Web Content Manager Version 8.0

Version 8 Release 0
## Contents

### Chapter 1. Overview ............................................. 1
   IBM Web Content Manager .................................... 1
   IBM Web Content Manager new features and improvements ... 3
   Supported specifications for inserting links to remote content .. 5
   Documentation resources .................................... 5
   Accessibility features ...................................... 7
   Types of websites .......................................... 7
   Intranet portal ............................................ 8
   E-business site ........................................... 9
   Brochureware site ......................................... 11
   E-library site ............................................ 12
   Partner site .............................................. 13
   Conceptual and functional divisions of a website .......... 15

### Chapter 2. Planning a website .................................. 19
   Defining the project ....................................... 19
   Human resource planning .................................. 21
      The project manager .................................. 21
      The business analyst .................................. 22
      The architecture and design team .................... 23
      The deployment team .................................. 27
      The development team ................................ 30
      The website creation team ............................ 32
      The content acquisition team ......................... 34
      The maintenance team ................................ 34
   Creating an analysis document ........................... 35
   Designing a prototype website using HTML .......... 37
   Creating a design document ............................... 39
      Server architecture .................................. 39
      Security architecture ................................ 39
      Information architecture ............................. 43
      Design architecture .................................. 44
      Authoring architecture ............................... 45
      Content acquisition architecture ................... 48
      Delivery architecture ................................ 49
      Maintenance architecture ............................. 50
   Road map to building a web content system .......... 51
      Deploying the authoring environment ............... 51
      Building the content authoring system ............ 52
      Importing and creating content ...................... 53
      Deploying the delivery environment .................. 54
      Final steps ........................................... 54

### Chapter 3. Installing and migrating ............................ 57

### Chapter 4. Configuring Web Content Manager .................. 59
   Configuring a web content authoring environment .... 59
      Installing the authoring portlet ..................... 59
   Additional authoring portlet configuration options .... 61
   Web content authoring options .......................... 62
   Configuring authoring portlet search ................... 66
   Importing large files and images ........................ 67
   Increasing time-outs ...................................... 69
   Configuring remote server access for links ............ 70
   Setting up support for federated documents .......... 71
   Configuring a web content staging environment ...... 76
   Configuring a web content delivery environment ...... 76
      Setting up site analysis for the web content viewer ... 76
   XML configuration interface parameters for the web content viewer .... 80
   Caching options ........................................ 83
   Pre-rendering options ................................ 89
   Disabling the site toolbar on a delivery server .... 93
   Reserving authoring portlet ............................ 94
   Configuring the reserved authoring portlet .......... 95
   Additional configuration options ....................... 96
      Controlling access to hosts specified in a URL .... 96
      Web content substitution variables ................. 97
      Enabling connect tags ................................ 98
      Remove authoring configuration task ................ 99
      Enabling email ....................................... 99
      Configuring managed pages ........................... 100
      Enabling managed pages .............................. 101
      Disabling managed pages ............................. 103
      Transferring content associations to the Portal Site library ................. 104
      Syndication properties .............................. 106
      Enabling search for web content ..................... 107
         Indexing web content .............................. 107
      Configuring Web Content Manager search options ... 109
      Configuring Search Center to search for web content .......................................................... 111
      Crawling web content with search seedlists ....... 111
      Managing tagging and rating for web content ........ 117
         Using tagging and rating scopes with web content ......................................................... 117
         Synchronizing scopes for web content ............ 118

### Chapter 5. Setting up a site .................................... 121
   Site toolbar ............................................ 121
   Previewing as another user ............................... 123
   Libraries .................................................. 124
      Web content libraries ................................ 124
      Web content library default items .................. 125
   Pages ...................................................... 126
      Page creation and navigation ......................... 127
      Creating a page from the site toolbar ............... 128
      Page templates ....................................... 129
      Creating page from a template using the site toolbar ...................................................... 130
      Page layout and style ................................ 130
      Changing page style .................................. 130
      Changing page layout ................................ 131
      Moving a page ........................................ 131
Comparison of features support by types of portal pages ........................................... 131
Projects and workflows ........................................... 132
Projects overview ........................................... 133
Workflow and change management ........................................... 135
Best practices for projects ........................................... 151
Managed pages ........................................... 152
WebSphere Portal artifacts affected by managed pages ........................................... 152
Drafts and projects for changes to managed pages ........................................... 153
Scope of edits for managed pages ........................................... 154
Edit mode and administration pages ........................................... 155
Managed pages and site management ........................................... 155
Managed pages and the authoring portlet ........................................... 155
System content associations ........................................... 156
Best practices for managed pages ........................................... 156
Troubleshooting managed pages ........................................... 157
Known issues for managed pages ........................................... 158
Administering managed pages ........................................... 160
Web content items ........................................... 173
Creating authoring templates ........................................... 173
Building a site framework ........................................... 180
Presentation templates ........................................... 184
Template maps ........................................... 188
Content items ........................................... 189
Components ........................................... 190
Profiling strategies ........................................... 191
Elements ........................................... 193
Creating links and navigation ........................................... 193
Using a search element ........................................... 201
Storing text and HTML ........................................... 205
Storing files and images ........................................... 207
Selection elements ........................................... 211
Personalized content ........................................... 212
Using a page navigation element ........................................... 219
URL generation using PathCmpnt and URLCmpnt tags ........................................... 222
Personalizing federated documents ........................................... 222
Inserting a link to remote content ........................................... 227
Using start and end attributes ........................................... 229
Setting parameters to format dates ........................................... 230
Note on formatting numbers ........................................... 232
Tags ........................................... 232
Web content tags ........................................... 232
Indenting element designs ........................................... 233
Writing links to Web content ........................................... 234
Contextual linking ........................................... 236
Using custom caching ........................................... 236
Rendering plug-ins provided with Web Content Manager ........................................... 241
Access control ........................................... 251
Users, Groups and Roles ........................................... 252
Authoring interface ........................................... 262
Custom portal pages for authoring ........................................... 262
Authoring system access strategies ........................................... 263
Authoring portlet customization ........................................... 263
Custom authoring interfaces ........................................... 268
Web content inline editing strategies ........................................... 269
Content ........................................... 273
IBM Web Content Integrator ........................................... 273
WebDAV ........................................... 321
Blogs ........................................... 332
Wikis ........................................... 336
Item management ........................................... 339
Working with folders ........................................... 339
Managing versions of items ........................................... 340
Locked and draft items ........................................... 341
Deleting an item ........................................... 342
Viewing item references ........................................... 343
Personalization ........................................... 343
How a site is personalized ........................................... 345
Personalization terms ........................................... 347
The Personalization interface ........................................... 371
Publishing personalization rules overview ........................................... 373
The Web Content resource collection ........................................... 383
The Portal User resource collection ........................................... 386
LikeMinds Recommendations ........................................... 388
Feedback and analytics ........................................... 417
Developing a personalized portlet ........................................... 455
Personalization programming reference ........................................... 512

Chapter 6. Delivering web content ........................................... 529
Accessing web content via a servlet ........................................... 529
Delivering web content on a portal page ........................................... 530
Getting started with delivering web content on a portal page ........................................... 531
Displaying content with web content viewers ........................................... 547
Customizing web content delivery ........................................... 549
Performing remote rendering with WSRP and the web content viewer ........................................... 565
Advanced administrative examples ........................................... 566
Web content viewer best practices and limitations ........................................... 577
Pre-rendered delivery ........................................... 578
Pre-rendering a website ........................................... 579
Accessing the pre-rendered site ........................................... 580

Chapter 7. Maintaining web content ........................................... 583
Using the web content member fixer task ........................................... 583
Member fixer with syndication ........................................... 589
Using the Update Security task ........................................... 590
Using the workflow update tool ........................................... 593
Clearing item history ........................................... 594
Clearing version history ........................................... 596
Resetting the web content event log ........................................... 597
Using the export cache settings task ........................................... 598
Exporting and importing web content libraries ........................................... 599
Exporting and importing a web content library ........................................... 601
Exporting and importing a web content library in a z/OS environment ........................................... 605
Exporting and importing a web content library copy ........................................... 607
Cloning a web content repository ........................................... 613
Cloning preparation ........................................... 613
Cloning data ........................................... 614

Chapter 8. Developing ........................................... 617
The IBM Web Content Manager API ........................................... 617
Using the API ........................................... 617
Web Content Manager JSP tags ........................................... 618
Chapter 1. Overview

IBM® Web Content Manager is included with IBM WebSphere® Portal to provide a robust web content management solution. It includes an authoring interface that can be customized and extended, workflow, versioning, taxonomy, and much more.

**IBM Web Content Manager**

Web Content Manager is used to create, manage, and deliver content for your website. You can create content using the Web content authoring portlet, or create your own customized authoring interface. Web content stored in external content management systems can also be referenced within a Web Content Manager system. You can deliver your web content using Web content viewer portlets, the Web Content Manager servlet or pre-render your site to HTML.

**Website creation**

Before setting up your web content system, you should read the entire WebSphere Portal Planning and Installing sections to determine the best overall configuration for your WebSphere Portal system.

You should also review the [WebSphere Portal Family Wiki](http://www-01.ibm.com/support/docview.wss?uid=swg27029110). This site covers a wide variety of topics that product experts have found critical to a successful implementation.

In a Web Content Manager solution, the layout and design elements of a website are managed separately from the content of a website. This allows you to change the layout and design of a web page without changing the content, and change the content without having to update the layout and design. Many design features of your website, such as navigation, are generated automatically by using predefined elements and components.

**Web content authoring**

One of the primary uses of Web Content Manager is to build a Web content authoring system for your web content creators. In a Web Content Manager system, the design and layout of your website are separated from the content displayed within a website. This allows web content authors to create the content for your site without having to understand how to build a website. The web content authoring systems you create are designed to deliver different web content authoring experiences to different types of users.

- “Authoring interface” on page 262
Accessing external content

The web content used in a Web Content Manager can also be stored and managed in external content management systems. The Web Content Integrator is used to import content into a Web Content Manager system using the RSS feed format. Content from federated content systems can be linked directly into a Web Content Manager system. You can also use WebDAV to import content from a file system into a Web Content Manager system.

- "IBM Web Content Integrator" on page 273
- "WebDAV" on page 321
- "Setting up support for federated documents" on page 71

Managing web content

Web Content Manager includes a set of features to help you maintain and manage your Web Content Manager system. This includes version control, change management and approval, multi-item projects and user-defined folders to group sets of web content items.

- "Item management" on page 339

Preinstalled web content libraries

A set of preinstalled web content libraries are supplied to allow you to add blog and wiki features to your websites. Use blogs, blog libraries, and wikis to tap into the power of the community and to change the way you work.

- Blogs and wikis

Delivering web content

Web content can be delivered to your website viewers using Web Content Viewer portlets, a servlet or as pre-rendered HTML files.

- Chapter 6, “Delivering web content,” on page 529

Planning your Web Content Manager system

This information center includes many topics designed to help you plan and manage your Web Content Manager system. Use these topics to ensure that your Web Content Manager system is designed for optimal performance and ease of use, and to ensure that your Web Content Manager system is sufficiently planned and resourced.

- "Human resource planning" on page 21
- Chapter 2, “Planning a website,” on page 19
- Server topologies
- Web Content Manager topologies
- “Road map to building a web content system” on page 51

Configuring and administering Web Content Manager

When installing a Web Content Manager system, you configure the servers in your system differently depending on their role in your system. Once configured, there are also a set of administration functions that you use to manage web content libraries, syndication, and feeds.

- Chapter 4, “Configuring Web Content Manager,” on page 59
IBM Web Content Manager new features and improvements

Web Content Manager includes new features and various improvements to existing features.

The authoring home page

The new authoring home page provides a customized entry point for different types of users. It is a simplified, single-page user interface allowing:
- Casual authors to easily create and submit content.
- Frequent contributors to create and work with their items.
- Power users to easily see what is going on across the site and to quickly create or edit any of the items they’re responsible for.

Navigating the authoring portlet

CMIS support for federated documents

The federated documents feature enables you to insert links to content that resides in a remote content repository. You can now insert links to documents from repositories that support Content Management Interoperability Services 1.0 (CMIS 1.0).

“Setting up support for federated documents” on page 71

Content templating enhancements

The content templating features of Web Content Manager have been expanded to make creating and delivering web content faster and easier. Web content viewers, web content page templates, and content associations work together to provide a flexible framework that you can use to quickly assemble pages containing web content. To illustrate how these building blocks work together, Web Content Manager includes preinstalled web content libraries with sample web content.

- “Getting started with delivering web content on a portal page” on page 531
- “Creating content with sample web content template items” on page 539

The Web Content Manager REST service

The new Web Content Manager REST service makes information in the repository easily accessible, while retaining security. A simple URL interface allows developers to create queries and to utilize this information to easily extend the authoring user interface.

“REST services for Web Content Manager” on page 692

Site area templates

Site areas are now treated as content, allowing you to take advantage of services such as workflow, versioning and metadata profiling. Site administrators can restrict what type of items can be created within different site areas, providing more granular control. You can now map site areas to presentation templates, allowing you to directly render site areas. These enhancements provide authors
with the ability to create "compound" documents, with site areas acting as a parent content item containing both child site areas and content items.

"Creating authoring templates" on page 173

New menu and navigator options

Enhancements to menu and navigator components provide new configuration options including scoping menu queries to a library, and defining the starting point for navigators based on the page context or via a query string. This enables the reuse of the same components within different parts of the site instead of saving the same component multiple times with different queries or starting points. Additionally, navigators now support hierarchical markup making it possible to follow modern web patterns and also produce accessible page navigation.

- "Using a menu element" on page 193
- "Using a navigator element" on page 197

Managed pages

Version 8.0 introduces support for managed pages, which are portal pages that are stored in Web Content Manager. By managing portal pages from within Web Content Manager, you can apply web content features like workflow, version control, and syndication to portal pages. This support provides a robust and integrated method for performing site management by ensuring that changes are coordinated and published together.

Key benefits of managed pages:

- Any changes you make to a page are made in a draft state, without affecting the live server.
- Drafts are organized within projects in Web Content Manager. When you publish draft changes to the live server, a project coordinates the updates and ensures that all drafts are published at the same time.
- A workflow process ensures that page changes are made available on the live server only when the changes are approved. You can create customized workflows to suit the requirements of your site and authoring environment.
- A graphical toolbar integrates managed pages functions directly into your portal pages. The toolbar provides quick access to editing features, projects, and workflow actions that apply to the page. You can also easily change the scope of your changes so that you can manage the changes with projects or directly edit the published site.
- Syndication provides a convenient and flexible method for transferring managed pages from a staging environment to a production environment. With syndication, all required page artifacts and content are transferred at the same time.
- Scripting support is also available for automating managed pages tasks. The Portal Scripting Interface includes methods for working with projects and web content libraries, and the XML configuration interface includes similar support for projects.

Learn more about managed pages.
Supported specifications for inserting links to remote content

The federated documents feature enables you to retrieve information about documents in a remote repository and insert links to those documents in your web content. Several types of feeds are supported for accessing remote content.

The federated documents feature can access content from remote systems that support the following methods:

- Content Management Interoperability Services 1.0 (CMIS 1.0)
- Document Services remote interfaces, as supported by Lotus® Quickr®, IBM Content Manager, and FileNet Content Manager
- Atom feeds (Personalization rules only)

For details on configuring and using the federated documents feature, see the following topics:

- Enable access to remote servers: “Setting up support for federated documents” on page 71
- Using rules with federated documents: “Personalizing federated documents” on page 222
- Inserting links in web content: “Inserting a link to remote content” on page 227

Documentation resources

The starting point for information is the product documentation. The product documentation is delivered in the WebSphere Portal Family wiki. There are still other sites and resources available to you when working with WebSphere Portal, but this consolidation is intended to make finding information easier. It is also intended to drive improvements into the content and let the community edit and comment on the documentation. Knowing where to look for information can save you time and money. Learn more about primary and secondary resources for Web Content Manager and WebSphere Portal documentation and supplemental content.

There are two primary sources for content: the WebSphere Portal Family wiki and WebSphere Portal Support site. An excellent secondary source is developerWorks, where you can find examples and tutorial-based articles.
Each content source links to the other sources to help you navigate between the resources. Each content source has a specific objective and is intended to be used with the other sources.

The product documentation delivered in the wiki (on the Product Documentation tab) is developed by IBM to help you take advantage of features based on expected usage patterns. You can contribute to the content to reflect your experience with the product. The wiki content accessed from the Home tab is developed by the community, both inside and outside of IBM. The intent is to share experiences with the product and based on actual usage patterns and use cases. The Support content is developed by IBM Support to help you avoid and diagnose issues with the intent of being as responsive as possible.

