="${namespace}.batchUpload"
    template="portal_content/com.sap.pct/admin.templates/iviews/
    ${namespace}.batchUpload"
    objectClass="com.sapportals.portal.iview" create_as="1"
    title="User Data Import" />

</Context>
```

4.1.8.2 Creating Related Items

In addition to the basic attributes required by the `Context` element, the following attributes are used to create a related item:

Attribute	Mandatory	Description and Valid Values
`relatedItem`	Yes	Set to **`true`** to specify that the object is a related item
`relatedItemType`	Yes	Specifies the type of the related item. The following are valid values: • **`dynamicNavigation:`** Creates a Dynamic Navigation iView • **`relatedLinks:`** Creates a Related Item link • **`targetComponents:`** Creates a Drag&Relate link

The following creates an iView with the ID `DNiView`, and then creates for this iView a Dynamic Navigation iView based on a delta link of the page at the PCD address `portal_content/myIviews/pages/myPage`.

```
<Context name="DNiView" objectClass="com.sapportals.portal.iview"
template="portal_content/com.sap.pct/admin.templates/iviews/com.sap.porta
l.pageBuilderDefault" create_as="1" >

    <Context name="DN1" objectClass="com.sapportals.portal.page"
    create_as="1" relatedItem="true"
    relatedItemType="dynamicNavigation"
    template="portal_content/myIviews/pages/myPage"/>

</Context>
```

4.1.8.3 Assigning an IView or a Page as an OBN Target

You can assign an iView or a Page as an OBN (object-based navigation) target.

For more information OBN, refer to the documentation in SAP NetWeaver Library at **`help.sap.com/netweaver`** → *SAP NetWeaver by Key Capability* → *People Integration by Key Capability* → *Portal* → *Portal Administration* → *Content Administration* → *Navigation* → *Object-Based Navigation*.

In addition to the basic attributes required by the `Context` element, the following attributes are used to assign an iView or a page as an OBN target.

Attribute	Mandatory	Description and Valid Values
`BOID`	Yes	The (technical) name of the business object under which the operation is created.
`iViewURL`	Yes	The PCD URL of the iView or page.
`OperationID`	Yes	The (technical) name of the operation.
`SystemAlias`	No	Either the alias of an existing system, or any string that can be used to uniquely identify a business object, such as namespace.

The following example assigns the iView `portal_content/Role1/CustomerDetails` as the OBN target for a business object named `alias1.businessObject1` and an operation named `ShowDetails`.

> **Important**
>
> By the time the `operationImplementation` tag is processed, the iView (or page), business object and operation must already exist. This means that either they were created previously in the same XML, or that they already exist in the PCD.

```
<Context name="OBN_Tagging_and_Mapping_test"
objectClass="com.sap.portal.obn.operationImplementation"
originalLocale="${locale}">


<Attributes>
  <Attribute name="BOID" type="string">
    <AttributeValue value="BO1"/>
  </Attribute>
```

```
  <Attribute name="SystemAlias" type="string">
    <AttributeValue value="alias"/>
  </Attribute>
  <Attribute name="OperationID" type="string">
    <AttributeValue value="op1"/>
  </Attribute>
  <Attribute name="iViewURL" type="string">
    <AttributeValue value="portal_content/Role1/obn04tst_OBNTarget"/>
  </Attribute>
</Attributes>
</Context>
```

4.1.9 Creating Role Folders

The following creates a role folder named `Portal`.

```
<Context name="portal" objectClass="com.sapportals.portal.rolefolder"
entryPoint="false" title="Portal"/>
```

4.1.10 Creating Roles

The following creates a role named `Delegated User Admin`.

```
<Context name="${namespace}.delegated_user_admin_role"
objectClass="com.sapportals.portal.role" entryPoint="false"
collection="${collection}" domain="EP" originalLocale="${locale}"
title="Delegated User Admin">

. . .

</Context>
```

4.1.11 Creating Systems

The following creates a new system template, from which administrators can create system objects.

The example is based on the JDBC system template, which is delivered with the portal.

```
<Context name="${namespace}.JDBCConnectorSystem"
objectClass="com.sapportals.portal.system" create_as="0"
noTemplateNeeded="true" parent="${parent}">
  <Attributes>
    <Attribute name="com.sap.portal.pcm.Title" type="text">
      <AttributeValue value="JDBC system" locale="${locale}" />
    </Attribute>
    <Attribute
name="com.sap.portal.reserved.system.ConnectionFactoryClass"
type="string">
      <AttributeValue value="JDBCFactory" />
    </Attribute>
    <Attribute
name="com.sap.portal.reserved.system.ConnectionFactoryClass-
plainDescription" type="text">
      <AttributeValue value="Connection Factory Class" locale="${locale}"
/>
    </Attribute>
    <Attribute name="ComponentType" type="string">
      <AttributeValue value="com.sapportals.portal.system" />
    </Attribute>
    <Attribute name="ComponentType-plainDescription" type="text">
      <AttributeValue value="Component Type" locale="${locale}" />
    </Attribute>
    <Attribute name="ComponentType-administration" type="string">
      <AttributeValue value="DIALOG-READ-ONLY" />
    </Attribute>
    <Attribute name="url" type="string">
      <AttributeValue value="" />
    </Attribute>

...

  </Attributes>
</Context>
```

4.1.12 Creating Translation Worklists

In addition to the basic attributes required by the `Context` element, the following attributes are used to create a translation worklist.

Attribute	Mandatory	Description and Valid Values
`filter`	No	Enables you to supply a JNDI search string to specify a subset of objects in the `root` folders.
`root`	Yes	Specifies the root folders to use for the objects to include in the translation worklist. When specifying more than one folder, use a comma to separate between folders.

The following creates a translation worklist named `Sample Translation Worklist`. The worklist is made up of content from the `portal_content/gc_samples/content_iviews` and `portal_content/gc_samples/systems` folders, as defined in the `root` attribute. The `filter` attribute specifies a subset of objects in the root folders.

```
<Context name="sample_translation_wl"
objectClass="com.sap.portal.pcd.translation.TranslationWorklist"
collection="${collection}" domain="domain" originalLocale="${locale}"
root="pcd:portal_content/gc_samples/content_iviews,
pcd:portal_content/gc_samples/systems"
filter="(|(com.sap.portal.pcd.gl.AtomicName=*)(com.sap.portal.pcd.gl.Obje
ctClass=com.sap.portal.pcd.gl.GlContext))"
title="Sample Translation Worklist">

    <Attributes>
        <Attribute name="com.sap.portal.pcm.Description"
        type="string">
            <AttributeValue value="Translation Worklist"/>
        </Attribute>
    </Attributes>
</Context>
```

4.1.13 Creating Transport Packages

In addition to the basic attributes required by the `Context` element, the following attributes are used to create a transport package.

Attribute	Mandatory	Description and Valid Values
`filter`	No	Enables you to supply a JNDI search string to specify a subset of objects in the root folders.
`resolveReferences`	No	Indicates whether to resolve references of objects in the transport package, to other objects on which they depend, and include the depended-upon objects in the transport package as well. Default value is **`true`**.
`root`	Yes	Specifies the root folders to use for the objects to include in the transport package. When specifying more than one folder, use a comma to separate between folders.
`singleObjects`	No	Enables to define additional single objects that do not reside under the `root` structure, to be added to the transport package. The value of this attribute is a comma-separated list of single objects, such as a role, a workset, a page; (objects that are not DirContext).

The following creates a transport package named `Sample Content Package`.

```
<Context parent="pcd:portal_content/package_tests"
name="sample_package"
objectClass="com.sapportals.portal.transport.TransportPackage"
collection="${collection}" domain="EP11" originalLocale="${locale}"
root="pcd:portal_content/RootFolder"
singleObjects="pcd:portal_content/Folder1/iView1,
pcd:portal_content/Folder2/Role2folders"
filter="(|(com.sap.portal.pcd.gl.AtomicName=*)(com.sap.portal.pcd.
gl.ObjectClass=com.sap.portal.pcd.gl.GlContext))"
title="Sample Content Package"
resolveReferences="true" >

    <Attributes>
        <Attribute name="com.sap.portal.pcm.Description"
        type="string">
            <AttributeValue value="Transport Package"/>
        </Attribute>
    </Attributes>
</Context>
```

4.1.14 Creating Worksets

The following creates a workset named `Company`, and sets the value of the `MergeId` attribute for the workset object.

```
<Context name="${namespace}.home.company"
objectClass="com.sapportals.portal.workset" entryPoint="false"
asUnit="true" title="Company" collection="${collection}" domain="EP"
originalLocale="${locale}">

    <Attributes>
        <Attribute name="com.sap.portal.navigation.MergeId"
        type="string">
            <AttributeValue value="${namespace}.home.company"/>
        </Attribute>
    </Attributes>

</Context>
```

4.2 Code Samples for Actions

This section contains basic XML code samples for executing actions.

Each heading in this section contains the ID suffix for each action, which should be preceded by `com.sap.portal`. For example, the action ID for configuring proxy settings is `com.sap.portal.proxy`.

4.2.1 Adding/Removing System Aliases (alias.handler)

This action enables you to add and remove a system alias, and to set a default system alias.

In addition to the basic attributes required by the `Action` element, the following attribute is expected by this action:

Attribute	Mandatory	Description
`system`	Yes	The ID of the system to which to modify its aliases

The following adds aliases `a1`, `a2`, `a3` and `a4`, sets the default alias to `a3`, and deletes aliases `d1` and `d2`, for a system whose PCD address is `portal_content/samples/mySystem`.

```
<Action id="com.sap.portal.alias.handler"
system="portal_content/samples/mySystem">
    <Attributes>
        <Attribute name="addAlias">
            <AttributeValue value="a1"/>
            <AttributeValue value="a2"/>
            <AttributeValue value="a3"/>
            <AttributeValue value="a4"/>
        </Attribute>
        <Attribute name="changeDefaultAlias">
            <AttributeValue value="a3"/>
        </Attribute>
        <Attribute name="removeAlias">
            <AttributeValue value="d1"/>
            <AttributeValue value="d2"/>
        </Attribute>
     </Attributes>
</Action>
```

4.2.2 Assigning Users/Groups to Roles (roleassignment)

This action enables you to assign a user or group to a role.

Within the `Action` tag, specify the roles to which you want to assign users and groups, using a `Roles` tag. Within the `Roles` tag, use a `Role` tag for each role, and specify the users and groups to assign to the role.

> **Note**
>
> Specify a role by its PCD address, starting with the `pcd:` prefix. If only the role name is specified, the first role with the name is selected.

The following adds `user1`, `user2` and `group1` to the roles `role1` and `role2`.