**WebSphere Portal Wiki**

The wiki includes:

- Product Documentation tab which includes:
  - Overview of the product with new feature highlights, product features, and accessibility
  - Planning information for deployment
  - Installation instructions targeted to single server for proof-of-concepts or development servers, stand-alone production, and clustered production environments
  - Configuration options that are typically done one time, or infrequently, and have global effect on the portal
  - Administration tasks for day to day usage
  - Integration instructions
  - Development information to help you develop portlets and applications
  - Troubleshooting information with logging and tracing information
  - Messages to help you diagnose and troubleshoot issues
- Supplemental guides, such as the Performance and Tuning Guide
- IBM Redbooks
- Best Practices
- Deployment scenarios
- Multimedia offerings, such as task-based demonstrations and videos
- Reference cards

Key points to remember about the wiki:

- Information previously delivered in the information center is now delivered from the Product Documentation tab of the wiki.
- The content is experience driven
- You can edit articles, add comments, and author your own articles
- The wiki is monitored by IBM
- You can subscribe to RSS feeds for new articles, comments, and recent edits

**WebSphere Portal Support page**

The Support page includes:

- Technotes written in response to issues with the product or the documentation
- Fix pack downloads including instructions for applying fixes
Troubleshooting information
- Flashes for high priority issues

Key points to remember about the Support page:
- You can subscribe to updates and new Support content using MyNotifications
- You can download tools, such as the IBM Support Assistant

**Accessibility features**

Accessibility features help users who have a physical disability, such as restricted mobility or limited vision, to use software products successfully.

This version of IBM WebSphere Portal:
- Supports installation through a command-line interface known as *console mode*. This is the accessible equivalent of installing by using the graphical user interface.
- Supports interfaces commonly used by screen readers and screen magnifiers (Windows only)
- Supports use of screen-reader software and digital speech synthesizers to hear what is displayed on the screen.
- Can be operated using only the keyboard
- Allows the user to request more time to complete timed responses
- Supports customization of display attributes such as color, contrast, and font size
- Communicates all information independently of color
- Supports the attachment of alternative input and output devices
- Supports alternatives to audio information
- Supports adjustable volume control
- Does not flash the screen at rates that can induce epileptic seizures
- Provides documentation in an accessible format

Recommended browser: Firefox 10.

Recommended screen reader: JAWS 12.

The documentation includes the following features to aid accessibility:
- All documentation is available in HTML formats to give the maximum opportunity for users to apply screen-reader software technology.
- All images in the documentation are provided with alternative text so that users with vision impairments can understand the contents of the images.

When appropriate, the documentation for specific product features contains additional information about accessibility.

See the related links for information about IBM’s commitment to accessibility.

**Types of websites**

Different types of websites will require different solutions and use different applications and features. This section contains examples of different websites and the types of applications and features required to deliver them.
Intranet portal

An intranet portal site is designed to allow information to be quickly disseminated to employees, to make common internal business processes more efficient and to provide a sense of community within an organization.

The site provides access to:
- news about what is happening within the organization.
- alerts that contain information that employees should be aware of and potentially action.
- forms for working with various internal processes such as leave, purchases, and travel.
- a policies and procedures library with online versions of all policy and procedure documentation.
- organizational communities with collaborative features like blogs and forums.
- a search system to enable employees to find content.

The intranet portal also has a personalized home page that is dynamically built using a set of rules that retrieve content based on the current user’s role, department, and location. By appropriately tagging the content, and then matching this content with the current user, the content that is displayed can appropriate for the user.

A number of components are used together to make this intranet portal work:
- WebSphere Portal is used to provide a platform for the integration of content and applications that form the intranet portal.
- The overall theme and top-level navigation of the intranet is also managed using WebSphere Portal.
- Web Content Manager provides the micro-level layout of the content within the portal, and is also used to directly build the news, alerts and communities content.
- The forms are online applications built using the Forms/Lists system, being pulled into the site using Web Content Manager.
- Policies and Procedures are document libraries managed using IBM FileNet Content Manager.
- A custom feed producer would be written to allow the Web Content Integrator to consume a feed from IBM FileNet Content Manager and Web Content Manager would be used to present the documents in a website.
- Personalization is used associate content with users, dynamically generating the content that is displayed in the personalized home page.
- A third-party discussion system is used to deliver collaborative forums in the communities section of the site using a custom portlet.
- Community areas could also be based on Lotus Quickr team rooms.
- WebSphere Portal search is used with the Web Content Manager categorization to provide both a simple text search and a more advanced category-based search.

The size of an intranet portal tends to scale with the size of the organization. A large organization has more information to disseminate, more business processes, more communities of employees. This means that the content and the user population tend to grow at the same rate.
Related concepts:

Designing and setting up a portal site

WebSphere Portal 7 provides improved page builder and improved client side rendering and has a new default portal theme, the Page Builder theme. Themes define how your portal site will look. After installation, a default theme is deployed and you can either customize that theme or create your own. The new themes approach introduced in 7 involves less editing of JSPs and allows you to mix iWidgets and portlets on the same portal page and take advantage of both client side and server side rendering mode. WebSphere Portal 7 continues to support other themes, including your custom themes. If you have an existing portal site you can continue to use your existing themes or you can migrate your themes to the new standard.

“Road map to building a web content system” on page 51

To build a web content system you need to deploy hardware, configure servers, design an authoring system, configure a delivery environment and enable syndication. Get an overview of the steps required to build your web content system. Keep in mind the analysis and design documents developed during the planning phase of a project as you review the road map.

“IBM Web Content Integrator” on page 273

The Web Content Integrator is a solution for integrating externally managed Web content with WebSphere Portal. Through the use of standard content syndication feed technologies based on RSS 2.0, the Web Content Integrator provides a loosely-coupled mechanism for transferring published content and metadata to the portal after they have been approved in the source system. Once the content and metadata have been transferred to the portal, it is possible to use the built-in content management features of WebSphere Portal to secure, personalize, and display the content to users.

Personalization of content

Personalization allows a portal or website to choose which content should appear for a particular user. The WebSphere Portal Personalization component selects content for users, based on information in their profiles and on business rules. Using Portal Personalization, business experts can classify site visitors into segments and target relevant content to each segment. For example, a site using Personalization might show different news articles to managers than to regular employees or different information to valued customers. Personalization offers analytic capabilities to record site usage patterns and includes the LikeMinds recommendation engine, which provides collaborative filtering capabilities. Collaborative filtering uses statistical techniques to identify groups of users with similar interests or behaviors. Inferences can be made about what a particular user might be interested in, based on the interests of the other members of the group.

Related information:

Lotus Quickr

FileNet Content Manager

E-business site

An e-business site is an externally facing site designed to market a company’s products and services to consumers and allow them to purchase these items online. The focus of the site is on helping a consumers match their needs to the appropriate product or service and maximizing their purchases.

The site provides:
press releases about the latest news relating the company’s products and services.

- a promotions area with information about products and services including sales, special package deals, and discounts.
- a products and services catalog area with the entire catalog displayed as a searchable taxonomy.
- a shopping area where users can see:
  - a searchable list of available products and services available for purchase online.
  - a summary of the items they are intending to purchase.
  - an area to calculate tax, shipping and other costs.
  - an area to purchase these items online using mail order, or using a follow-up call from a salesperson.
  - an area to track orders they have already initiated.
- articles where the organization can help customers using articles discussing their products and services.
- a support area with technical documentation, contact numbers, support request forms, FAQs, and downloads.
- a service for subscribing to a newsletter where the user is sent a regular update highlighting new promotions, products or articles.

The home page of the site is used to highlight the latest promotions, and to give a strong sense of the brand of the organization. Throughout the site, collaborative filtering is used to help suggest products and services to consumers based on their purchasing and navigational activity. A discussion system may also be used to allow users to build a community where they can comment on and rate various products and services.

A number of components are used together to build the e-business site:

- WebSphere Portal is used to provide a platform for the integration of content and applications that form the e-business site.
- The overall theme and top-level navigation of the intranet is also managed using WebSphere Portal.
- Web Content Manager provides the micro-level layout of the content within the site, and is also used to directly build the press releases, FAQs, articles, and promotional content.
- The catalog and shopping areas are being run using WebSphere Commerce.
- Personalization is used to suggest products to users based on collaborative filtering, and to email the newsletter.
- A third-party discussion system is used to deliver collaborative forums in the communities section of the site using a custom portlet.

An e-business site can generate a large amount of traffic with a relatively small amount of content.
Related concepts:

WebSphere Portal 7 provides improved page builder and improved client side rendering and has a new default portal theme, the Page Builder theme. Themes define how your portal site will look. After installation, a default theme is deployed and you can either customize that theme or create your own. The new themes approach introduced in 7 involves less editing of JSPs and allows you to mix iWidgets and portlets on the same portal page and take advantage of both client side and server side rendering mode. WebSphere Portal 7 continues to support other themes, including your custom themes. If you have an existing portal site you can continue to use your existing themes or you can migrate your themes to the new standard.

“Road map to building a web content system” on page 51

To build a web content system you need to deploy hardware, configure servers, design an authoring system, configure a delivery environment and enable syndication. Get an overview of the steps required to build your web content system. Keep in mind the analysis and design documents developed during the planning phase of a project as you review the road map.

Personalization of content

Personalization allows a portal or website to choose which content should appear for a particular user. The WebSphere Portal Personalization component selects content for users, based on information in their profiles and on business rules. Using Portal Personalization, business experts can classify site visitors into segments and target relevant content to each segment. For example, a site using Personalization might show different news articles to managers than to regular employees or different information to valued customers. Personalization offers analytic capabilities to record site usage patterns and includes the LikeMinds recommendation engine, which provides collaborative filtering capabilities. Collaborative filtering uses statistical techniques to identify groups of users with similar interests or behaviors. Inferences can be made about what a particular user might be interested in, based on the interests of the other members of the group.

Related information:

IBM WebSphere Commerce

Brochureware site

A brochureware site is an externally facing site that acts as an organization’s presence on the World Wide Web. The overriding focus is on representing the organization’s brand to its potential customers.

This type of site is relatively static and does not need the aggregation capabilities of WebSphere Portal and is instead delivered using both the servlet delivery and pre-rendering features of Web Content Manager. The site is also designed to be easily indexed by search engines like Google.

This site would include:

- information about the organization including its ethics, goals, mission, and history.
- a news area where the organization can talk about what they are doing now and into the future.
- a contact area with lists of locations, phone numbers, and online contact forms where users may submit queries, ask for follow-up calls.
- a job opportunities area where an organization can advertise positions
a partners area where partners of the organization are listed with information about them and their relationship with the organization, and links to partner websites and partner contact details

• information about the organization's products and services including case studies and testimonials

The components in this site are limited with Web Content Manager providing all the presentation, navigation, and content.

A brochureware site can generate a large amount of traffic with a relatively small amount of content.

Related concepts:

“Road map to building a web content system” on page 51

To build a web content system you need to deploy hardware, configure servers, design an authoring system, configure a delivery environment and enable syndication. Get an overview of the steps required to build your web content system. Keep in mind the analysis and design documents developed during the planning phase of a project as you review the road map.

E-library site

An online library site is dedicated to providing access to a large amount of content. The prime example is a news site, where new content is created throughout the day, every day of the week and is published online and then archived as it becomes out of date. Other examples include journals, analyst reports, and software libraries.

Users would typically have to register for these sites, and may then be sent e-mails on a regular basis summarizing the latest content. On a news site the latest news may be free, but users may have to pay for access to archived content. On other types of “library” sites, you may have to pay for access to any of the content.

A number of components are used together to deliver an e-library site:

• WebSphere Portal is used to provide a platform for the integration of content and applications that form the e-business site.
• The overall theme and top-level navigation of the intranet is also managed using WebSphere Portal.
• Web Content Manager provides the micro-level layout of the content within the site, and is also used to directly build content.
• For e-libraries where the content is presented as documents, then IBM FileNet Content Manager would be used to store and manage the documents.
• A custom feed producer would be written to allow the Web Content Integrator to consume a feed from IBM FileNet Content Manager and Web Content Manager would be used to present the documents in a website.

An e-library site may have a large audience and a large amount of content.
Related concepts:

- **Designing and setting up a portal site**

  WebSphere Portal 7 provides improved page builder and improved client side rendering and has a new default portal theme, the Page Builder theme. Themes define how your portal site will look. After installation, a default theme is deployed and you can either customize that theme or create your own. The new themes approach introduced in 7 involves less editing of JSFs and allows you to mix iWidgets and portlets on the same portal page and take advantage of both client side and server side rendering mode. WebSphere Portal 7 continues to support other themes, including your custom themes. If you have an existing portal site you can continue to use your existing themes or you can migrate your themes to the new standard.

- **“Road map to building a web content system” on page 51**

  To build a web content system you need to deploy hardware, configure servers, design an authoring system, configure a delivery environment and enable syndication. Get an overview of the steps required to build your web content system. Keep in mind the analysis and design documents developed during the planning phase of a project as you review the road map.

- **“IBM Web Content Integrator” on page 273**

  The Web Content Integrator is a solution for integrating externally managed Web content with WebSphere Portal. Through the use of standard content syndication feed technologies based on RSS 2.0, the Web Content Integrator provides a loosely-coupled mechanism for transferring published content and metadata to the portal after they have been approved in the source system. Once the content and metadata have been transferred to the portal, it is possible to use the built-in content management features of WebSphere Portal to secure, personalize, and display the content to users.

**Related information**

- [FileNet Content Manager](#)

### Partner site

The audience for a partner site are the partners of an organization. The site provides information and applications that are not applicable to the broader audience of consumers. A partner site would typically require a login but would not use an automated registration system. The business partners would be known to the company and given a login to access the partner site.

A partner site may include the following areas:

- a news area focused on partners. Larger organizations working in multiple areas of business could personalize the news area to suit each partner.
- a promotions area detailing special deals for partners.
- online services, providing useful tools and resources for partners to use, such as calculators, forms, and catalogs.
- a billing area where partners can see the current bill status for services contracted from the organization.
- a community area where partners may post information about themselves, and also interact to help each other and receive help from the organization.

A number of components are used together to deliver a partner site:

- WebSphere Portal is used to provide a platform for the integration of content and applications that form the intranet portal.
- The overall theme and top-level navigation of the intranet is also managed using WebSphere Portal.
- Web Content Manager provides the micro-level layout of the content within the portal, and is also used to directly build the news and promotions content.
- Various IBM products plus third-party applications are used to build and deploy the online services area.
- The catalog and billing services in the site is run using WebSphere Commerce.
- A third-party discussion system is used to deliver collaborative forums in the communities section of the site using a custom portlet.
- Personalization is being used to filter content throughout the site to ensure that the appropriate content and inline services are reaching the appropriate partners.

A partner site is likely to have a limited audience but a large amount of content.
Related concepts:

- Designing and setting up a portal site

WebSphere Portal 7 provides improved page builder and improved client side rendering and has a new default portal theme, the Page Builder theme. Themes define how your portal site will look. After installation, a default theme is deployed and you can either customize that theme or create your own. The new themes approach introduced in 7 involves less editing of JSPs and allows you to mix iWidgets and portlets on the same portal page and take advantage of both client side and server side rendering mode. WebSphere Portal 7 continues to support other themes, including your custom themes. If you have an existing portal site you can continue to use your existing themes or you can migrate your themes to the new standard.

“Road map to building a web content system” on page 51

To build a web content system you need to deploy hardware, configure servers, design an authoring system, configure a delivery environment and enable syndication. Get an overview of the steps required to build your web content system. Keep in mind the analysis and design documents developed during the planning phase of a project as you review the road map.

Personalization of content

Personalization allows a portal or website to choose which content should appear for a particular user. The WebSphere PortalPersonalization component selects content for users, based on information in their profiles and on business rules. Using Portal Personalization, business experts can classify site visitors into segments and target relevant content to each segment. For example, a site using Personalization might show different news articles to managers than to regular employees or different information to valued customers. Personalization offers analytic capabilities to record site usage patterns and includes the LikeMinds recommendation engine, which provides collaborative filtering capabilities. Collaborative filtering uses statistical techniques to identify groups of users with similar interests or behaviors. Inferences can be made about what a particular user might be interested in, based on the interests of the other members of the group.

Related information:

- IBM WebSphere Commerce

Conceptual and functional divisions of a website

When building a website using Web Content Manager, you break up your website into the following conceptual and functional divisions.

*Table 1. Conceptual and functional divisions for layout*

<table>
<thead>
<tr>
<th>Content</th>
<th>Layout</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Content items</td>
<td>• Presentation templates</td>
</tr>
<tr>
<td>• Site areas</td>
<td>• Component designs</td>
</tr>
<tr>
<td>• Components</td>
<td>• Page layout and portlets</td>
</tr>
<tr>
<td></td>
<td>• Themes</td>
</tr>
</tbody>
</table>
When you are developing a website using Web Content Manager and WebSphere Portal, you are splitting the elements of your website between content, context, layout, and style.

**Content:**

There are two types of content:

- **Dynamic**
  - Dynamic content is generated dynamically based on a set of preconfigured parameters, such as navigator or menu.

- **Static**
  - Static content is content where you store markup or files directly in a component, content item, or site area. Where you store your static content depends on how the content is used:
    - Page-specific content is stored in content items.
    - Content related to a section of your site can be stored in a site area.
    - Content that is reused in multiple sections of your site is stored in components.

**Layout:**

The layout and structure of each page in your website are defined using these features:

- The overall structure of each page is determined by the theme you are using, the page layout you choose, and the web content viewer portlets you add to the page layout.
- The layout of the content displayed within each web content viewer is determined by markup used by the current presentation template, and by the markup stored within the elements and components referenced within the presentation template.