```
<Action id="com.sap.portal.roleassignment">
    <Roles>
        <Role name="role1">
            <Principal type="user" id="user1"/>
            <Principal type="group" id="group1"/>
            <Principal type="user" id="group2"/>
        </Role>
        <Role name="role2">
            <Principal type="user" id="user1"/>
            <Principal type="group" id="group1"/>
            <Principal type="user" id="user2"/>
        </Role>
    </Roles>
</Action>
```

4.2.3 Configuring Proxy Settings (proxy)

This action configures the portal's proxy settings.

The proxy settings configured by this action are the same as can be configured in the portal's `com.sap.portal.ivs.httpservice.proxy` service.

> **CAUTION**
>
> Set the proxy service's *updateSettings* property to **`false`** using the Service Configuration tool in the portal. Otherwise, the settings defined in the XML file will be lost the next time the service is restarted.

In addition to the basic attributes required by the `Action` element, the following attributes are expected by this action:

Attribute	Mandatory	Description
`CreateMode`	Yes	Indicates whether to set to null all attributes that are not specified. The following are valid values: **1:** Sets to null any attributes that are not specified **3:** Does not change any attributes that are not specified
`firewallHost`	No	The name or IP address of the firewall server.
`firewallPort`	No	The port of the firewall server.
`firewallSet`	No	Enables use of the firewall for all requests
`ftp.bypass`	No	For FTP requests, the hosts to which to connect directly and not via the proxy server. For more information, see `http.bypass`.
`ftp.host`	No	The name or IP address of the proxy server for FTP requests.
`ftp.port`	No	The port of the proxy server for FTP requests.
`ftp.set`	No	Enables use of the proxy server for FTP requests, either **`true`** or **`false`**
`http.bypass`	No	For HTTP requests, the hosts to which to connect directly and not via the proxy server. The value can be a list of hosts separated by any of the following: • `space` • `pipe (|)` • `semi-colon (;)` • `comma (,)` You may use a wildcard character (**`*`**) for matching. Example: **`*.goofy.sap.com|*.mickey.sap.com;*.donald.sap.com`**

`http.host`	No	The name or IP address of the proxy server for HTTP requests.
`http.port`	No	The port of the proxy server for HTTP requests.
`http.set`	No	Enables use of the proxy server for HTTP requests, either **`true`** or **`false`**
`https.bypass`	No	For HTTPS requests, the hosts to which to connect directly and not via the proxy server. For more information, see `http.bypass`.
`https.host`	No	The name or IP address of the proxy server for HTTPS requests.
`https.port`	No	The port of the proxy server for HTTPS requests.
`https.set`	No	Enables use of the proxy server for HTTPS requests, either **`true`** or **`false`**
`proxyPassword`	No	The password for proxy basic authentication.
`proxyUser`	No	The user name for proxy basic authentication.

The following sets the portal's proxy settings:

```
<Action id ="com.sap.portal.proxy"
    CreateMode ="3"
    http.set ="true"
    http.bypass="*.tlv.sap.corp|*.dhcp.tlv.sap.corp"
    http.host ="proxy"
    http.port ="8080"/>
```

4.2.4 Copying Content (copy)

This action enables you to copy content from one folder to another.

The following copies `portal_content/iview1` and `portal_content/page2` to the folder `portal_content/target1`, and copies `portal_content/role25` and `portal_content/workset26` to folder `portal_content/target2`:

```
<Action id="com.sap.portal.copy">
    <Attributes>
        <Attribute name="portal_content/target1" >
            <AttributeValue value="portal_content/iview1" />
            <AttributeValue value="portal_content/page2" />
        </Attribute>
        <Attribute name="portal_content/target2" >
            <AttributeValue value="portal_content/role25" />
            <AttributeValue value="portal_content/workset26" />
        </Attribute>
    </Attributes>
</Action>
```

4.2.5 Deleting Content (gc.deepCleaner)

This action enables you to delete content in the PCD. You specify the start folder and the action is performed recursively on all subfolders. You can also specify content to exclude from the deletion.

This action is different to the `clean` mode execution specified in `Context` elements. Whereas the `clean` mode only deals with objects specified in the XML, the deep clean action is performed on any semantic object located in the specified folder.

> **CAUTION**
>
> It is advised to use this action with extreme caution. In some instances the `DeepCleaner` may unknowingly delete PCD data that is not within the specified folder.

In addition to the basic attributes required by the `Action` element, the following attributes are expected by this action:

Attribute	Mandatory	Description
`exclude.folder`	No	The ID of the folder which the deep cleaner must ignore. This must be a folder within the hierarchy of the `root.folder` attribute. You must enter an absolute path; in other words, do not enter an ID that is relative to the `root.folder` attribute. You cannot enter more than one folder to exclude. The exclusion is recursive from the specified folder onward.
`root.folder`	Yes	The ID of the folder from which to start the deep clean process

The following deletes all content in the `portal_content/test` folder except for the content in the `portal_content/test/myFolder` folder.

```
<Action id="com.sap.portal.gc.deepCleaner" ignore="false"
root.folder="pcd:portal_content/test"
exclude.folder="pcd:portal_content/test/myFolder" />
```

4.2.6 Mirroring Content (mirror)

This action enables you to mirror content from one folder to another.

In addition to the basic attributes required by the `Action` element, the following attributes are expected by this action:

Attribute	Mandatory	Description
`object.types`	No	A comma-separated list of semantic object types to mirror. If the attribute is not provided, all supported object types are mirrored.
`objects.prefix`	No	The prefix for all copied objects.
`source.path`	Yes	The folder that contains the objects to mirror.
`target.path`	Yes	The folder to which to mirror the objects.

The following mirrors all role, workset, page and system objects in `pcd:portal_content/source2` to folder `pcd:portal_content/target2` and adds the prefix `sap.xyz.com` to the mirrored objects:

```
<Action id="com.sap.portal.mirror"
    source.path="pcd:portal_content/source2"
    target.path="pcd:portal_content/target2"
    objects.prefix="sap.xyz.com."
    object.types="role, workset, page, system"/>
```

4.2.7 Running Another Script (script.runner)

This action enables you to run another XML script from within an XML script. You can run the script several times, in a loop, and set generic properties to the script.

In addition to the basic attributes required by the `Action` element, the following attributes are expected by this action:

Attribute	Mandatory	Description
`file.name`	Yes	The full path of the script to run. The script must be located on the portal server.
`loop`	Yes	The number of times to run the script.

You can set general properties for all iterations of the script by supplying an attribute named `external.properties`.

You can also set general properties so that a property has a different value for each iteration of the script. You can define these properties in an attribute named `loop.properties`. For each property defined within this attribute, supply a value for each iteration of the script. The number of values for each property must equal the value defined for the `loop` attribute in the `action` tag.

The following sample starts running a script whose path is `c:\usr\myScript.xml`. The script is run twice, with the property `myProperty1` set to `25` on the first iteration and `50` on the second iteration, and the property `myProperty2` set to `10` on the first iteration and `20` on the second iteration. For both iterations, the property `myGeneralProperty1` is set to `Mike` and `myGeneralProperty2` is set to `Joe`.

```
<Action id="com.sap.portal.script.runner" file.name="c:\usr\myScript.xml"
loop="2">
    <Attributes>
        <Attribute name="external.properties">
            <Attribute name="myGeneralProperty1">
                <AttributeValue value="Mike"/>
            </Attribute>
            <Attribute name="myGeneralProperty2">
                <AttributeValue value="Joe"/>
            </Attribute>
        </Attribute>
        <Attribute name="loop.properties">
            <Attribute name="myProperty1">
                <AttributeValue value="25" />
                <AttributeValue value="50"/>
            </Attribute>
            <Attribute name=" myProperty2">
                <AttributeValue value="10"/>
                <AttributeValue value="20"/>
            </Attribute>
        </Attribute>
    </Attributes>
</Action>
```

4.2.8 Setting Permissions

This action enables you to set the permissions for any portal object by replacing the ACL (access control list) for the portal object. The ACL is a collection of ACEs (access control entries), which define specific permissions for specific users, groups or roles.

> **Note**
>
> This action does not use the `Action` element and has its own element and syntax.

To set permissions, use the following elements:

- **ACL:** Create an ACL element for each portal object whose permissions you want to set. The element takes the following attributes:

Attribute	Mandatory	Description
`handlerId`	Yes	Always set to `ACL`
`objectID`	Yes	The PCD address of the portal object whose permissions you want to set

Within each `ACL` element, nest an `ACEs` element.

- **ACEs:** Nest an `ACEs` element inside an `ACL` element. Within the `ACEs` element, nest one or more `ACE` elements for the current portal object.
- **ACE:** Nest one or more `ACE` elements inside an `ACEs` element for each ACE that you want to create for the current portal object. The element takes the following attributes:

Attribute	Mandatory	Description
`endUserRead`	No	Indicates whether the principal gets end user permission. Valid values are `true` or `false` (default).
`permission`	Yes	The permission to grant to the principal specified by the `principalID` attribute The following are valid values: • **NONE:** No administration permission. • **Pcd.Read:** The principal can read the object. • **Pcd.ReadWrite:** The principal can read and change the object. • **Pcd.FullControl:** The principal can read, change and delete the object. • **Owner:** The principal can read, change and delete the object, and change the permissions of the object.
`principalID`	Yes	The principal receiving the permission
`roleAssign`	No	Indicates whether the principal gets role assigner permission. Valid values are `true` or `false` (default). This permission can only be assigned for role objects, and folders containing role objects that inherit permissions from the folder.
`type`	Yes	The type of principal specified by the `principalID` attribute, either `user`, `group` or `role`

The following example assigns permissions to the portal object with the PCD address `pcd:portal_content`:

```
<ACL objectID="pcd:portal_content" handlerId="ACL">
    <ACEs>
        <ACE type="role"
            principalID="pcd:portal_content/administrator/
content_admin/content_admin_role"
            permission="Pcd.FullControl"
            endUserRead="true" />
        <ACE type="group"
            principalID="GRUP.SUPER_GROUPS_DATASOURCE.EVERYONE"
            permission="NONE"
            endUserRead="true"
            roleAssign="true" />
        <ACE type="role"
            principalID="pcd:portal_content/administrator
/super_admin/super_admin_role"
            permission="owner"
            endUserRead="true"
            roleAssign="true" />
        <ACE type="role"
            principalID="pcd:portal_content/administrator
/system_admin/system_admin_role"
            permission="Pcd.ReadWrite"
            endUserRead="true" />
    </ACEs>
</ACL>
```

Exporting Permissions

You can export the existing permissions of objects in a portal to an XML file, and then import the XML file to set those permissions in another portal.

More Information

For more information on portal permissions, refer to the documentation in SAP NetWeaver Library at **`help.sap.com/netweaver`** → *Functional View* → *SAP NetWeaver by Key Capability* → *People Integration by Key Capability* → *Portal* → *Portal Administration Guide* → *System Administration* → *Permissions*, *Role/User Distribution*, *and Object Locking* → *Portal Permissions*:

- *Transporting Permissions*
- *Using the Permission Editor*

4.3 Tips and Tricks

4.3.1 General Tips

- Avoid using special characters in the object ID (`name`) of content objects.
- Use the correct data types and locale for properties in content objects.
- Angle brackets (< >) are reserved for XML script. If you need to use them elsewhere, use **`&lt`** and **`&gt`**.

4.3.2 Executing Specific XML Blocks

If you want to re-use an XML script to make pinpoint changes to content that has already been created using the file, apply the `ignore` attribute to skip all blocks in the XML document except for those you want to execute. The `ignore` attribute is applied to each block within the block that defines it, unless specified otherwise.

For example, configure the XML document as follows:

1. Set the `ignore` value to **`true`** in the `GenericCreator` root element.
2. For all blocks to be executed, insert the `ignore` attribute and set it to **`false`**.
3. In all child blocks that should be skipped, make sure they do not declare the `ignore` attribute. If they do, set the value to **`true`**.

This procedure supports both `execute` and `clean` modes.

4.3.3 Creating Hierarchies Without Nested Elements

Typically, you nest one `Context` element within another to generate object hierarchies in the PCD. However, you can also create a hierarchy by using the `parent` attribute in the `Context` element. The attribute specifies the ID and path of the parent folder for the defined object. Thus, the `parent` attribute enables you to improve the readability of the XML file, by using a single `Context` element for an object, instead of nested elements.

> **Note**
>
> The `parent` attribute can only be used in a root `Context` element. You cannot use it in a `Context` element that is nested in another `Context` element.

The following shows the use of the `parent` attribute to create a nested iView:

```
<Context name="${namespace}.portletProxyIview"
parent="portal_content/com.sap.pct/templates/iviews" ignore="false"
template="par:/applications/com.sap.portal.ivs.wsrpservice/components/Pro
xyPortalComponent" objectClass="com.sapportals.portal.iview"
create_as="0" collection="${collection}" domain="EP"
originalLocale="${locale}" title="Portlet Proxy iView" />
```

5. Exporting/Importing Content and Actions

The portal provides the following XML Content and Actions tools:

- **Export Tool:** Enables you to create an XML file based on existing content, as described in *Exporting Content* on page 45. This file can be edited and imported into a portal (via the import tool) in order to create content.
- **Import Tool:** Enables you to import an XML file in order to create content and perform actions, as described in *Importing Content and Actions* on page 48.

By default, these tools are assigned to the standard system administration role, and can be accessed by navigating to *System Administration* → *Transport* → *XML Content and Actions*.

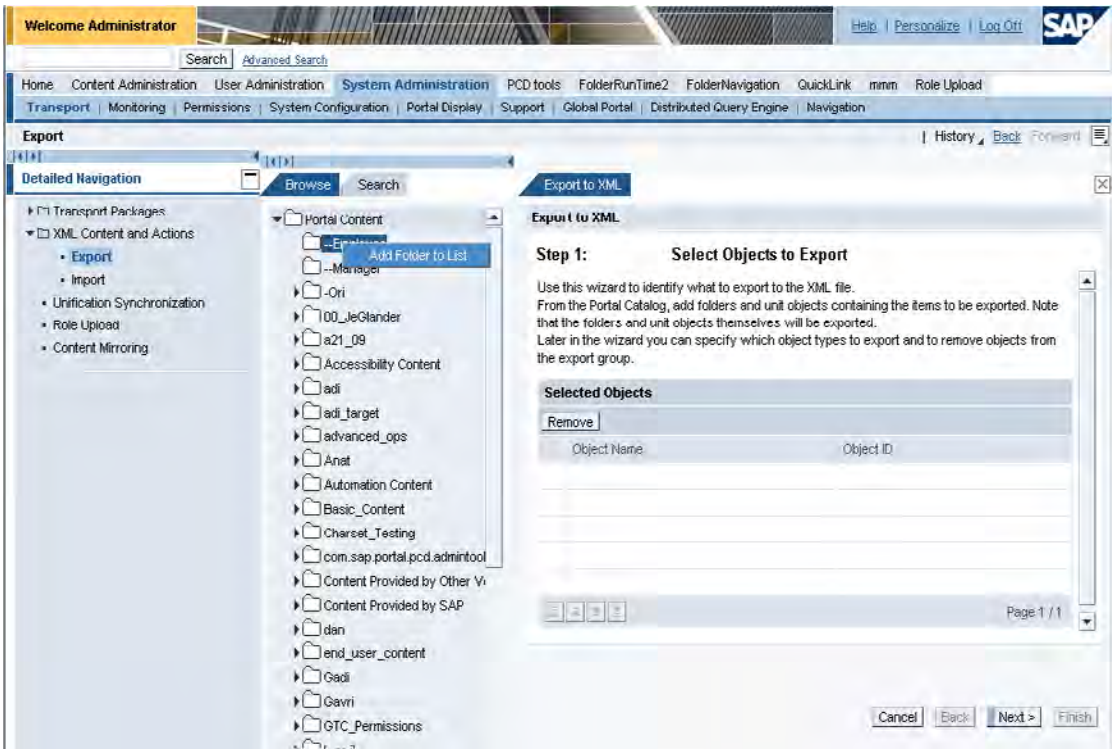
> **CAUTION**
>
> Imported XML files can execute any number of actions in the portal, including overwriting and deleting existing content. Running an incorrect XML file may cause permanent damage to the portal. It is highly recommended to perform test runs initially on a non-production portal or on test content before using it in a live environment.
>
> It is recommended to restrict access to the iView to administrators trained to use it.

5.1 Exporting Content

The XML Content and Actions export tool automatically creates an XML file based on existing portal content. This file can be edited and imported into a portal in order to create content.

Procedure

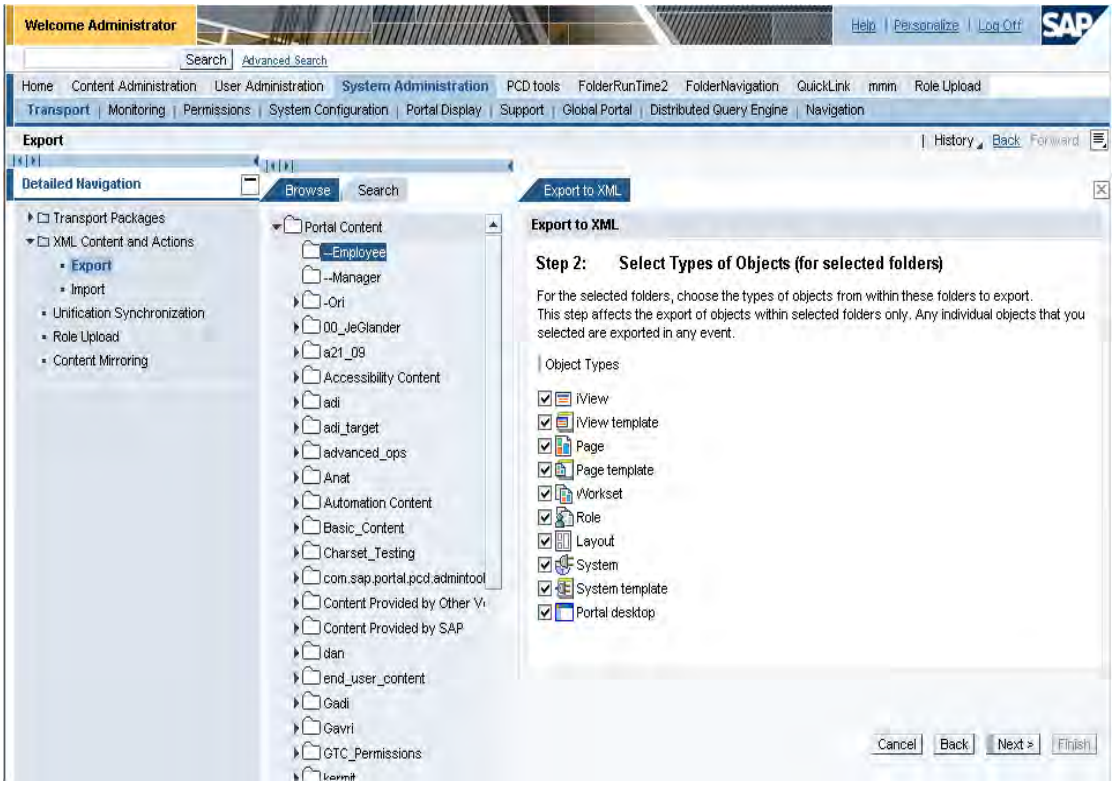
1. In the portal, navigate to *System Administration* → *Transport* → *XML Content and Actions* → Export.



2. Select the content from which to generate the XML by right-clicking each object in the Portal Catalog and clicking *Add <Object> to List*.
3. Click *Next*.

4. Select the type of objects to export.

   This step is displayed only when at least one folder was selected in the previous step. This step determines the types of objects that are exported from within the selected folders.



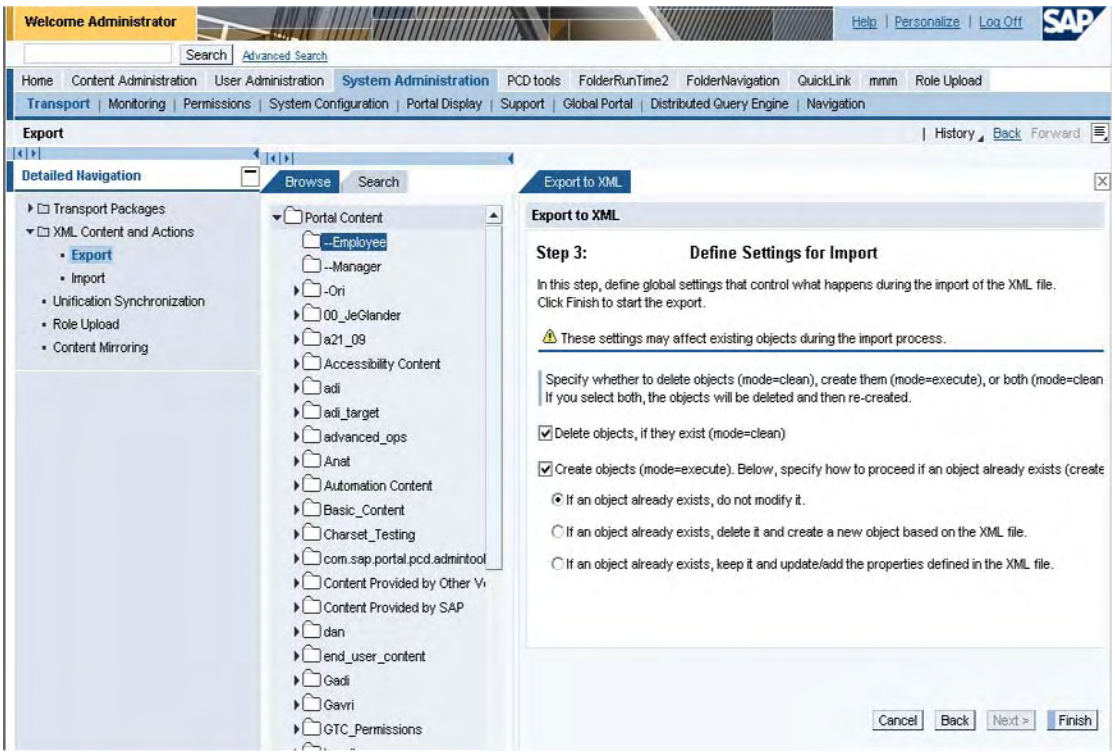