**Context:**

The context of the content currently displayed is also important. The layout and design of the page where a content item is displayed is different depending on:

- The current portal page (Different pages can use different themes, layouts, web content viewer portlets, and even different presentation templates.)
- The current site area (Content linked to different site areas can use different template maps that map to different site areas.)
- The current user (Different users can have different access to various page elements, including individual pages, portlets, and web content items.)

Each of these contextual variables allows you to display content or components in different ways, depending on the current context.
Additionally, the profile of the current portlet, content item, or current user can determine what is displayed on a page, as do any personalization features used by your website.

**Style:** While you can place stylistic elements directly within your HTML, it is becoming increasingly more common to use CSS to store and manage all the stylistic elements of a website. In a WCM site, the CSS can be:
- Stored server-side and referenced within a WebSphere Portal theme
- Stored directly within a WCM content item or component

One good practice is to store your CSS markup within an HTML field stored in a content item. This allows you quick access to the CSS if you need to make a change. In addition, you can use advanced features such as workflows and inline editing to help you maintain your CSS. By storing all your stylistic elements in CSS, you can quickly make stylistic changes to your website simply by editing a CSS file instead of editing multiple WCM items.

When a web page is rendered, Web Content Manager takes all these individual item types and combines them to build a complete web page.
Chapter 2. Planning a website

Before building a website, take the time to analyze, plan, and design the entire system that supports the website. You must plan not only the website, but the web content system that manages the website and the infrastructure required for the system. You will also define in advance the roles and users involved in the website and install infrastructure.

Defining the project

It is important to have an understanding of the objectives, deliverables, and scope of a web content system. Your project definition outlines the what, why, and who of the project and can be used throughout the life of your web content system.

You define the following information in your project plan:

**Background**

Define the background information for the project. This information can include why the project is being undertaken and any history of previous and current related projects.

**Mission**

What is the mission of the project? Writing a mission statement can help in determining the requirements of the website.

**Objectives**

It is essential to determine the objectives of the project. These objectives can be provided by the project team, determined from meetings with the project stakeholders, and usability workshops with users. It is important that all stakeholders agree on the final objectives of the site. Each objective should be clear and concise. There should be no room for assumptions or varied interpretations.

**Business objectives**

These objectives define what the business wants to achieve. They focus on concerns such as profit and brand value.

Examples of business objectives for an internet site include:

- Reduce costs of distributing press releases and sales materials
- Reduce the number of phone calls taken by the support team
- Strengthen existing customer loyalty
- Discover potential customers online

Examples of business objectives for an intranet include:
• Provide specialized and tailored content to key groups within the company
• Ensure that employees feel valued
• Reduce business costs by making staff more productive by improving their core tasks

Operational objectives
Operational objectives can be grouped according to short-term and long-term objectives.

Examples of operational objectives include:
• Provide information to company employees
• Provide information to customers
• Develop skills within the company to administer a website
• Develop single sign-on functionality

User objectives
These objectives define the needs of the user of the website and are crucial to developing the site objectives, structure, and functionality.

Examples of a user objective for an internet site include:
• Make it easy for me to find what I want
• Make the information understandable and relevant
• Let me know where I am in the site
• Retain my privacy

Examples of a user objective for an intranet include:
• Find up to date, relevant information as quickly as possible
• Keep me informed of latest news and updates
• Help me perform core business functions such as completing time-sheets and applying for leave
• Reduce my frustrations
• Let me publish information
• Help me feel connected, supported, and valued

Site objectives
Site objectives are derived from the business, operational, and user objectives. They may be the result of the intersection of the business, operational and user objectives, or they may be additional objectives that result from analysis.

Examples of site objectives for an internet site include:
• Provide clear and easy to understand navigation that enables users to find information quickly
• Provide a search feature
• Write content for the web so that it is easy to read and understand
• Provide an FAQ section addressing the most frequently asked questions of the support team
• Provide a framework that structures content for the user and not the business division

Examples of site objectives for an intranet include:
• Provide clear and easy to understand navigation that enables users to find information quickly
• Support key common tasks such as time-sheet entry
• Provide a customized news section
• Provide an FAQ section addressing the most frequently asked questions of the support team
• Provide a framework that structures content for the user and not the business division
• Enable staff to enter content that is then put through a workflow before being published on the intranet

**Project teams**

Define the role of each of the teams involved in the project and the organization of the teams. Some project team examples include:

**Executive sponsors**
The owners and drivers of the project

**Project team**
Responsible for the day-to-day management, analysis, and construction of the new site

**Reference group**
Business unit representatives consulted to ensure that their needs are addressed

**Focus group**
User representatives consulted to ensure that the new site is user focused

**Deliverables**

Document all the deliverables of the project. Give a description of what the deliverable are and the expected timeframe for delivery.

---

**Human resource planning**

These roles are examples of the type of work performed by the people who create and manage a website. A single person can be responsible for more than one of the roles outlined in this section. The roles you implement in your organization to support your website are determined by the size and complexity of the system being deployed. Not all the following roles are required for every website, but all aspects of these roles must be considered during any system deployment.

**The project manager**

Large complex deployments require a project manager to ensure that there are sufficient resources available to deploy a system, that the correct users have been assigned to the project and that the tasks required to deploy the system occur at the correct time in the overall deployment process.

**Defining the project:**
It is important to have an understanding of the objectives, deliverables, and scope of a web content system. The project manager is responsible for creating a project plan to document the overall scope and goals of the project.

• "Defining the project" on page 19
Overseeing the development of a site prototype and design documents:
After defining the project and creating an analysis document, the project manager is responsible for overseeing the development of a site prototype and design documents.

- “Designing a prototype website using HTML” on page 37
- “Creating an analysis document” on page 35

Ensuring that the project has the optimum level of human and physical resources:
The project manager is responsible for ensuring that sufficient human and physical resources have been planned for the initial design, development, and ongoing maintenance of the website and related infrastructure.

Project managing the deployment and installation of the hardware and software used to manage and deliver the website:
The project manager is responsible for overseeing the deployment and installation of the hardware and software used to manage and deliver the website. The project manager monitors the progress of the overall deployment, and takes appropriate action to ensure that milestones are reached on time and on budget.

- Installing

Project managing the ongoing maintenance and upkeep of the hardware and software used to manage and deliver the website:
The project manager also develops a plan for the ongoing monitoring and maintenance of the website and related infrastructure.

- Chapter 7, “Maintaining web content,” on page 583

Related information:
- Planning to install WebSphere Portal
- Chapter 2, “Planning a website,” on page 19

Before building a website, take the time to analyze, plan, and design the entire system that supports the website. You must plan not only the website, but the web content system that manages the website and the infrastructure required for the system. You will also define in advance the roles and users involved in the website and install infrastructure.

The business analyst
A business analyst is responsible for developing an analysis document.

The business analyst is responsible for developing an analysis document. This document is used to record the information gathered from stakeholders that determine the design of the website, its content, and its features. The analysis document includes information on:

- The types of users that will use the system
- What features and applications will be required on the website
- What existing content will need to be imported into the new web content system, and what new content will need to be created
Related concepts:
"Creating an analysis document” on page 35
An analysis document is used to record the information gathered from stakeholders that determines the design of the website, its content, and its features.

Related information:
Chapter 2, “Planning a website,” on page 19

Before building a website, take the time to analyze, plan, and design the entire system that supports the website. You must plan not only the website, but the web content system that manages the website and the infrastructure required for the system. You will also define in advance the roles and users involved in the website and install infrastructure.

The architecture and design team

Based on the data collected in the analysis document, the architecture and design team documents all the things required by the WebSphere Portal system. The design documents the team develops are used by the deployment and development teams to deploy and manage the WebSphere Portal system.

The technical architecture team

The technical architects are responsible for designing the overall server topology required for the entire system and the architecture of the servers that make up the system. The technical architecture team can include technical architects, database architects, security architects, performance engineers, and other resources as required.

Determining a server architecture:
The technical architect is responsible for designing a server architecture describing all the servers and related software required to deliver the website.

- "Server architecture” on page 39
- Server topologies
- Web Content Management topologies

Determining a database architecture:
The database architect is responsible for designing a database architecture describing all the configuration settings required for the databases used in the system.

- "Server architecture” on page 39
- Database considerations
- User registry considerations

Determining syndication strategies:
The technical architect determines what syndication strategies are required between the different environments in the web content system.

- "Server architecture” on page 39
- "Syndication overview” on page 747
Determining a security architecture:
The security architect is responsible for designing a security architecture describing all groups and roles, and access levels required to ensure that different types of users have access only to the functions and content appropriate to their roles.

- "Security architecture" on page 39
- "Users, Groups and Roles" on page 252
- Users and groups
- Controlling access

Determining a content acquisition architecture:
The technical architect is responsible for designing a content acquisition describing the technical details of the methods to use when importing existing content in the new web content system. The content to import is based on the information collected in the analysis document.

- "Content acquisition architecture" on page 48
- "Creating an analysis document" on page 35
- "The IBM Web Content Manager API" on page 617
- "IBM Web Content Integrator" on page 273
- "WebDAV" on page 321

Determining delivery strategies:
The technical architect is responsible for determining which delivery methods are best suited for the types of websites being delivered.

- "Delivery architecture" on page 49
- "Types of websites" on page 7
- Chapter 6, “Delivering web content,” on page 529

Determining system maintenance strategies:
The technical architecture team is also responsible for determining the ongoing maintenance strategies and procedures of the system.

- "Maintenance architecture" on page 50

Related information:
Web Content Manager topologies
Installing WebSphere Portal
Chapter 2, “Planning a website,” on page 19

Before building a website, take the time to analyze, plan, and design the entire system that supports the website. You must plan not only the website, but the web content system that manages the website and the infrastructure required for the system. You will also define in advance the roles and users involved in the website and install infrastructure.

The information architect
The information architect determines the overall informational structure of the website based on requirements gathered in the analysis document.
Determining the website structure:
The web site structure designed by the information architect determines:

- What Portal pages will need to be created if using portlet delivery.
- What site areas will need to be created
- “Managed pages” on page 152
- “Planning a site framework” on page 181

Determining what content types are required:
The content types required for each section in the website structure determine:

- What authoring templates are required
- What presentation templates are required
- What template maps are required
- “Working with authoring templates” on page 173
- “Presentation templates” on page 184
- “Template maps” on page 188

Determining a content profiling architecture:
The information architect determines what content profiling is required for features such as menus.

- “Profiling strategies” on page 191

Related concepts:
“Item management” on page 339

IBM Web Content Manager includes a range of features that enable you manage the web content items used in your system.

Related information:
Web Content Manager topologies
Installing WebSphere Portal
Chapter 6, “Delivering web content,” on page 529

The type of delivery method you use to deliver web content to your viewers will depend on the type of content being delivered, and the type of viewers your website is intended for.

The website designer
A website designer is responsible for designing the layout and style of the web pages in a website. A graphic designer might also be employed to create graphics and color schemes for the website.

Determining what themes, style sheets, and templates are required:
The website designer determines what themes, style sheets, and presentation templates are required to deliver the website.

- “Design architecture” on page 44
- “Designing a prototype website using HTML” on page 37
- Customizing the portal
- Creating a presentation template
Determining what elements and components are required:
The page types and presentation templates required for your site determine:
- What components are required
- What elements are required for each authoring template
- What elements are required to be stored in site areas
  - "Components” on page 190

Determining personalization strategies:
The website designer determines what personalization strategies are required for the website.
- "Personalization” on page 345

The authoring system architect
The authoring system architect is responsible for determining the architecture of the authoring system used to create and manage web content.

Designing the library architecture:
The authoring system architect determines what libraries are required for the website.
- "Web content libraries” on page 124

Determining what authoring templates are required:
The authoring system architect documents the configuration of each authoring template used in the authoring system and what template maps are required to link authoring templates and presentation templates.
- "Working with authoring templates” on page 173
- "Template maps” on page 188

Designing a folder architecture:
The authoring system architect determines what folders are required to store item-types within logical groupings.
- "Working with folders” on page 339

Design a change management model:
The authoring system architect determines what workflows are required to manage approval processes and what items require a workflow.
- "Workflow and change management” on page 135

Determine version control strategies:
The authoring system architect determines what version control strategies are applied to the web content system.
- "Managing versions of items” on page 340
IBM Web Content Manager includes a range of features that enable you to manage the web content items used in your system.

Related concepts:
“Item management” on page 339

Related information:
Chapter 6, “Delivering web content,” on page 529

The type of delivery method you use to deliver web content to your viewers will depend on the type of content being delivered, and the type of viewers your website is intended for.

The deployment team
Based on the design documents developed by the technical design team, the deployment team builds and manages the WebSphere Portal system.

The database administrator
A database administrator is responsible for deploying the database servers and data repositories based on the technical architecture developed by the database architect.

The database administrator is responsible for installing all databases used by the website and related systems. The databases the administrator sets up are based on the technical architecture developed by the technical and database architects. The database administrator will need to become familiar with the user documentation of the data and user repositories used by the web content system.

Related concepts:
“Server architecture” on page 39

Your technical architects define what web content environments are required for your system and the servers required for each environment. This information ensures that you have fully resourced the hardware required to support your web content system.

Related information:
Web Content Manager topologies
Installing WebSphere Portal
Database considerations
User registry considerations

The WebSphere Portal administrator
A WebSphere Portal administrator is responsible for the overall deployment and management of the servers in a WebSphere Portal deployment based on the architecture developed by the technical architect.

Installing and configuring WebSphere Portal:
The WebSphere Portal administrator is responsible for installing and configuring all the instances of WebSphere Portal used in the overall system.

• Installing
Configuring Migration:
The WebSphere Portal administrator is responsible for migrating data and configuration settings from older versions of WebSphere Portal.

Related concepts:

"Security architecture" on page 39
The security architecture describes what groups are required for your site and what access is required for different groups to the authoring portlet and rendered website.

Related information:

Web Content Manager topologies
Installing WebSphere Portal

The web content administrator
A web content administrator is responsible for configuring the web content servers within a WebSphere Portal system.

Web content authoring environments
In a web content authoring environment, a web content administrator performs the following tasks:

Create new pages
An administrator might need to create new pages to display additional authoring portlets used by different users, or for displaying web content viewer portlets to preview sites within.

Configure an authoring portlet
Each authoring portlet in an authoring environment needs to be configured to ensure that it has been configured correctly for the users using each authoring portlet.

Configure a web content viewer portlet
Each web content viewer portlet in an authoring environment needs to be configured to display the content that is being previewed.

Create web content libraries
The web content administrator creates a set of web content libraries based on the recommendations of the web content architect.

Clone a web content repository
Before enabling syndication, a web content administrator, with a database administrator, clones a web content repository from the authoring environment to other environments.

Manage syndication
Syndication relationships are normally created on subscriber servers. As
you typically syndicate only out from an authoring environment to other environments, you would not normally create a syndication relationship from an authoring environment, but a web content administrator might need to manage syndication settings and configurations from time to time.

- “Syndication overview” on page 747
- Working with syndicators and subscribers
- “Syndication tuning” on page 758

Create and manage feed configurations and jobs
Sometimes web content is stored and maintained in external systems. You use the Web Content Integrator to consume content from external systems using feeds.

- Web content feed management

Tune an authoring environment configuration settings
Authoring environment configuration settings can be tuned for specific features.

- “Configuring a web content authoring environment” on page 59

Web content delivery environments
In a web content delivery environment, a web content administrator performs the following tasks:

Create new pages
An administrator might need to create new pages when displaying content using web content viewer portlets.

- “Pages” on page 126

Configure a web content viewer portlet
Each web content viewer portlet in a delivery environment needs to be configured to display the correct content.

- “Web content viewers” on page 531
- “Displaying content with web content viewers” on page 547

Manage syndication
A web content administrator creates syndication relationships to the delivery environment from the web content staging and authoring environments.

- “Syndication overview” on page 747
- Working with syndicators and subscribers
- “Syndication tuning” on page 758

Tune a delivery environment configuration settings
Delivery environment configuration settings can be tuned for specific features.

- “Configuring a web content delivery environment” on page 76

In a UAT environment:

A user acceptance testing environment is set up like authoring and delivery environments depending on the type of UAT being performed. The tasks required to set up the UAT environment are the same as those for the authoring and delivery environments.
Related concepts:

“Delivery architecture” on page 49
Your technical architect and information architect need to define what delivery methods are most appropriate for the website you are delivering.

“Server architecture” on page 39
Your technical architects define what web content environments are required for your system and the servers required for each environment. This information ensures that you have fully resourced the hardware required to support your web content system.

Related information:

Web Content Manager topologies
Installing WebSphere Portal

The security administrator
A security administrator is responsible for securing the overall system including access control strategies and firewalls.

Implementing the security plan outlined in the security architecture document:
The security administrator is responsible for deploying the security features and processes outlined in the security architecture document developed by the security architect.

- Security and authentication considerations
- “Security architecture” on page 39

Managing the user registry:
The security administrator maintains the security registry by running various update delete tasks.

- Managing user data

Managing users and groups:
The security administrator maintains the users and groups stored in the user registry.