Important

This step does not affect whether the tool exports the individual portal objects (that is, non-folder objects) that were selected in the previous step.

For example, if you select in this step only iViews, only iViews from within selected folder are exported. All the individual iViews, pages, roles and other objects that were selected in the previous step are exported in any event.

5. Click *Next*.

6. Select the settings to be used when the exported XML file is imported into a portal. This step affects XML attributes in the exported file.



7. Click Finish.

   The XML file is created, and a link to the file (**`Open XML`**) is displayed. Click the link to view the file.

Location of XML Files

Copies of the XML files created with the export tool are placed in the folder `portalapps\com.sap.portal.content.export\xml` under the portal root folder.

These files can be large. System administrations should clean out this folder from time to time.

5.2 Importing Content and Actions

The XML Content and Actions import tool enables you to import an XML file for creating content and performing actions.

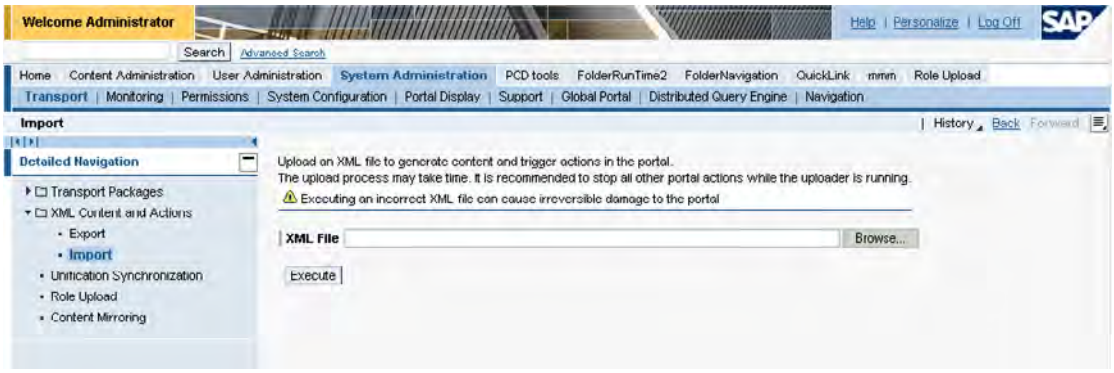
Prerequisites

A well-formed XML file that is valid for the XML Content and Actions feature.

> **Important**
>
> It is highly recommended to stop all other portal actions while the import tool is running.

Procedure

1. In the portal, navigate to *System Administration* → *Transport* → *XML Content and Actions* → *Import*.



2. Click *Browse*.
3. Locate and select an XML file to import.

> **CAUTION**
>
> Once you click *Execute* to import the file, you cannot stop the import process. All actions are irreversible and there is no rollback feature in case the process aborts in the middle.

4. Click *Execute* to begin the import process. With large files, the process may be time consuming.

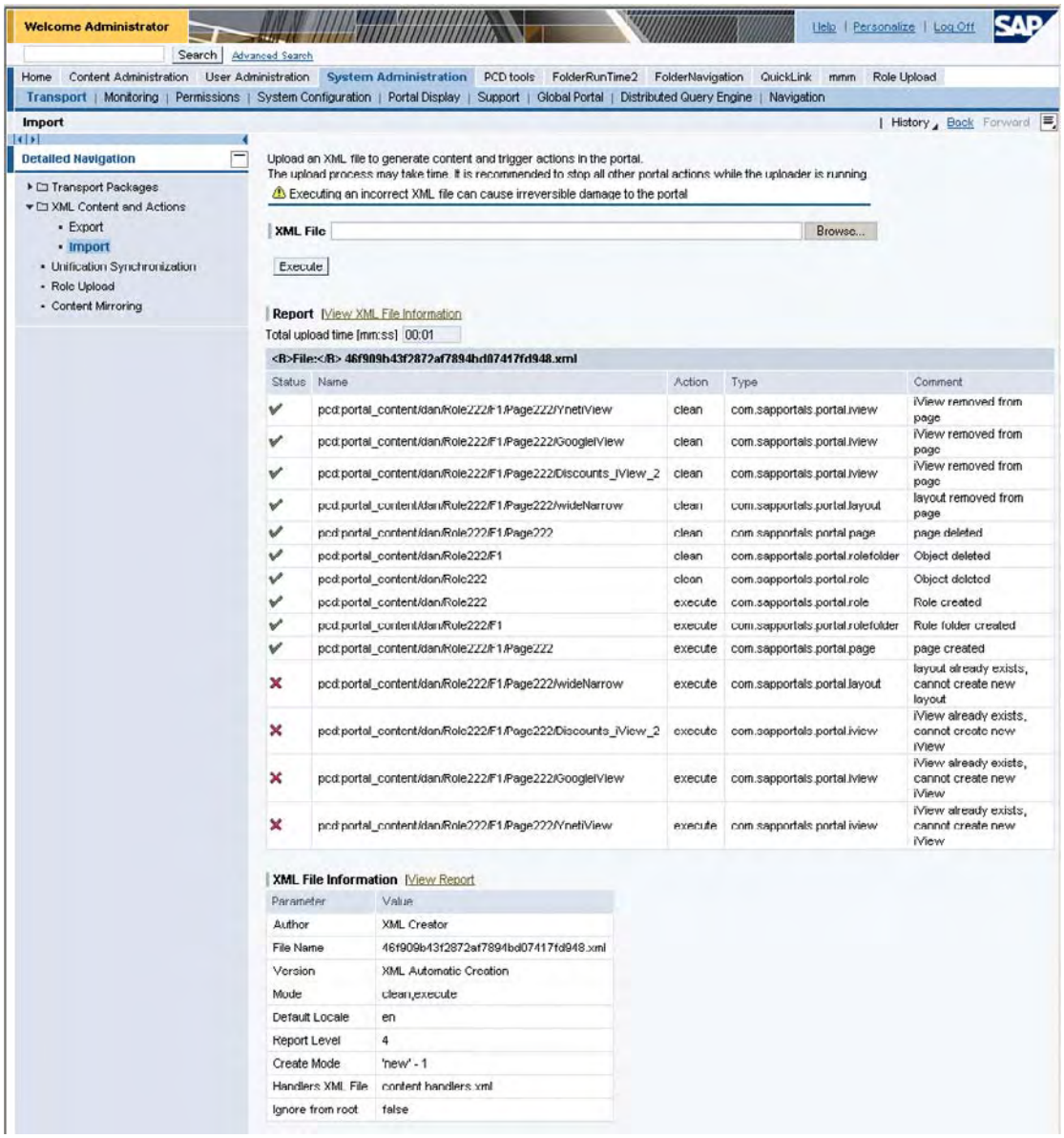
> **CAUTION**
>
> Do not perform any actions in the portal during the import process.

Result

When the script finishes, the results are displayed in the following tables:

- **Report:** Describes how each `<Context>` and `<Action>` element was handled; for example, whether a new portal object was created.
- **XML File Information:** Provides basic information about the uploaded file and global settings.