- Users and groups

Managing access control
The security administrator manages access controls to pages, portlets, modules, and other applications.

- Controlling access

Additional security tasks
There are various other tasks carried out by the security administrator depending on what environment is being used.

- Enabling step-up authentication, the Remember me cookie, or both

Related information:

Web Content Manager topologies
Installing WebSphere Portal

The development team
Based on the design documents developed by the technical design team, the development team is responsible for creating custom applications using the product API.

The portlet developer
The portlet developer is responsible for developing new portlets.
The portlet developer is primarily responsible for creating custom portlets used to aggregate content and function from external applications. The portlets they develop are based on the portlets identified in the design document.

Related concepts:
- “Design architecture” on page 44

The design architecture describes what your website will look like, and what components will be needed to build your site. You will need to define presentation templates, components, and themes.

Related information:
- Developing

The theme developer
A theme developer is responsible for creating new themes based on the designs developed by the information architect and the graphic designer.

The theme developer is primarily responsible for creating custom themes based on the Web site designs outlined in the design architecture and the HTML used in the website prototype. Themes are a set of JSPs, images, and style sheets packaged together in a common directory structure. They can either be packaged directly in the wps.war or inside a separate WAR file of their own, or with other themes, skins, and resources.

Related concepts:
- “Design architecture” on page 44
- “Designing a prototype website using HTML” on page 37

Before you create a design document for your web content system, it can be useful to create a prototype of your site using HTML. This prototype should be based on the outline defined in your project plan and the data gathered in your analysis document. The site structure, design, and HTML code you develop for your prototype can be used as the basis for many of the items defined in your design document.

Related information:
- Customizing the portal

The web content developer
A web content developer is responsible for extending Web Content Manager by using the Web Content Manager API, developing JSP components and creating web content plug-ins.

The Web Content Manager API
You can use the Web Content Manager API to extend the functions of Web Content Manager.
- “The IBM Web Content Manager API” on page 617

Custom authoring interfaces
Custom authoring interfaces are used by content authors as an alternative to using the authoring portlet.
- “Creating a custom launch page” on page 639
- “Using remote actions” on page 627

Custom plug-ins
You can create custom plug-ins to add custom features to your site such as custom workflow actions and text providers for multi-locale sites.
Related concepts:

“Design architecture” on page 44

The design architecture describes what your website will look like, and what components will be needed to build your site. You will need to define presentation templates, components, and themes.

The website creation team

Based on the design documents developed by the technical design team, the website creation team is responsible for creating the website and related systems.

The website creator

A website creator is responsible for building a website by creating presentation templates, authoring templates, site areas, components and categories. The web site creator is also responsible for creating content management items such as folders and workflows. The items created by a website creator are based on the designs created by the information architect and graphic designer.

Creating web content management items

Before creating a website, the web content creator begins by creating any required content management items:

Creating folders
Folders are used to store item-types within logical groupings.
- “Working with folders” on page 339

Creating workflows
Workflows are required to manage the approval processes of items.
- “Workflow and change management” on page 135

Creating projects
Projects are used to manage the approval process of a collection of items.
- “Projects overview” on page 133

Creating a website

To create a website the following Web Content Manager items will need to be created:

Site areas
Site areas are used to define the site framework of a site. Content items that form part of the site framework will be stored within the site framework.
- Creating site areas

Authoring templates
Authoring templates are required for the different types of pages in a website. Content items created by content authors will be based in the authoring templates created by the website creator.
- “Creating authoring templates” on page 173

Presentation templates
Presentation templates determine the layout of the pages in a website and are linked to authoring templates by the template maps defined in site areas.
- Creating a presentation template
Elements and components
Elements are added to authoring templates and site areas to store different types of content. Components are used to create single elements that can be reused across a website.

- "Elements" on page 193

Categories
Categories are used to profile items that have been configured to allow content profiling.

- "Planning a taxonomy" on page 192

Related concepts:
- "Information architecture" on page 43
  The information architecture describes the information structure of the site and how users will navigate through the site.
- "Design architecture" on page 44
  The design architecture describes what your website will look like, and what components will be needed to build your site. You will need to define presentation templates, components, and themes.

The web content author
A web content author is responsible for creating web content for the sites developed and managed using Web Content Manager.

Content authors create content either by accessing an authoring portlet, or by using a custom content authoring interface.

Creating content items
Presentation templates determine the layout of the pages in a website and are linked to authoring templates by the template maps defined in site areas.

- "Content items" on page 189

Working with elements and components
Elements are added to authoring templates and site areas to store different types of content. Components are used to create single elements that can be reused across a website.

- "Elements" on page 193

The web content manager
A web content manager is responsible for performing ongoing site maintenance activities.

Moving, linking and copying items:
From time to time a web content manager will need to redesign the structure of a website by moving or copying items.

Managing authoring templates:
From time to time a web content manager will need to reapply or change the authoring template used by content items and site areas.

Restoring items:
From time to time a web content manager will need to restore an item to a previously saved version.

Editing user profiles:
Profiling information specific to a web content system can be added to a user profile to allow you to personalize content for a specific set of users.

Chapter 2. Planning a website
Managing access controls
From time to time a web content manager will need to reapply or change
the access controls of a set of items.

The content acquisition team
Based on the content identified in the analysis document, the content acquisition
team is responsible for importing content from legacy systems into the new
website. The team consists of a mixture of administrators and developers. Team
members need to be experts in both legacy products and WebSphere Portal

The Web Content Manager API:
You can use the Web Content Manager API to import content into Web
Content Manager.
- “The IBM Web Content Manager API” on page 617

Web content feeds
You use the Web Content Integrator to consume content from external
systems using feeds.
- “IBM Web Content Integrator” on page 273

Using WebDAV for web content:
With WebDAV for WebSphere Portal, you can use standard operating
system tools to create, modify, and delete web content rather than the
standard authoring portlet.
- “WebDAV” on page 321

Related concepts:
“Content acquisition architecture” on page 48
The content acquisition architecture is used to define what existing content will be
imported into your web content system, how it will be imported, and what content
will need to be created.

The maintenance team
Based on the maintenance architecture developed by the technical design team, the
maintenance team is responsible for the ongoing monitoring and maintenance of
the overall system.

The database administrator
After all databases have been setup, the database administrator is then responsible
for the ongoing monitoring and maintenance of all databases in the system.

The WebSphere Portal administrator
The WebSphere Portal administrator is responsible for ongoing maintenance of the
WebSphere Portal servers in the system.
- Administering

The web content administrator
Manage syndication:
Syndication relationships are normally created on subscriber servers. As
you typically syndicate only out from an authoring environment to other
environments, you would not normally create a syndication relationship
from an authoring environment, but a web content administrator might
need to manage syndication settings and configurations from time to time.
Manage feed configurations and jobs:
Sometimes web content is stored and maintained in external systems. You use the Web Content Integrator to consume content from external systems using feeds.
- Web content feed management

Maintain an authoring system:
From time to time a web content administrator needs to perform some maintenance tasks.
- Chapter 7, “Maintaining web content,” on page 583

The security administrator

Managing the user registry:
The security administrator maintains the security registry by running various update delete tasks.
- Managing user data

Managing the users and groups:
The security administrator maintains the users and groups stored in the user registry.
- Users and groups

Managing access control:
The security administrator manages access controls to pages, portlets, modules, and other applications.
- Controlling access

Additional security tasks:
There are various other tasks carried out by the security administrator depending on the environment being used.
- Enabling step-up authentication, the Remember me cookie, or both

The performance engineer

The performance engineer is responsible for ensuring that all the hardware and software in the system are tuned for maximum performance and reliability.
- Troubleshooting

Related concepts:
“Maintenance architecture” on page 50
There are a set of tasks that you need to plan for that will maintain the health and integrity of your web content system. Your technical architect and database architect need to define what maintenance procedures are required for your system, and when and how often they need to be run.

Creating an analysis document

An analysis document is used to record the information gathered from stakeholders that determines the design of the website, its content, and its features.

These are some examples of the analysis that can be undertaken when designing a web content system.
User analysis

To design a website that supports the needs of the company and the users, you must know who your audience is. It is important to determine your users at this early stage of the project. Some of the things you want to discover are:

- Who they are?
- Who are the most important groups?
- What do they want to do on the site?
- What will make them return to the site?
- What is their level of experience with the web?

To help you understand your main user groups even further, you can develop personas and scenarios:

Personas

A persona is a fictional person who represents a major user group for your site. By using the information gathered about your users, create a person who represents each main user group. Give them:

- A name and picture
- Demographics, such as age, education, and family status
- Job role and responsibilities
- Their goals and tasks in relation to the site
- A background on their computer and web usage

Scenarios

A scenario is a story of how users might experience the site. They help you visualize the site and its users. They can help you view the navigation process as a whole. Scenarios are also useful in validating the website design after it is finished and can be used in usability testing. Use your personas, and give them a task to accomplish on the site. Write a story about how the character uses the site to finish the given task. Be creative.

Competitive Analysis

If you are building a public website it is useful to look at what the competition is doing. Generate a list of competitors and document things you like and dislike about their Internet sites. For intranet sites where you cannot compare your site with those of competitors, you can instead ensure that your intranet meets current standards.

Website requirements

Website requirements describe the features and functions of a website. They document what the site must have and also what users can do. The requirements are not used to describe how to build the website, which is detailed in the design document.

For example:

General features:

- Search
- Contact details

Allow users to:

- Purchase a product
• Sign up for a newsletter
• Fill out a time-sheet online

Include the following content and site areas:
• Press releases
• Policies and guides
• Links to related articles

Content Inventory

It is useful to identify the types of content that make up the site. As your new website might be a redesign of an existing site, identify what content exists and what new content needs to be written. Create a content inventory, and add any existing web pages and potential types of content that you can think of. Types of content include:
• Static content such as copyright notices and privacy statements
• Dynamic content such as latest news and product campaigns
• Transactional content such as logon pages and registration pages for email newsletters

When creating a content inventory you can collect the following information:
• A brief description
• Topic area or category
• Priority
• Format, such as a web page, a file, or on paper
• Intended audience
• Related content
• Created date
• Last modified date
• Owner
• Author
• Expiration date

Designing a prototype website using HTML

Before you create a design document for your web content system, it can be useful to create a prototype of your site using HTML. This prototype should be based on the outline defined in your project plan and the data gathered in your analysis document. The site structure, design, and HTML code you develop for your prototype can be used as the basis for many of the items defined in your design document.

By creating a prototype website using HTML, you get to see what your actual website will look like, plus you develop many of the features used by your web content system and website.

Building the prototype
• The prototype website should include example pages of each page type defined in your project plan and analysis document. It should also include the basic site structure of your proposed website.
• The prototype should be a functional website. It should include functioning navigational features to enable you to browse through the prototype. Examples of real content are used to populate the prototype.
• Take care to ensure that the HTML and other code you write to create the prototype meets coding best practices because much of the code you develop for the prototype will be reused in your Web Content Manager site.

Using HTML to plan the design of your site

A useful method to begin the design of your content management system is to use HTML to develop a prototype of your site. Do this task in consultation with the web content information architect.

The HTML site should include:
• The design for the main home pages of each section of your site
• The design for content pages stored in each section
• The navigation features of your site

The HTML site can then be used to help you determine what web content items need to be created for your web content management system.

For example, the design of each web page in the HTML mock-up can determine different parts of your Web Content Manager site:
• The layout of each HTML page determines the layout of your presentation templates.
• The design of the navigation features determines what menus and navigators you will need to create.

Other features of your design, such as images and style sheets, will determine what components need to be created and whether you need to develop a new WebSphere Portal theme.
Creating a design document

After defining the project and creating an analysis document, you define the requirements of your web content system in a design document. The design document outlines what types of content will be needed for your site, how they will be structured, how content will be authored, and what the final website will look like.

Server architecture

Your technical architects define what web content environments are required for your system and the servers required for each environment. This information ensures that you have fully resourced the hardware required to support your web content system.

Environmental architecture

Every Web Content Manager system consists of at least one authoring environment and one delivery environment. The authoring environment is used to create and manage the individual items that make up your website. The delivery system is used to deliver the web content to your viewers. In addition to these environments, you might also require testing environments where you perform user acceptance testing (UAT).

The technical architecture team documents the overall environmental architecture required for your Web Content Manager system including:

- What authoring, delivery and UAT environments are required
- What WebSphere Portal servers or clustered are required in each environment
- What database servers are required to store the data repositories used by each WebSphere Portal server or cluster
- What syndication relationships are required between each environment
- What user repositories, such as LDAP, are required

Server architectures

The technical architecture team also documents a server architecture for each server described in the environmental architecture. A server architecture describes the detailed setup of each server, including:

- What hardware is required for each server
- What operating system is required for each server
- What software is required
- Configuration settings for all hardware, software, and operating systems
- Other relationships such as remote data repositories, and user repositories

Security architecture

The security architecture describes what groups are required for your site and what access is required for different groups to the authoring portlet and rendered website.

Note: The following example describes the type of security architecture required for an authoring environment. In most cases, the security architecture for a staging or delivery environment would be much simpler with only the All Portal User...
Groups group being assigned user access to the library. This prevents users from being able to edit content and disables features like authoring tools from being displayed on the published site.

In this example, item type roles are applied to the following groups:

Table 3. Groups

<table>
<thead>
<tr>
<th>Group</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>WCMAdmins</td>
<td>Members of this group require access to all features of the authoring portlet.</td>
</tr>
<tr>
<td>SiteAdmins</td>
<td>Members of this group require access to all features of the authoring portlet except workflow.</td>
</tr>
<tr>
<td>SiteDesigners</td>
<td>Members of this group require access to content items, presentation templates, authoring templates, and components.</td>
</tr>
<tr>
<td>ContentAuthors</td>
<td>Members of this group require editor access to content items only.</td>
</tr>
<tr>
<td>ContentApprovers</td>
<td>Members of this group require contributor access to content items only.</td>
</tr>
</tbody>
</table>

Library access

The simplest method of setting library access is to grant contributor access to all your groups. This access gives all users and groups contributor access to the library and authoring portlet. Additional access is then granted to each group using resource permissions. You can also grant the Anonymous Portal User group user access to ensure all anonymous users can access the library if anonymous access is required for your website.

Table 4. Library access

<table>
<thead>
<tr>
<th>Roles</th>
<th>Allow propagation</th>
<th>Allow inheritance</th>
<th>User or group</th>
</tr>
</thead>
<tbody>
<tr>
<td>Administrator</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Manager</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Editor</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>User</td>
<td>No</td>
<td>Yes</td>
<td>Anonymous Portal User</td>
</tr>
<tr>
<td>Contributor</td>
<td>Yes</td>
<td>Yes</td>
<td>WCMAdmins, SiteAdmins, SiteDesigners, ContentAuthors, ContentApprovers</td>
</tr>
</tbody>
</table>

Resource permissions

Set the following resource permissions for each role type:

- The WCMAdmins group is assigned the administrator role for all resources.
- The SiteAdmins group is assigned the manager role to all resources except workflow and workflow elements as they do not require access to these resources.
- The other groups are assigned roles for each resource according to the following tables.
Authoring templates

The SiteDesigners group is assigned editor access to authoring templates as they are required to create new authoring templates.

Table 5. Authoring templates

<table>
<thead>
<tr>
<th>Roles</th>
<th>Allow propagation</th>
<th>Allow inheritance</th>
<th>User or group</th>
</tr>
</thead>
<tbody>
<tr>
<td>Administrator</td>
<td>Yes</td>
<td>Yes</td>
<td>WCMAdmins</td>
</tr>
<tr>
<td>Manager</td>
<td>Yes</td>
<td>Yes</td>
<td>SiteAdmins</td>
</tr>
<tr>
<td>Editor</td>
<td>Yes</td>
<td>Yes</td>
<td>SiteDesigners</td>
</tr>
<tr>
<td>User</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Contributor</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
</tbody>
</table>

Components

SiteDesigners are assigned editor access to components as they are required to create components.

Table 6. Components

<table>
<thead>
<tr>
<th>Roles</th>
<th>Allow propagation</th>
<th>Allow inheritance</th>
<th>User or group</th>
</tr>
</thead>
<tbody>
<tr>
<td>Administrator</td>
<td>Yes</td>
<td>Yes</td>
<td>WCMAdmins</td>
</tr>
<tr>
<td>Manager</td>
<td>Yes</td>
<td>Yes</td>
<td>SiteAdmins</td>
</tr>
<tr>
<td>Editor</td>
<td>Yes</td>
<td>Yes</td>
<td>SiteDesigners</td>
</tr>
<tr>
<td>User</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Contributor</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
</tbody>
</table>

Content

Both the SiteDesigners and ContentAuthors groups are assigned editor access to content as they are required to create content items.

The ContentApprovers group is assigned Contributor access only, because they are not required to create new content items, but need approve access to content items during a workflow. You must also assign the ContentApprovers group approve access in the properties section of any workflow stages that ContentApprovers will use to approve content items.

Table 7. Content

<table>
<thead>
<tr>
<th>Roles</th>
<th>Allow propagation</th>
<th>Allow inheritance</th>
<th>User or group</th>
</tr>
</thead>
<tbody>
<tr>
<td>Administrator</td>
<td>Yes</td>
<td>Yes</td>
<td>WCMAdmins</td>
</tr>
<tr>
<td>Manager</td>
<td>Yes</td>
<td>Yes</td>
<td>SiteAdmins</td>
</tr>
<tr>
<td>Editor</td>
<td>Yes</td>
<td>Yes</td>
<td>SiteDesigners</td>
</tr>
<tr>
<td>User</td>
<td>Yes</td>
<td>Yes</td>
<td>ContentAuthors</td>
</tr>
<tr>
<td>Contributor</td>
<td>Yes</td>
<td>Yes</td>
<td>ContentApprovers</td>
</tr>
</tbody>
</table>

Presentation Templates

The SiteDesigners group is assigned editor access to presentation templates as they are required to create new presentation templates.
Table 8. Presentation templates

<table>
<thead>
<tr>
<th>Roles</th>
<th>Allow propagation</th>
<th>Allow inheritance</th>
<th>User or group</th>
</tr>
</thead>
<tbody>
<tr>
<td>Administrator</td>
<td>Yes</td>
<td>Yes</td>
<td>WCMAdmins</td>
</tr>
<tr>
<td>Manager</td>
<td>Yes</td>
<td>Yes</td>
<td>SiteAdmins</td>
</tr>
<tr>
<td>Editor</td>
<td>Yes</td>
<td>Yes</td>
<td>SiteDesigners</td>
</tr>
<tr>
<td>User</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Contributor</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
</tbody>
</table>

Site areas and pages

Only the WCMAdmins and SiteAdmins groups require access to site areas and pages as these are the only groups that build site frameworks.

Table 9. Site areas and pages

<table>
<thead>
<tr>
<th>Roles</th>
<th>Allow propagation</th>
<th>Allow inheritance</th>
<th>User or group</th>
</tr>
</thead>
<tbody>
<tr>
<td>Administrator</td>
<td>Yes</td>
<td>Yes</td>
<td>WCMAdmins</td>
</tr>
<tr>
<td>Manager</td>
<td>Yes</td>
<td>Yes</td>
<td>SiteAdmins</td>
</tr>
<tr>
<td>Editor</td>
<td>Yes</td>
<td>Yes</td>
<td>SiteDesigners</td>
</tr>
<tr>
<td>User</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Contributor</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
</tbody>
</table>

Taxonomy

Only the WCMAdmins and SiteAdmins groups require access to taxonomies as these are the only groups that build taxonomies.

Table 10. Taxonomy

<table>
<thead>
<tr>
<th>Roles</th>
<th>Allow propagation</th>
<th>Allow inheritance</th>
<th>User or group</th>
</tr>
</thead>
<tbody>
<tr>
<td>Administrator</td>
<td>Yes</td>
<td>Yes</td>
<td>WCMAdmins</td>
</tr>
<tr>
<td>Manager</td>
<td>Yes</td>
<td>Yes</td>
<td>SiteAdmins</td>
</tr>
<tr>
<td>Editor</td>
<td>Yes</td>
<td>Yes</td>
<td>SiteDesigners</td>
</tr>
<tr>
<td>User</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Contributor</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
</tbody>
</table>

Workflow Items

Only the WCMAdmins group requires access to workflow items as this is the only group that creates workflows. The groups that use workflows do not require access to the workflow items resource permissions.

Table 11. Workflow Items

<table>
<thead>
<tr>
<th>Roles</th>
<th>Allow propagation</th>
<th>Allow inheritance</th>
<th>User or group</th>
</tr>
</thead>
<tbody>
<tr>
<td>Administrator</td>
<td>Yes</td>
<td>Yes</td>
<td>WCMAdmins</td>
</tr>
<tr>
<td>Manager</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Editor</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>User</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Contributor</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
</tbody>
</table>
Item-level security inheritance

By default, each role's access is automatically inherited down to each item in a library. To prevent a user or group from automatically having inherited access to an item, you need to turn off inheritance on that item.

The permissions set for item type do not automatically give you access to individual items. They give you access only to specific tasks and views within the authoring portlet.

You can also assign specific access to individual groups or users on each item.

**Related concepts:**

"Users, Groups and Roles" on page 252

**Related information:**

User registry considerations

Information architecture

The information architecture describes the information structure of the site and how users will navigate through the site.

The site map

The website structure designed by the information architect determines what pages and site areas will need to be created to give your site a hierarchical structure. The site map describes the structure of the site and determines what pages and site areas are required for your site. For example:

- Home page
  - News Page
  - Products Page
    - Product Site Area 1
    - Product Site Area 2
    - Product Site Area 3
  - Downloads Page
  - Support Page

Content types

The content types identified by the information architect determine what authoring templates are required for your authoring system. For example, your site might require the following content types:

- Section home pages
- News articles
- Employee profiles
- Product information
- Photo galleries
- Legal disclaimers
Content profiling and taxonomies

The information architect is responsible for determining what taxonomies are required to allow users to profile content. This information determines what content will be displayed within menu components.

This is an example of a taxonomy for a financial services company:

- **MetaBank taxonomy**:
  - Financial
    - Banking Solutions
    - Interest Rates
  - Personal
  - Business
  - Corporate
  - News

**Related concepts:**

- “Managed pages” on page 152
  Managed pages streamline site management in your portal by simplifying how you create pages and add content. Because page information and content are stored in web content libraries, you can more easily coordinate and publish changes with syndication.

- “Planning a site framework” on page 181
  Intelligent pages and site areas are used to define a hierarchical site framework. Content items are saved within the site framework to give your content structure and context.

- “Working with authoring templates” on page 173
  When working with authoring templates you define settings for the authoring template itself, as well as defining default settings for the items created using the authoring template.

- “Presentation templates” on page 184
  Presentation templates determine the structure of each web page in your site and which elements and components are displayed on each page.

- “Profiling strategies” on page 191
  You use the profiling features of IBM Web Content Manager to group content items into different types of content.

Design architecture

The design architecture describes what your website will look like, and what components will be needed to build your site. You will need to define presentation templates, components, and themes.

Themes and style-sheets

You list the themes, style-sheets and styles in the design architecture. These designs can be based on the styles developed for your prototype website.

Page wireframes

Page wireframes can be used to describe the basic structure of each page-type in your site and the elements used on each page.
**Presentation templates**

You list each presentation template required for your website plus its HTML in the design architecture. The HTML you developed for the prototype website can be used as the basis for the HTML in the design document. The presentation template design includes:

- The HTML design of the presentation template
- Access settings
- Whether a workflow is required and, if so, which one

**Components and elements**

Each component and element required for your website is listed in the design architecture as well as additional information such as:

- HTML design for any required component fields
- Parameter selections for any component parameters
- Access settings
- Whether a workflow is required and, if so, which one

**Personalization strategies**

The website designer also determines what content needs to be personalized for specific users. For example, you might want to display a personalized home page for each individual user based on the user profile. The personalization strategies documented in the design architecture will determine what rules, content spots, and campaigns need to be created when using the Personalization application.

**Related concepts:**

- "Presentation templates" on page 184
  Presentation templates determine the structure of each web page in your site and which elements and components are displayed on each page.
- "Components" on page 190
  You use components to store elements that are used in more than one area of your website. For example, a company logo or a copyright notice.
- "Personalization" on page 343
  Portal Personalization provides automatic customization of website content for individual users and user groups.

**Related information:**

- Customizing the portal

**Authoring architecture**

The authoring architecture describes what types of content will be required for the site and what change management strategies will be applied when updating content and design. You will need to define what authoring templates are required for your system, what workflows will be used to manage changes and what folders will be used to group items in the authoring portlet.

**Library architecture**

The library architecture describes where you will store your web content items. For example, you could split your site between the items that control the look and feel of your site, such as presentation templates and components, from the content of
your site. You could also separate your libraries into different team libraries such as "human resources" and "support".

**Authoring templates**

You will need a separate authoring template for each type of content item and site area used by your site. The list of authoring templates defined in your authoring architecture is based on the different page types identified in the project plan, analysis document, and prototype website.

The information specified for each authoring template includes:

- Which content type to use. For example:
  - A site area for a parent item
  - A content item for the children of the parent item
  - A site area for a sub-section of a parent item
- For the authoring template itself:
  - Name
  - Display title
  - Description
  - Template type
  - Access settings
- Default form settings including:
  - Default presentation template, if required
  - Default style sheet, if required
  - Form layout
  - Whether editors can manage elements
  - Actions to hide
  - Where to save items
  - Help text
  - Versioning strategy
- Default content and fields for the item:
  - Name and field display options
  - Display title and field display options
  - Description and field display options
  - Required elements
    - HTML design for any default element content
    - Default parameter selections for any component parameters
    - Field display options
- Default properties for each item:
  - Any default profiling and field display options
  - Any default workflow parameters and field display options
  - Any default access settings and field display options
  - Field display options for the History section
**Workflows**

You use workflows to control the access to, verification of, and eventual approval of items. Only if an item is approved at all stages up to a published stage can it be viewed on your website. You can use a workflow to:

- Review the accuracy of content
- Review content for any legal implications
- Review content to ensure it meets accessibility guidelines
- Ensure that no malicious code such as cross scripting attacks have been added to content

Each workflow, workflow stage and workflow action required for your web content system is listed in the authoring architecture.

**Folders**

Folders are used to group sets of item types into logical groupings. This is useful when you have large numbers of items in your library and want to distinguish between different groups of items within each item type view. Each folder required for your web content system is listed in the authoring architecture.

For example, you could use a set of folders to groups different sets of image components:

- Logos
- Diagrams
- Staff photos
- Product photos

**Version management strategies**

You can configure your system to either automatically save a version of an item each time it is published, or allow users to select when to save a version of an item. The version management strategy for different item types are documented in the authoring architecture.
Related concepts:

“Web content libraries” on page 124
Your web content system can contain multiple libraries. The number of libraries required is determined by the type of website you are creating, and the types of users who require access to each library.

“Users, Groups and Roles” on page 252
Your content management system will require different types of users. You will need to create a different group for each type of user and then assign those groups different roles within your system.

“Working with folders” on page 339
You use folders to group sets of item types into logical groupings.

“Workflow and change management” on page 135
You can manage changes to web content items either by creating drafts, using workflows or adding items to projects.

“Managing versions of items” on page 340
You can configure your system to either automatically save a version of an item each time it is published, or allow users to select when to save a version of an item. You can restore items individually, or choose to restore a set of items within a library that either have the same label or were versioned at, or before, a specified date and time. Versioning is controlled through the Web Content Manager configuration services in the WebSphere Integrated Solutions Console and is enabled by default.

**Content acquisition architecture**

The content acquisition architecture is used to define what existing content will be imported into your web content system, how it will be imported, and what content will need to be created.

The content acquisition architecture is based on the content inventory that was collected during the analysis phase of the project. For each piece of content identified in the content inventory you define the following information:

- Will this content be imported into the web content system, or will it be created afresh.
- If the content is to be imported, what tool will be used. For example, WebDav or the Web Content Integrator.

Additionally, the type of items being created need to be identified. For example:

- Reusable pieces of content are best created as components
- Web page specific content is best created as a content item
- Content specific to a particular section of a website is best saved within a site area
You can use the Web Content Manager API to extend functions of Web Content Manager.

The Web Content Integrator is a solution for integrating externally managed Web content with WebSphere Portal. Through the use of standard content syndication feed technologies based on RSS 2.0, the Web Content Integrator provides a loosely-coupled mechanism for transferring published content and metadata to the portal after they have been approved in the source system. Once the content and metadata have been transferred to the portal, it is possible to use the built-in content management features of WebSphere Portal to secure, personalize, and display the content to users.

With WebDAV for IBM WebSphere Portal, you can use standard operating system tools to create, modify, and delete web content rather than the standard authoring portlet.

**Delivery architecture**

Your technical architect and information architect need to define what delivery methods are most appropriate for the website you are delivering.

The delivery architecture describes which delivery methods are required for your web content system.

**Pre-rendered delivery**

You can pre-render a complete website into HTML and save it to disk. The pre-rendered site can then be used as your live site and displayed to users using either Web Content Manager or a web server. You deploy a pre-rendered site when you are not using any WebSphere Portal features, such as portlets, and your content is static and is updated only periodically.

**Servlet delivery**

Users can access content displayed via the Web Content Manager servlet. A servlet-delivered website should be used when you don't need to use any WebSphere Portal based features such as authoring tools.

**Local web content viewer delivery**

Web content viewers are portlets that display content from a web content library as part of a portal page. If your presentation is simple, a single web content viewer can be sufficient, but you can also use multiple web content viewers to aggregate content from different libraries and provide a richer experience for your users. A local web content view portlet is used to display content within your web content delivery environment.

**Remote web content viewer delivery**

A remote web content view portlet is used to display content on a remote WebSphere Portal server or cluster.
The type of delivery method you use to deliver web content to your viewers will depend on the type of content being delivered, and the type of viewers your website is intended for.

**Maintenance architecture**

There are a set of tasks that you need to plan for that will maintain the health and integrity of your web content system. Your technical architect and database architect need to define what maintenance procedures are required for your system, and when and how often they need to be run.

**Backup and restore strategies**

You should document backup and restore strategies for all the software and repositories in your web content system. This approach includes documenting procedures to back-up and restore:

- Data repositories
- Themes
- Portlets
- Page structures
- Configuration settings
- Custom application

**Applying updates**

The process and timing of software updates are defined in the maintenance architecture. This not only applies to Web Content Manager and WebSphere Portal, but to all software products in your web content system.

For example, you would document procedures to install cumulative fixes for Web Content Manager on a regular basis.

**Running maintenance tools**

You document the process and timing of running each maintenance tool. This documentation not only applies to Web Content Manager and WebSphere Portal, but to all software products in your web content system.

For example, you would document when and how often to run maintenance tools, such as the member fixer task, and when to clear item and version histories to maintain server performance.
Related concepts:

Chapter 7, “Maintaining web content,” on page 583

You use these tools and features to ensure the efficient operation of your Web Content Manager system.

Road map to building a web content system

To build a web content system you need to deploy hardware, configure servers, design an authoring system, configure a delivery environment and enable syndication. Get an overview of the steps required to build your web content system. Keep in mind the analysis and design documents developed during the planning phase of a project as you review the road map.

Related concepts:

Related tasks:

IBM WebSphere Portal provides flexible deployment options ranging from proof-of-concept where you can examine and test functionality to a highly available and scalable production environment.

Deploying the authoring environment

The first environment to deploy is your authoring environment. This environment can initially be used by a small team as a development environment, but can eventually be used as the authoring environment for all your users. The authoring environment is deployed based on the technical architecture defined in the project design documents.

1. Based on the database architecture defined in the project design document, the database administrator deploys a database server for the authoring environment.
   - Database considerations
2. Based on the security architecture defined in the project design document, the LDAP administrator:
   a. Creates a new LDAP server if one does not exist.
b. Creates all the groups and users required for the web content system.

- Managing user data
- “Security architecture” on page 39

3. Based on the server architecture defined in the project design document, the WebSphere Portal administrator does the following:

a. Installs a WebSphere Portal server or cluster of servers.

b. Configures the WebSphere Portal server or cluster to use the database server setup by the database administrator.

c. Configures WebSphere Portal to use the LDAP server as its user repository.

d. Configures various WebSphere Application Server, WebSphere Portal and Web Content Manager configuration properties to ensure the system is correctly setup for web content authoring and is tuned for optimal performance.

- Installing
- Configuring
- Chapter 4, “Configuring Web Content Manager,” on page 59

4. Based on the information architecture and security architecture defined in the project design document, the WebSphere Portal administrator:

a. Creates all the libraries required for the web content system.

b. Assigns users and groups to the roles and access permissions.

c. Create all pages required by the web content system.

d. Adds all required authoring portlets to the appropriate pages and configures the authoring portlets for use by different teams to create and manage web content items.

e. Adds all required web content viewer portlets to the required pages for use as preview pages.

- “Users, Groups and Roles” on page 252
- “Security architecture” on page 39
- Working with managed pages
- “Authoring portlet settings” on page 264
- “Displaying content with web content viewers” on page 547

5. Final testing and tuning of the authoring environment is undertaken by all administrators.

The authoring environment is now ready to use.

**Building the content authoring system**

Before you can start adding content to your web content system, you need to create all the items that will be used to manage and deliver your web content.

1. Based on the design architecture defined in the project design document, the web content developer creates plug-ins and JSP code in preparation for the creation of the content authoring system.

- “The web content developer” on page 31
- “Design architecture” on page 44
- Chapter 8, “Developing,” on page 617

2. Based on the information architecture defined in the project design document, the website creator does the following:

a. Creates site areas.
b. Creates taxonomies and categories.

3. Based on the content authoring architecture defined in the project design document, the website creator does the following:
   a. Creates workflow actions, stages, and defines workflows.
   b. Creates authoring templates.

4. Based on the design architecture defined in the project design document, the website creator does the following:
   a. Creates components.
   b. Creates presentation templates.

5. Based on the information architecture defined in the project design document, the website creator edits site areas and defines template maps.

6. If required, the web content developer also creates a custom content authoring interface based on the content authoring architecture defined in the project design document.