A link (*View …*) is provided next to each table heading to enable moving directly to the other table.


Welcome Administrator
Help | Personalize | Log Off
Search
Advanced Search
Home
Content Administration
User Administration
System Administration
PCD tools
FolderRunTime2
FolderNavigation
QuickLink
mmm
Role Upload
Transport | Monitoring | Permissions | System Configuration | Portal Display | Support | Global Portal | Distributed Query Engine | Navigation
Import
History Back Forward
Detailed Navigation
Transport Packages
XML Content and Actions
Export
Import
Unification Synchronization
Role Upload
Content Mirroring
Upload an XML file to generate content and trigger actions in the portal.
The upload process may take time. It is recommended to stop all other portal actions while the uploader is running
Executing an incorrect XML file can cause irreversible damage to the portal
XML File
Browse...
Execute
Report View XML File Information
Total upload time [mm:ss] 00:01
<R>File:</R> 46f909b43f2872af7894bd07417fd948.xml
Status | Name | Action | Type | Comment
pcd:portal_content/dan/Role222/F1/Page222/YnetiView | clean | com.sapportals.portal.iview | iView removed from page
pcd:portal_content/dan/Role222/F1/Page222/GoogleiView | clean | com.sapportals.portal.iview | iView removed from page
pcd:portal_content/dan/Role222/F1/Page222/Discounts_iView_2 | clean | com.sapportals.portal.iview | iView removed from page
pcd:portal_content/dan/Role222/F1/Page222/wideNarrow | clean | com.sapportals.portal.layout | layout removed from page
pcd:portal_content/dan/Role222/F1/Page222 | clean | com.sapportals.portal.page | page deleted
pcd:portal_content/dan/Role222/F1 | clean | com.sapportals.portal.rolefolder | Object deleted
pcd:portal_content/dan/Role222 | clean | com.sapportals.portal.role | Object deleted
pcd:portal_content/dan/Role222 | execute | com.sapportals.portal.role | Role created
pcd:portal_content/dan/Role222/F1 | execute | com.sapportals.portal.rolefolder | Role folder created
pcd:portal_content/dan/Role222/F1/Page222 | execute | com.sapportals.portal.page | page created
pcd:portal_content/dan/Role222/F1/Page222/wideNarrow | execute | com.sapportals.portal.layout | layout already exists, cannot create new layout
pcd:portal_content/dan/Role222/F1/Page222/Discounts_iView_2 | execute | com.sapportals.portal.iview | iView already exists, cannot create new iView
pcd:portal_content/dan/Role222/F1/Page222/GoogleiView | execute | com.sapportals.portal.iview | iView already exists, cannot create new iView
pcd:portal_content/dan/Role222/F1/Page222/YnetiView | execute | com.sapportals.portal.iview | iView already exists, cannot create new iView
XML File Information View Report
Parameter | Value
Author | XML Creator
File Name | 46f909b43f2872af7894bd07417fd948.xml
Version | XML Automatic Creation
Mode | clean,execute
Default Locale | en
Report Level | 4
Create Mode | 'new' - 1
Handlers XML File | content handlers.xml
Ignore from root | false

The *Report* table contains the following fields:

Field	Description
`Status`	Indicates whether the action was successful. The status can be one of the following: 1. `debug` 2. `info` 3. `warning` 4. `success` 5. `fail` Results are filtered based on each action's status and the report level filter defined in the `report.level` attribute of the `GenericCreator` node. Results are displayed for actions with the selected report level and higher. For example, if the report level is `warning`, then actions with a status of `warning`, `success` and `fail` are also displayed. If `debug` is defined, then all result types are displayed.
`Name`	Specifies the full path and name of the object that was created or modified.
`Action`	Specifies the operation mode as defined in the `mode` attribute of the root element. The value can be one of the following: • **execute:** When a semantic object is created in the PCD, or when an action is performed • **clean:** When a semantic object is deleted in the PCD
`Type`	Specifies the object's class, as specified in the XML.
`Comments`	Provides a summary of the action performed.

Appendix A APIs

The portal provides APIs for the XML Content and Actions feature for doing the following:

- **Running an XML Script:** You can run an XML script from within Java code, without using the administration user interfaces.
- **Developing Handlers:** You can develop your own handlers for easily performing routine tasks.

The APIs are described in *Automating Content Creation with XML* in the *Developing Applications for the Portal* section of the SAP NetWeaver Developer Studio documentation.

www.sdn.sap.com/irj/sdn/howtoguides