7. The website creator then creates some test content to test the content authoring system, and previews the test content. Any defects found in the content authoring system are addressed and further changes made to improve performance and usability.

**Importing and creating content**

When the authoring system is ready, you then import and create the default content for your system. This is often managed by a separate team from the team that builds the web content authoring system.

1. Based on data gathered during the analysis phase of the project, the content acquisition team identifies and prepares any existing content for import into the web content authoring system.

2. The content acquisition team then imports existing content into the web content authoring system using features such as the Web Content Integrator, WebDav or the web content API.
3. The content acquisition team then create new content based on the information architecture defined in the project design document.
   - “Information architecture” on page 43

4. The content acquisition team then test the authoring system, and preview the default content. Any defects found in the authoring system are addressed and further changes made to improve performance and usability.

### Deploying the delivery environment

The delivery environment is used to deliver your website to your website viewers. The delivery environment is deployed based on the requirements defined in the project design document.

1. Based on the database architecture defined in the project design document, the database administrator does the following:
   a. Deploys a database server for the delivery environment.
   b. Clones the data stored on the authoring database onto the delivery database.
      - Database considerations
      - “Cloning a web content repository” on page 613

2. Based on the server architecture defined in the project design document, the WebSphere Portal administrator does the following:
   a. Installs a WebSphere Portal server or cluster of servers.
   b. Configures the WebSphere Portal server or cluster to use the database server setup by the database administrator.
   c. Configures various WebSphere Application Server, WebSphere Portal and Web Content Manager configuration properties to ensure the system is correctly setup for web content delivery and is tuned for optimal performance.
      - Installing
      - Configuring
      - Managing user data
      - Chapter 4, “Configuring Web Content Manager,” on page 59

3. Based on the information architecture and security architecture defined in the project design document, the WebSphere Portal administrator:
   a. Create all pages required by the web content system.
   b. Adds all required web content viewer portlets to the appropriate pages.
      - Working with managed pages
      - “Displaying content with web content viewers” on page 547

4. The WebSphere Portal administrator configures and enables syndication.
   - Syndication concepts
   - Working with syndicators and subscribers

5. Final testing and tuning of the delivery environment is undertaken by all administrators.

The delivery environment is now ready to use.

### Final steps

Once your environments have been installed, the authoring system completed, added default content to your site, and fully tested the system, you are ready to go live.
1. Ensure your stakeholders are ready to use your Web site and authoring system:
   a. Ensure your content authors have had training on how to use the new web content authoring system.
   b. Ensure any legacy authoring or delivery systems are redirected to the new system.
   c. Launch marketing campaigns to bring visitors to the new website.

2. Based on the maintenance architecture defined in the project design document, you perform regular monitoring and maintenance tasks to ensure your system is healthy and operating efficiently.
   - “Maintenance architecture” on page 50
   - Chapter 7, “Maintaining web content,” on page 583
IBM Web Content Manager is installed using the WebSphere Portal installation interface. Complete installation instructions are included in the WebSphere Portal product documentation. Follow the installation instructions for WebSphere Portal, then configure the authoring and delivery environments based on your needs and instructions provided in this documentation. Migration instructions are also included in the WebSphere Portal product documentation; Web Content Manager is migrated as part of the overall WebSphere Portal migration process.
Chapter 4. Configuring Web Content Manager

Set up a content server by installing IBM Web Content Manager in various deployments to provide robust and flexible environments for web content development and delivery. After you install the content server, there are additional configuration steps you need to complete, according to the role that the server plays in your web content environment.

Configuration for Web Content Manager is managed through services defined as resource environment providers in IBM WebSphere Application Server. For each service you can edit existing properties and add new properties through the WebSphere Integrated Solutions Console.

Related tasks:
- Setting service configuration properties

IBM WebSphere Portal comprises a framework of services to accommodate the different scenarios that portals need to address. Services are available for both WebSphere Portal and IBM Web Content Manager. You can configure some of these services.

Related reference:
- Web Content Manager configuration services

Configuration services for IBM Web Content Manager contain settings for the general operation of the web content system, including settings for messaging, pre-rendering, and searching.

Configuring a web content authoring environment

Set up the authoring environment by installing the authoring portlet and enabling other features required to support the authoring environment.

Installing the authoring portlet

Pages that include the Web Content Manager authoring portlet and the local rendering portlet are created when you first install WebSphere Portal. You only run the authoring portlet configuration task if you have previously uninstalled your authoring or local rendering portlets. The authoring portlet configuration task will automatically create Web Content Manager pages and install the authoring portlet and local rendering portlets.

Installing on AIX®, IBM i, Linux, Solaris, and Windows

1. Open a command prompt.
2. Run the configure-wcm-authoring task from the \wp_profile_root\ConfigEngine directory.
AIX Linux Solaris
./ConfigEngine.sh configure-wcm-authoring
-DPortalAdminPwd=password -DWasUserid=username
-DWasPassword=password

IBM i ConfigEngine.sh configure-wcm-authoring -DPortalAdminPwd=password
-DWasUserid=username -DWasPassword=password

Windows
ConfigEngine.bat configure-wcm-authoring
-DPortalAdminPwd=password -DWasUserid=username
-DWasPassword=password

3. Log out of the portal and log back in.
4. Select Web Content from the product banner to access the authoring portlet.

Cluster note: If the authoring portlet does not display after installing in a cluster, you might need to activate the portlet.

Installing on z/OS®
1. Start the Customization Dialog.
2. Navigate to the Configure Web Content Manager panel: Configure WebSphere Portal > WebSphere Portal for z/OS > Application Configuration Tasks > Configure Web Content Management.
4. Select Define variables.
5. Select the option for the variables you want to define, and follow the panels to define the variables.
6. Generate the customization jobs.
7. Follow the Customization Dialog instructions for submitting the customization jobs.

Cluster note: If the authoring portlet does not display after installing in a cluster, you might need to activate the portlet.

Locale consistency

The language displayed in the authoring portlet is determined by the region or locale of the user. There are, however, some elements of the authoring portlet that use WebSphere Portal functions, such as date selection fields. These will be displayed using the locale of the WebSphere Portal server. For this reason, the language and locales of the site being created, the client and server should be consistent.

If your site contains content in different languages, then a separate Web Content Manager authoring applications should be setup for each language on different WebSphere Portal Servers. These can then be combined into one site using a staging server.

Note: If a user changes their locale, any currently opened Web Content Manager dialogs will be closed. A user will also have to start a new session before it is displayed using the new locale. It is best practice to have the client's correct locale set prior to using Web Content Manager.
Related tasks:
Managing portlets in your cluster

Because all WebSphere Portal servers in the cluster share the same database, any node can be used to manage portlets. Cluster nodes do not need to be stopped when managing portlets. When you deploy a portlet, WebSphere Portal stores the portlet configuration data in the WebSphere Portal database and then forwards the portlet application's Web module and associated configuration to the deployment manager. The deployment manager is responsible for pushing the Web module to each node in the cluster.

**Additional authoring portlet configuration options**

Additional authoring portlet configuration options can be specified using the portlet administration view.

To add additional authoring portlet configuration parameters:

1. Open the portal administration view.
2. Go to **Administration > Portlet Management > Portlets**.
3. Search for the web Content Authoring portlet.
4. Open the configuration view.

You can add any of the following optional configuration parameters:

*Table 12. Additional authoring portlet configuration parameters*

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>htmlfield.rows</td>
<td>Defines the number of rows to display in an HTML field used in an element design or presentation template. If not specified, the default setting of 15 rows is used.</td>
</tr>
<tr>
<td>htmlfield.columns</td>
<td>Defines the width of an HTML field used in an element design or presentation template. The width is defined as the number of characters to display per row. If not specified, the default setting of 85 characters is used.</td>
</tr>
<tr>
<td>htmlfield.embedded.rows</td>
<td>Defines the number of rows to display in an HTML field used in an element design or presentation template, but not an HTML component. If not specified, the number of rows defined using htmlfield.rows is used.</td>
</tr>
<tr>
<td>htmlfield.embedded.columns</td>
<td>Defines the width of an HTML field used in an element design or presentation template, but not an HTML component. The width is defined as the number of characters to display per row. If not specified, the number of characters defined using htmlfield.columns is used.</td>
</tr>
<tr>
<td>htmlfield.htmlcomponent.rows</td>
<td>Defines the number of rows to display in the HTML field used in an HTML component. If not specified, the number of rows defined using htmlfield.rows is used.</td>
</tr>
<tr>
<td>htmlfield.htmlcomponent.columns</td>
<td>Defines the width of the HTML field used in an HTML component. The width is defined as the number of characters to display per row. If not specified, the number of characters defined using htmlfield.columns is used.</td>
</tr>
<tr>
<td>htmlfield.presentation.rows</td>
<td>Defines the number of rows to display in the HTML field used in a presentation template. If not specified, the number of rows defined using htmlfield.rows is used.</td>
</tr>
</tbody>
</table>
Table 12. Additional authoring portlet configuration parameters (continued)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>htmlfield.presentation.columns</td>
<td>Defines the width of the HTML field used in a presentation template. The width is defined as the number of characters to display per row. If not specified, the number of characters defined using htmlfield.columns is used.</td>
</tr>
<tr>
<td>EDIT_LIVE_CUSTOM_LICENSE</td>
<td>Use this is enter a custom license key to use in place of the OEM license for Ephox EditLive using this format: Account ID,Domain,Expiration,License Key,Licensee,Product,Release</td>
</tr>
</tbody>
</table>

**Note:** All users will need to logoff and login before any configuration changes will appear in the authoring portlet.

**Web content authoring options**

You can tailor the authoring behavior of your web content environment by changing configuration settings such as workflow, profiling, and version control.

You define and manage authoring options in the WCM WCMConfigService service using the WebSphere Integrated Solutions Console.

**Enabling workflows**

As default, the IBM Web Content Manager application will workflow content items only. You can update the WCM WCMConfigService service to enable workflows for different items.

To enable workflows, create a new property for the item type to which you want to apply workflow, and specify a value of com.aptrix.pluto.workflow.WorkflowControl for the property. You can enable workflow for the following item types:
- Content items (control.Content)
- Presentation templates (control.Style)
- Authoring templates (control.Template)
- Taxonomy items (control.Taxonomy)
- Categories (control.Category)
- Site area items (control.SiteArea)
- Library components (control.Cmpnt)

For example to enable workflow for authoring templates, you would add the following property:
- Property name: control.Template
- Value: com.aptrix.pluto.workflow.WorkflowControl

To disable workflow for an item type, you set the property to “false”. For example, to disable workflow for authoring templates, you would do the following:
- Property name: control.Template
- Value: false

**Note:** If workflows are enabled for the following items, a workflow view will not be available in the item views navigator.
- Site areas.
• Taxonomies and categories.
• Workflows, workflow stages, or workflow actions.

Individual items can still be moved through workflow stages by accessing them through the normal item views and approving them.

**Note:** Only content items can be moved through a workflow using the web content API. If you enable workflows for other item types, you will not be able to approve or reject these items using the API.

**Enabling profiling**

As default, the Web Content Manager application will profile content items only. You can update the WCM WCMConfigService service to enable profiling for different items.

To enable profiling, create a new property for the item type to which you want to apply profiling, and specify a value of com.aptrix.pluto.taxonomy.ProfileControl for the property. You can enable workflow for the following item types:

• Content items (control.Content)
• Presentation templates (control.Style)
• Authoring templates (control.Template)
• Taxonomy items (control.Taxonomy)
• Categories (control.Category)
• Site area items (control.SiteArea)
• Library components (control.Cmpnt)

For example to enable profiling for components, you would add the following property:

• Property name: control.Cmpnt
• Value: com.aptrix.pluto.taxonomy.ProfileControl

To disable profiling for an item type, you set the property to "false". For example, to disable profiling on components, you would do the following:

• Property name: control.Cmpnt
• Value: false

**Version control options**

By default version control is enabled with the following properties:

• versioningStrategy.AuthoringTemplate
• versioningStrategy.Component
• versioningStrategy.Content
• versioningStrategy.PresentationTemplate
• versioningStrategy.Taxonomy
• versioningStrategy.Workflow
• versioningStrategy.Default

You can use the following values to specify version control settings:
always

A version is saved every time a non-workflowed item is saved, or every time a workflowed item is published.

manual

Versions will only be saved when a user with at least editor access chooses to save a version. This setting causes the following changes in the interface:

- The **Save Version** button is available in the read mode of non-workflowed items and in workflowed items in the published state.
- The **Save and Version** button is available in the edit mode of non-workflowed items and in workflowed items in the published state.

never

Disable version control for an item type.

If a version control strategy is not defined for an item type, then the version control strategy specified in by the `versioningStrategy.Default` property is used.

**Inheritance options**

By default, inheritance is automatically propagated down to each item. You can disable automatic inheritance by specifying the following property:

- Property name: `default.inherit.permissions.enabled`
- Value: `false`

When this setting is specified, it is applied only to new items. The inheritance on existing items will remain unchanged.

**Hierarchical item locking options**

When a content item is being edited, it is locked. Other users are prevented from editing the content item until it is unlocked. Locking of site areas, taxonomies and categories is configurable and is not enabled by default. To enable locking for hierarchical item types, specify the following properties: change the following parameters to "true":

<table>
<thead>
<tr>
<th>Property name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>wcm.authoringui.lock.taxonomies</code></td>
<td>true</td>
</tr>
<tr>
<td><code>wcm.authoringui.lock.categories</code></td>
<td>true</td>
</tr>
<tr>
<td><code>wcm.authoringui.lock.siteareas</code></td>
<td>true</td>
</tr>
<tr>
<td><code>wcm.authoringui.lock.projects</code></td>
<td>true</td>
</tr>
</tbody>
</table>

When locking is enabled for site areas, you cannot create any children within the locked site area. For example, if a site area is locked, you cannot create any new site areas or content items within that site area until it is unlocked. This applies only to direct children of the locked parent. Items that are descendants of the children of a locked parent are not affected.

**Defining valid mime types for the image element**

You define the mime types of files that are allowed to be uploaded into the image element using the `imageresourcecmpnt.allowedmimetypes` property and a list of mime types for the value. For example:

- Property name: `imageresourcecmpnt.allowedmimetypes`
- Value: `image/gif,image/jpeg`
This will prevent users uploading non-image files into the image element.

**Active content filtering**

Active content filtering provides the ability to strip specified HTML fragments from HTML entered in elements. This includes rich text and HTML elements. Active content filtering is configured using the `active.content.filtering.enable` property. By default, active content filtering is enabled. If enabled, this will prevent a user from introducing malicious code into a website such as cross site scripting.

For example, if a user entered this code into an HTML element:

```
Welcome
<a href="javascript:window.alert("boo!")">my link</a>
<script language="javascript">window.alert("boo 2!")</script>
Click the link for a surprise.
```

It would be changed to this when saved:

```
Welcome
<a href="#" active content removed --&gt;">my link</a>
"" active content removed --&gt;
Click the link for a surprise.
```

**Setting the default child placement position**

You can set the parameter `wcm.authoringui.childPlacementDefault` to specify the default placement of new content items.

<table>
<thead>
<tr>
<th>Property value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>start</td>
<td>This setting will, by default, place a new content item as the first content item within a site area.</td>
</tr>
<tr>
<td>end</td>
<td>This setting will, by default, place a new content item as the last content item within a site area.</td>
</tr>
</tbody>
</table>

- If this parameter is not set, the default child position will be "end".
- The default placement position specified in an authoring template will override this setting for content items created with that authoring template.

**Setting the size of the breadcrumb library dropdown**

You can set the parameter `wcm.authoringui.breadcrumbLibrariesMaximum` to specify the number of libraries that should be shown in the authoring interface breadcrumb. For example, `wcm.authoringui.breadcrumbLibrariesMaximum=16`

- If this parameter is not set, only the first 10 libraries are displayed.
- The value of this parameter must be an integer between 5 and 50.
- It is recommended that its value should be between 10 and 20.
- If more than this number of libraries exist, the remaining libraries are accessible using the Select from all libraries option.

**Expired items**

By default, expired items are displayed alongside published and draft items.
To determine if expired items are listed in views, you can specify the `wcm.authoringui.showexpireditems` property in the WCM WCMConfigService service using the WebSphere Integrated Solutions Console:

- If set to true, expired items are displayed alongside published and draft items.
- If set to false, only published and draft items are displayed.
- If not specified, this setting defaults to true.

**Related tasks:**
- [Setting service configuration properties](#)

IBM WebSphere Portal comprises a framework of services to accommodate the different scenarios that portals need to address. Services are available for both WebSphere Portal and IBM Web Content Manager. You can configure some of these services.

### Configuring authoring portlet search

You can change the configuration of the authoring portlet to change how search works.

You define and manage authoring portlet search options in the WCM WCMConfigService service using the WebSphere Integrated Solutions Console.

You can specify the following properties:

**wcm.authoringui.advancedsearch.searchonselection**

- Possible values are true or false.
- If set to true, when you click **Advanced Search**, an advanced search will automatically be executed based on any text currently entered in the basic search. If nothing has been entered in the basic search, advanced search is not automatically executed.
- If set to false, advanced search will not automatically be executed if there is existing text in the basic search field.
- If the property is not specified, this setting will default to the false behavior.

**wcm.authoringui.simplesearch.addstar**

- Possible values are true or false.
- If set to true, a wildcard character is added to the end of text entered in the basic search. For example, searching for Span will automatically search for Span* and will display search results that have a title, description or keywords that begin with the word Span such as Spanish.
- If set to false, only exact matches to the text entered in the basic search field will be searched for.
- If the property is not specified, this setting will default to the false behavior.

**wcm.authoringui.advancedsearch.addstar**

- Possible values are true or false.
- If set to true, a wildcard character is added to the end of text entered in the advanced search. For example, searching for Span will automatically search for Span* and will display search results that have a title, description or keywords that begin with the word Span such as Spanish.
- If set to false, only exact matches to the text entered in the advanced search field will be searched for.
• If the property is not specified, this setting will default to the false behavior.

Related tasks:
Setting service configuration properties

IBM WebSphere Portal comprises a framework of services to accommodate the different scenarios that portals need to address. Services are available for both WebSphere Portal and IBM Web Content Manager. You can configure some of these services.

Importing large files and images

Because importing large files into IBM Web Content Manager can have a negative impact on performance, you can adjust several settings to ensure better performance when importing files.

Before updating the settings for large file handling, review the following considerations:

UNIX note: If you are running on a UNIX operating system, ensure that you have used the ulimit -f command to set the maximum size of files that can be created to be at least the size of the largest file you would need to upload to the content server. The command ulimit -f unlimited removes any limit on file size. When setting the size, also make sure the system has sufficient disk space to support the setting.

Disk space requirements: When importing files, a temporary directory is used to store the files during the upload process. If the size of the uploaded files exceeds the available disk space for the temporary directory, the import operation fails. When uploading large files, ensure that there is sufficient disk space to accommodate the import. The location of the temporary directory is specified by the jcr.binaryValueFileDir property in the wp_profile_root/PortalServer/jcr/lib/com/ibm/icm/icm.properties file.

1. Log in to the WebSphere Integrated Solutions Console.
2. Click Resources > Resource Environment > Resource Environment Providers > WCM WCMConfigService > Custom properties.

Cluster note: If you are using this web content server as part of a cluster, ensure that you use the WebSphere Integrated Solutions Console for the deployment manager when manipulating configuration properties.

3. For the resource.maxUploadSize property, specify a value in megabytes corresponding to the size of the largest file that you want to allow to be imported. For example, if you do not want to allow files larger than 34 MB to be imported, update the resource.maxUploadSize property to have a value of 34. Although it is recommended that this value not exceed 100 MB, you cannot upload files larger than 512 MB.

4. For the resourceserver.maxCacheObjectSize property, specify a value of 300 KB or less.

5. Add the transaction.sync.remove property, and specify a value of true.

6. Click Servers > Server Types > WebSphere application servers > portal_server > Server infrastructure > Administration > Custom Properties

7. Add the protocol_http_large_data_inbound_buffer property, and for the value specify the maximum file size in bytes. This value should correspond to the value you set for the resource.maxUploadSize property in the WCM WCMConfigService service.
Note that the protocol_http_large_data_inbound_buffer property uses bytes.
So if you specified a value of 34 MB for the resource.maxUploadSize property,
you would specify a value of 35651584 bytes for the
protocol_http_large_data_inbound_buffer property.

8. Click Resources > JDBC > Data sources > datasource_name > Custom properties

9. Specify the fullyMaterializeLobData property with a value of false.

10. Click Resource > JDBC > Data sources > datasource_name > Connection pool properties.

11. Increase the maximum number of database collections allowed for the application server by increasing the value of the Maximum connections field to a value greater than the default 50 connections.

12. If you are working with files larger than 100 MB, increase the web containers transaction timeout setting.
   a. Click Servers > Server Types > WebSphere application servers > portal_server > Container Services > Transaction service.
   b. Increase the value of the Total transaction lifetime timeout setting from the default setting of 120 seconds.

13. Increase the maximum number of threads allowed in the thread pool used by the web container.
   a. Click Servers > Server Types > WebSphere application servers > portal_server > Thread pools > WebContainer.
   b. Set the value of the Maximum Size field to 100 threads.

14. If you are using IBM HTTP Server Version 7, increase the connection timeout value for connections to the application.
   a. Click Servers > Server Types > web servers > web_server > plug-in properties > Custom properties > New.
   b. In the name field, enter ServerIOTimeout.
   c. In the value field, enter the timeout value in seconds.
   The default value is 60 seconds. However, when working with large files, this default value is typically insufficient and can cause a false server error response to be sent, which in turn causes the portal to reissue the request. Specify a timeout value that is long enough to allow a failing request to receive a response, or enter -1 for an unlimited timeout value.

15. Click Save to save your configuration changes.

16. Restart the portal for the settings to take effect.

**Note:** If the portal’s policy cache manager indicates that a number of web container threads are hung, set the cacheinstance.com.ibm.wps.policy.services.PolicyCacheManager.lifetime property in the WP CacheManagerService service to a value of -1. This setting reduces the database connections and load times and helps prevent threads from hanging.
Increasing time-outs

If users are experiencing timeout errors when saving items, you can increase the total transaction lifetime timeout setting of your WebSphere Portal server.

The total transaction lifetime timeout setting is changed using the WebSphere Integrated Solutions Console.

Go to Servers > Server Types > WebSphere application servers > portal_server > Container Services > Transaction Service.

The total transaction lifetime timeout setting should be changed to the same amount on all the servers in your web content system.

Alternatives to increasing server time-outs

Increasing the total transaction lifetime timeout setting may not always be the best solution to server time-outs as increasing this setting too much may cause a drop in performance. Timeout errors generated when saving items occur when the current transaction finishes before the item has been saved. If the item you are saving contains large amounts of data, it may be better to redesign the item rather than change the total transaction lifetime timeout setting:

<table>
<thead>
<tr>
<th>Option</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Authoring Templates</td>
<td>If a large number of elements have been added to an authoring template, you may experience a timeout error when saving. Instead of using a single authoring template, create multiple authoring templates that contain only those elements required for a specific task.</td>
</tr>
<tr>
<td>Presentation templates and components</td>
<td>You may receive timeout errors when saving presentation templates or components that contain large amounts of HTML or rich text in their designs. You should instead create multiple HTML or rich text components and then reference these in the presentation templates or component designs.</td>
</tr>
<tr>
<td>Site areas and content items</td>
<td>You may receive timeout errors when saving site areas and content items that contain elements that use large amounts of HTML. You should instead create multiple HTML or rich text components and then reference these in element designs. If a large number of elements have been added to a site area or content item, you may also experience a timeout error when saving. In this case, you should reduce the number of elements stored in the site area or content item.</td>
</tr>
<tr>
<td>Downloadable files</td>
<td>Another alternative to creating web content containing large amounts of HTML or rich text is to provide information on your website in the form of downloadable files. These can be stored as file resource elements.</td>
</tr>
</tbody>
</table>
**Configuring remote server access for links**

Before you can add links to files and documents stored in remote content management systems into web content elements, you must configure your server with information about the remote system and the settings used to handle communication with the system.

**IBM Content Manager and SSL usage:** IBM Content Manager Services for Lotus Quickr provides the capability to link to documents stored in IBM Content Manager. These links are generated according to the base service URL configured in IBM Content Manager Services for Lotus Quickr, as specified by the urlBaseService property in the cmpathservice.properties file. If you have enabled Secure Sockets Layer (SSL) encryption for your portal, verify that the base service URL reflects the https protocol and not the http protocol to ensure that links are generated correctly. Refer to the documentation for IBM Content Manager Services for Lotus Quickr for details on how to update the urlBaseService property.

To prevent linking to unsafe servers, you need to specify a list of allowed domains that your portal can access using the portal’s AJAX proxy component. You can use the global AJAX proxy configuration to customize the outgoing HTTP traffic, such as applying specific HTTP timeout values or configuring an outbound HTTP proxy server. To do this, map the URL patterns for the ECM server to the federated_documents_policy dynamic policy using the WP ConfigService configuration service.

1. Log in to the WebSphere Integrated Solutions Console.
2. Click **Resources > Resource Environment > Resource Environment Providers**.
3. Click **WP ConfigService**.
5. Click New, and enter the property name

   `wp.proxy.config.urlreplacement.federated_documents_policy.suffix`, and set

   the string value to the URL pattern of the ECM server.

   For example, to enable the server to access information from the ECM server `ecm.example.com` on port 10038 over HTTP, you would add the following property:

   `wp.proxy.config.urlreplacement.federated_documents_policy.1=http://ecm.example.com:10038/*`

   **Note:** The value of the property key `suffix` can be any value as long as it is unique within the set of keys mapping to the federated_documents_policy.

6. Create additional properties as needed for any other ECM servers that you need to access through the server.
7. Save your changes, and restart the portal server.

If a user tries to access a server (for example, `www.example.com`) that has not been added to the list of allowed domains, the following message is displayed:

`Access to remote server www.example.com has not been granted. Please contact your system administrator.`
Setting service configuration properties

IBM WebSphere Portal comprises a framework of services to accommodate the different scenarios that portals need to address. Services are available for both WebSphere Portal and IBM Web Content Manager. You can configure some of these services.

“Inserting a link to remote content” on page 227

You can insert links to remote content into elements containing a rich text field by using the Insert Link to Remote Document button in the rich text editor. Only remote content that is configured for remote server access can be selected by using this button.

Related reference:

Related tasks:

Setting service configuration properties

Inserting a link to remote content

You can insert links to remote content into elements containing a rich text field by using the Insert Link to Remote Document button in the rich text editor. Only remote content that is configured for remote server access can be selected by using this button.

Setting up support for federated documents

Before you can access metadata from federated documents, you need to configure access to the remote servers containing the documents and specify information about the feeds or service documents used to retrieve the documents. You can also tune the cache settings that are used with the federated documents feature.

Authentication requirement: Before you can use the federated documents feature, you must complete one of the following steps:

• Enable single sign-on (SSO) in IBM WebSphere Application Server between the portal server and the content management system.

• Use a content management system that supports HTTP basic authentication, and enable a credential vault slot that stores the credentials to authenticate with the remote server.

Important: If you are setting up single sign-on between IBM Lotus Quickr and your portal server, export the SSO key from the Lotus Quickr server and import it into the portal server, rather than the other way around.

Configuring access to remote systems for federated documents

To retrieve metadata information for documents on remote content management systems, configure the federated documents feature with information about the remote system and the settings used to handle communication with the system.

Because the federated documents feature uses the AJAX proxy component to access the remote content management system, you can use the global AJAX proxy configuration to customize the outgoing HTTP traffic, such as applying specific HTTP timeout values or configuring an outbound HTTP proxy server. You must list the individual content management servers to be accessed through the federated documents feature as allowed request targets in the AJAX proxy configuration, and enable LTPA cookie forwarding for those requests. To do this, map the URL patterns for the content management server to the federated_documents_policy dynamic policy using the WP ConfigService configuration service.

1. Log in to the WebSphere Integrated Solutions Console.
2. Click Resources > Resource Environment > Resource Environment Providers.
3. Click WP ConfigService.

5. Click New, and enter the property name `wp.proxy.config.urlreplacement.federated_documents_policy.suffix`, and set the string value to the URL pattern of the content management server.

For example, to enable the federated documents feature to access information from the content management server `ecm.example.com` on port 10038 over HTTP, you would add the following property:

```
wp.proxy.config.urlreplacement.federated_documents_policy.1=http://ecm.example.com:10038/*
```

**Note:** The value of the property key `suffix` can be any value as long as it is unique within the set of keys mapping to the `federated_documents_policy` dynamic policy.

6. Create additional properties as needed for any other content management servers that you need to access through the federated documents feature.

7. Optional: The federated documents feature can also consume arbitrary ATOM feeds. To enable this, you can map the URL prefix of the ATOM feed to the `default_policy` dynamic policy.

   a. Click New, and enter the property name `wp.proxy.config.urlreplacement.default_policy.suffix`, and set the string value to the URL pattern of the server providing the ATOM feed.

   For example, to enable the federated documents feature to access ATOM feeds from the server `www.example.com`, you would add the following property:

   `````
wp.proxy.config.urlreplacement.default_policy.1=http://www.example.com/*
``````

The value of the property key `suffix` can be any value as long as it is unique within the set of keys mapping to the `default_policy` dynamic policy.

**Important:** To prevent security token forwarding to untrusted servers, be sure that you do not use the `federated_documents_policy` dynamic policy for those servers.

   b. Create additional properties as needed for any other ATOM feed servers that you need to access through the federated documents feature.

8. Save your changes, and restart the portal server.

**Related tasks:**

Setting service configuration properties

IBM WebSphere Portal comprises a framework of services to accommodate the different scenarios that portals need to address. Services are available for both WebSphere Portal and IBM Web Content Manager. You can configure some of these services.

Configuring the federated documents feature

Configure the federated documents feature to specify information about the source servers for the documents that are available to users.

When the portal retrieves documents from a remote server, authentication might be required to access the documents on the remote server. You can use several types of authentication:

- Single sign-on (SSO) between the portal and the remote server
- User name and password information in the user interface. Only HTTP basic authentication is supported for CMIS servers.
Credential vault slots that handle HTTP authentication

In addition to enabling or disabling credential vault slots for authentication, you can identify the servers providing documents. For each server, you can define characteristics such as the type of document that the server returns and the title used to identify the server.

1. Log in to the WebSphere Integrated Solutions Console.
2. Click Resources > Resource Environment > Resource Environment Providers.
3. Click WP FederatedDocumentsService.
5. Specify whether credential vault slots are used for authentication with remote servers.

Because you can access federated documents through either the personalization editor or the rich text editor provided with Web Content Manager, you can configure credential vault slots for each method independently.

a. If you are accessing federated documents through the personalization editor, click wp.federated.documents.pzn.vaultselection.enabled. To enable credential vault slots, set the value to true, or, to disable credential vault slots, set the value to false. By default, the value is true.

b. If you are accessing federated documents through the rich text editor in Web Content Manager, click wp.federated.documents.wcm.vaultselection.enabled. To enable credential vault slots, set the value to true, or, to disable credential vault slots, set the value to false. By default, the value is true.

If you enable credential vault slots, users can select a credential vault slot in the user interface. You can also use the property wp.federated.documents.suffix.vault.slot to specify a default credential slot to be used with a given remote server.

6. Specify whether users can enter their own servers when accessing remote content or can use only predefined servers that you configure.

a. Click wp.federated.documents.custom.server.enabled.

b. To allow users to enter their own servers, set the value to true. To prevent users from entering custom servers, set the value to false. When set to false, the user interface does not display the entry field for custom servers. By default, the value is true.

7. Specify whether documents from servers supporting Document Services remote interfaces can be retrieved by the portal. Examples of products that support Document Services remote interfaces include IBM Lotus Quickr, IBM Content Manager, and IBM FileNet Content Manager.

a. Click wp.federated.documents.document.services.enabled.

b. To enable access to Document Services feeds, set the value to true. To disable access to Document Services feeds, set the value to false. If set to false, users can still access servers supporting CMIS or Atom feeds, but connections to Document Services servers are not supported. By default, the value is true.

8. For each remote server that contains documents you want to access from the portal, configure the server URL, feed type, and additional optional properties.
The value of the *suffix* portion of the property key is used to group related properties for each server. Use the same *suffix* value for properties related to the same server. The *suffix* can be any value as long as it is unique across the property keys.

For each property, click **New** and enter the name and value:

**wp.federated.documents.suffix.url**
- **Value:** The URL for an Atom feed or CMIS service document for the remote server. This property is required.

**wp.federated.documents.suffix.type**
- **Value:**
  - CMIS indicates that the remote server provides a CMIS service document.
  - DocumentServices indicates that the remote server supports Document Services remote interfaces.
  - ATOM indicates that the remote server provides a generic Atom feed.

If no value is specified, a default value of CMIS is used.

**wp.federated.documents.suffix.title.default**
- **Value:** The title used to identify this source server in the user interface, when there is no resource bundle defined to provide title text. If no default title and no resource bundle are defined, the value of the *wp.federated.documents.suffix.url* property is used in the user interface.

**wp.federated.documents.suffix.nls.resources**
- **Value:** The name of the resource bundle that contains the translated title and description used to identify this source server in the user interface. If this property is not defined, the default title is used. If no default title and no resource bundle are defined, the value of the *wp.federated.documents.suffix.url* property is used in the user interface.

**wp.federated.documents.suffix.vault.slot**
- **Value:** The name of the credential vault slot that stores the credentials used for authentication with the remote server. Credential vault slots are set up and managed by the portal administrator. This property defines the default credential vault slot that is predefined in the user interface, although the user can also select a different slot if one is available. If this property is not defined, the user interface does not display a default credential vault slot, but you can still select a slot from the available list. This property is optional.

**Note:** The credential vault slot must contain the credentials that are required for authentication with the remote server.

**wp.federated.documents.suffix.override.authentication.enabled**
- **Value:** true or false. When set to true, the user can change the authentication method for the server in the user interface. When set to false, the user interface does not display the field to change the authentication method. The default value is true.

9. Optional: Configure the amount of data returned for the summary metadata attribute of the document.
   a. Click **Resources > Resource Environment > Resource Environment Providers.**
   b. Click **WCM WCMConfigService.**
c. Under Additional Properties, click Custom Properties.
d. Click `wcm.pzn.ecm.max.field.length`, and enter the number of characters to be returned. If no value is specified, the default value is 128 characters.

10. Optional: Configure whether property changes are automatically loaded.
By default, the Federated Documents service automatically reloads properties at a specified interval, without requiring you to restart the portal. You can change the automatic reloading behavior or modify the reloading interval.
   b. Click WP FederatedDocumentsService.
   c. Under Additional Properties, click Custom Properties.
   d. Click `wp.federated.documents.document.service.reload.disabled`, and specify a value of true to disable automatic reloading of properties. The default value is false.
   e. Click `wp.federated.documents.document.service.reload.interval`, and specify the interval in seconds for reloading properties. The default value is 3 seconds.

11. Save your changes. The Federated Documents service automatically reloads any updated properties. If you have disabled automatic reloading, restart the portal server.

12. If you enable credential vault slots, grant access to credential vault slots for all authenticated users.
   a. Log in to the portal as an administrator.
   b. Click Administration > Access > Resource Permissions.
   c. From the list of resource types, navigate to Virtual Resources.
   d. For the `ADMIN_SLOTS` resource, click the Assign Access icon.
   e. Edit the User role, and add the All Authenticated Portal Users group to the role.

**Cache tuning for federated documents**
The federated documents feature uses the document list cache, the document data cache, and the feed type cache to manage information about the list of documents, the document data, and the types of feeds a server provides.

- The document list cache contains the list of document identifiers contained in the rule selection result of a specific user and a specific selection rule. The cache is activated by default with a default cache entry lifetime of 10 minutes.
- The document data cache contains the metadata of a specific document. The cache is activated by default with a default cache entry lifetime of 10 minutes.
- The feed type cache contains the type of feed for a given feed URL. The feed type can be Document Services, CMIS, or ATOM. The cache is activated by default with a default cache entry lifetime of 24 hours.

To tune these caches you can configure the Cache Manager Service (`WP CacheManagerService`) in the WebSphere Integrated Solutions Console using the following properties:

- Feed type cache: `cacheinstance.com.ibm.pzn.wcm.ecm.FeedTypeServerCache`
Updates occurring on the remote content management system might not immediately be reflected on the portal side if there is a corresponding entry found in the cache. The individual cache life time values determine the maximum time lag for corresponding updates.

**Note:**
- The time lag for new documents becoming visible and deleted documents being removed depends on the lifetime value for the configured document list cache.
- The time lag for updates in the metadata describing a document (for example, changes to the document title) depends on the configured lifetime value for the document list cache.

The user-specific document list cache is explicitly invalidated each time the user logs in, so that the most current list of available document identifiers is available upon login.

**Related tasks:**
- Setting service configuration properties

IBM WebSphere Portal comprises a framework of services to accommodate the different scenarios that portals need to address. Services are available for both WebSphere Portal and IBM Web Content Manager. You can configure some of these services.

### Configuring a web content staging environment

Configure the staging environment to emulate the web content delivery environment and allow for testing before deployment.

You define and manage staging environment options in the WCM WCMConfigService service using the WebSphere Integrated Solutions Console.
- If your staging server is to be used purely as a holding server where changes to your site are accumulated prior to syndicating these changes to a delivery environment, then you may only need to review the syndication settings of the staging server. In most cases you would ensure that automatic syndication is disabled.
- If you are using your staging environment for user acceptance testing prior to syndicating to a delivery environment then you will need to ensure that all other settings configured on your staging server match those set on the delivery server.

**Related tasks:**
- Setting service configuration properties

### Configuring a web content delivery environment

Set up your delivery environment by installing web content viewers and enabling any other required features.

**Setting up site analysis for the web content viewer**

To track usage data for the web content viewer, you can configure the portal for site analysis logging for the web content viewer.
Enabling the web content viewer logger

To take advantage of the site analysis logging available for the web content viewer, you need to configure the WP SiteAnalyzerLogService service and activate the SiteAnalyzerJSRPortletLogger service.

Before you activate the SiteAnalyzerJSRPortletLogger logger, you must ensure that site analysis has been enabled for the portal in general, as described in [Logging and analyzing server side site data](#).

1. Log in to the WebSphere Integrated Solutions Console.
2. Click Resources > Resource Environment > Resource Environment Providers.
3. Click WP SiteAnalyzerLogService.
4. Activate the SiteAnalyzerJSRPortletLogger logger through the WP SiteAnalyzerLogService by defining the parameter SiteAnalyzerJSRPortletLogger.isLogging and by setting the parameter value to true.
5. Save your changes, and restart the portal.

Site analysis example for the web content viewer

The site analysis log uses the NCSA Combined log format, which is a combination of NCSA Common log format and three additional fields: the referrer field, the user_agent field, and the cookie field. This example describes typical site analysis logging information for the web content viewer.

The IBM WebSphere Portal site analysis log is:

```
wp_profile_root/logs/WebSphere_Portal/sa_date_time.log
```

where `date_time` is the date and time the file was created. The current (active) log file is named `sa.log`.

**Note:** The WP SiteAnalyzerService might be configured to use different filenames.

The following example displays a sample entry in the site analysis log as it is written by the web content viewer if the SiteAnalyzerJSRPortletLogger is enabled.

```
9.37.3.88 - jdoe [22/Nov/2008:22:11:27 +0100] "GET /Portlet/ 5_8000CB1A0OU6B02NVSPH1G20G1/Web_Content_Visitor_(JSR_286)/Web%20Content%2fSite01%2fSiteArea01%2fSiteContent01 HTTP/1.1" 200 -1 "http://myserver.company.com/Page/ 6_8000CB1A0OU402FOJC25U1025/MyPage" "Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.8.1.18) Gecko/20081029 Firefox/2.0.0.18" "JSESSIONID=0000JwImO4x7btVlzWcJ9Qo-uji:1"
```

The table describes each field of the log format:
Table 14. Explanation of each field in the log format

<table>
<thead>
<tr>
<th>Field in the Example</th>
<th>Log Field Name and Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>9.37.3.88</td>
<td><strong>host</strong>: The IP address of the HTTP client that sent the request. <strong>Important</strong>: If there is a reverse proxy server between the client and the portal, the IP address logged is that of the reverse proxy server rather than the HTTP client. To log the IP address of the HTTP client, you must remove the reverse proxy server from the environment.</td>
</tr>
<tr>
<td>-</td>
<td><strong>rcf931</strong>: The identifier used to identify the client making the request. If the client identifier is not known, the field is set to the hyphen character (-).</td>
</tr>
<tr>
<td>jdoe</td>
<td><strong>username</strong>: The user ID for the client. If the user ID is not known, the field is set to the hyphen character (-).</td>
</tr>
<tr>
<td>Field in the Example</td>
<td>Log Field Name and Explanation</td>
</tr>
<tr>
<td>----------------------</td>
<td>--------------------------------</td>
</tr>
</tbody>
</table>
| "GET /Portlet/[…] HTTP/1.1" | **request** The HTTP method, the URI of the requested resource, and the version of HTTP used by the client. The URI is composed of the following elements:  
- The identifier *Portlet*.  
- The ID of the web content viewer instance that is requested.  
- The administrative name of the web content viewer (Note: This name is always the same unless the portlet has been cloned.).  
- The context path of the rendered Web Content Manager item encoded in UTF-8.  
- A query string containing the following parameters:  
  - **PortletPID** The ID of the web content viewer instance that is requested.  
  - **PortletMode** The mode in which the portlet is rendered. Note that the web content viewer writes log entries only in its view mode.  
  - **PortletState** The portlet window state.  
  - **RequestType** The request type (note that the web content viewer writes log entries only for render requests).  
  This is followed by a list of all request parameters that are available to the web content viewer instance as UTF-8 encoded key-value-pairs. |
| 200 | **statusCode** The HTTP status code for the request. |
| -1 | **bytes** The number of bytes of data transferred from the client as part of the request. A value of -1 indicates that the number of bytes is unknown. |
Table 14. Explanation of each field in the log format (continued)

<table>
<thead>
<tr>
<th>Field in the Example</th>
<th>Log Field Name and Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;<a href="http://myserver.company.com/Page/6_8000CB1A00UR402F0JC25U1O25/MyPage">http://myserver.company.com/Page/6_8000CB1A00UR402F0JC25U1O25/MyPage</a>&quot;</td>
<td>referrer</td>
</tr>
<tr>
<td>&quot;Mozilla/5.0 [...]&quot;</td>
<td>user_agent</td>
</tr>
<tr>
<td>&quot;JSESSIONID=0000JwIm04xm7btVLwzCj9Qo-u;j:-1&quot;</td>
<td>cookies</td>
</tr>
</tbody>
</table>

referrer
The referrer in case of portlet site analysis log entries identifies the portal page on which the web content viewer instance is rendered.

user_agent
The type of web browser used by the client.

cookies
The name and value of a cookie that was sent to the client browser as part of the request. If multiple cookies were sent, the list is delimited by the semicolon character.

Related reference:
[Understanding the site analysis log](#)
Learn more details about how to read the site analysis log.

XML configuration interface parameters for the web content viewer

As with other portlets in your portal, you can use the XML configuration interface (xmlaccess command) to deploy and configure the web content viewer. To simplify the configuration of the portlet with the XML configuration interface, the portlet parameters you can specify accept path values in addition to the standard IDs.

By default web content viewer is configured with unique IDs. This has the advantage that the configuration does not break if an item is renamed or moved. However, when configuring a portlet with the XML configuration interface, it can be difficult to determine the unique ID of an item. When configuring the Web content viewer, you can reference web content items by their path, as well as by their IDs, by using the following parameters:

**AUTHORINGTEMPLATE_OVERRIDE**
Specifies the authoring templates of the profile section. The parameter can contain multiple values, separated by commas. The list can contain both ID and path values.

**CATEGORY OVERRIDE**
Specifies the categories of the profile section. To list multiple categories, separate the categories by commas. You can use both ID values and path values.

**SITEAREA OVERRIDE**
Specifies the site areas of the profile section. To list multiple site areas, separate the site areas by commas. You can use both ID values and path values.

**WCM_BROADCASTS_TO**
Specifies the link broadcasting setting for the web content viewer. Values include:
• **WCM_LINKING_DYNAMIC**: Information about the web content item displayed in the web content viewer is used to dynamically determine to which page the context is broadcast.

• **WCM_LINKING_SELF**: The context of the current Web content viewer is broadcast to other web content viewers on the same portal page.

• **WCM_LINKING_OTHER**: The context of the current Web content viewer is broadcast to other web content viewers on another portal page, as specified by the `WCM_PORTAL_PAGE_ID` parameter.

• **WCM_LINKING_NONE**: The context of the current Web content viewer is not broadcast to other web content viewers.

**WCM_COMPONENT_IDR**

Specifies a library component and is only used if content type Component is selected.

**WCM_CONTENT_COMPONENT**

Specifies the name of the element to be displayed, when the `WCM_CONTENT_TYPE` parameter has the value `CONTENT_COMPONENT`.

**WCM_CONTENT_CONTEXT_IDR**

Specifies the content render context. It can be a content item or site area, as specified by the `WCM_CONTENT_CONTEXT_TYPE` parameter.

**WCM_CONTENT_CONTEXT_TYPE**

Specifies the type of the configured content context. Values include:

- **CONTENT**: Indicates that the content context is a content item.
- **PARENT**: Indicates that the content context is a site area.

**WCM_CONTENT_TYPE**

Specifies the item to be displayed. Values include:

- **CONTENT**: Indicates that the item to be displayed is a content item.
- **COMPONENT**: Indicates that the item to be displayed is a component.
- **CONTENT_COMPONENT**: Indicates that the item to be displayed is an element.

**WCM_DESIGN_IDR**

Specifies an alternate presentation template.

**WCM_LISTENS_TO**

Specifies how the web content viewer is configured to receive links broadcast from other web content viewers. Values include:

- **WCM_LINKING_OTHER**: Information is received from any web content viewer broadcasting links.
- **WCM_LINKING_SELF**: Information is received only from this web content viewer.
- **WCM_LINKING_NONE**: No information from other web content viewers is received.

**WCM_PAGE_TITLE**

Used with the `WCM_PAGE_TITLE_TYPE` parameter, this parameter specifies the page title for the web content viewer. Values include:

- The user-defined title for the page, if the `WCM_PAGE_TITLE_TYPE` parameter has a value of `WCM_PAGE_TITLE_TYPE_GENERAL`.
- The name of the resource bundle containing the title for the page, if the `WCM_PAGE_TITLE_TYPE` parameter has a value of `WCM_PAGE_TITLE_TYPE_RESBUN`.
**WCM_PAGE_TITLE_TYPE**

Specifies how the page title is displayed for the web content viewer. Value include:

- **WCM_PAGE_TITLE_TYPE_DEFAULT**: The default title defined in the portal's administration interface is used.
- **WCM_PAGE_TITLE_TYPE_GENERAL**: A user-defined title is used, as specified by **WCM_PAGE_TITLE** parameter.
- **WCM_PAGE_TITLE_TYPE_RESBUN**: The title is defined in a resource bundle, as specified by **WCM_PAGE_TITLE** parameter.
- **WCM_PAGE_TITLE_TYPE_DYN**: The title is defined by the value of the **Display title** field for the content item that is displayed in the web content viewer.

**WCM_PORTAL_PAGE_ID**

Specifies the unique name or object ID of the page which is the target for link broadcasts, when the **WCM_BROADCASTS_TO** parameter is set to **WCM_LINKING_OTHER**.

**WCM_PORTLET_TITLE**

Used with the **WCM_PORTLET_TITLE_TYPE** parameter, this parameter specifies the portlet title for the web content viewer. Values include:

- The user-defined title for the portlet, if the **WCM_PORTLET_TITLE_TYPE** parameter has a value of **WCM_PORTLET_TITLE_TYPE_GENERAL**.
- The name of the resource bundle containing the title for the portlet, if the **WCM_PORTLET_TITLE_TYPE** parameter has a value of **WCM_PORTLET_TITLE_TYPE_RESBUN**.

**WCM_PORTLET_TITLE_TYPE**

Specifies how the portlet title is displayed for the web content viewer. Value include:

- **WCM_PORTLET_TITLE_TYPE_DEFAULT**: The default title defined in the portal's administration interface is used.
- **WCM_PORTLET_TITLE_TYPE_GENERAL**: A user-defined title is used, as specified by **WCM_PORTLET_TITLE** parameter.
- **WCM_PORTLET_TITLE_TYPE_RESBUN**: The title is defined in a resource bundle, as specified by **WCM_PORTLET_TITLE** parameter.
- **WCM_PORTLET_TITLE_TYPE_DYN**: The title is defined by the value of the **Display title** field for the content item that is displayed in the web content viewer.

When specifying a content path, you must begin with the forward slash character (/) followed by the library name, as indicated in the following examples of valid content paths:

/mylib/myfolder/mysitearea/mycontent

or

/mylib/mypresentationtemplate

**Note**: If you configure an item by its path rather than by its ID, the portlet configuration can become invalid if the item is renamed or moved. If an item has been configured by its path, the web content viewer displays a small path icon after the item when you are in the **Edit Shared Settings** or **Configure** mode.
Important: When configuring an item by its path, you cannot build the path from the Display title fields of the items in the path. You must use the Name fields of the items when specifying the path.

Related concepts:
- The XML configuration interface
  Use the XML configuration interface for exchanging portal configurations.

Caching options
Both IBM Web Content Manager generated Web pages and content from external data sources can be cached by the Web Content Manager application. If utilized correctly, Web Content Manager caching can dramatically increase the performance of a site.

Web content cache types
Learn about the types of caching used by IBM Web Content Manager, basic web content caching and advanced web content caching.

Basic web content caching
This is the simplest caching option. The first time a web page is rendered by the Web Content Manager application, it is stored in a cache. Users then access this page from the cache until it expires. Only then is the web page rendered afresh. The main benefit of this scenario is improved performance. Basic caching should only be used on static content that does not require “real-time” access.

Advanced web content caching
There are two major differences between basic caching and advanced caching:
- Advanced caching can cache pages based on different user profiles.
- Cache parameters in connect tags and URL requests can be used to override your server’s default advanced web content caching settings allowing you to set custom cache settings for individual web pages or components.

Table 15. Advanced caching types

<table>
<thead>
<tr>
<th>Advanced caching type</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Site caching</td>
<td>This is the same as the basic web content cache except that cache parameters in connect tags and URL requests can be used to override your server’s default advanced web content caching settings.</td>
</tr>
<tr>
<td>Session caching</td>
<td>When session caching is enabled, a copy of each Web page a user visits is stored in the session cache. The User accesses the cached version of a web page until they start a new session, or until the cached web page is expired from the cache.</td>
</tr>
<tr>
<td>User caching</td>
<td>When user caching is enabled, a copy of each Web page a user visits is stored in the user cache. The user accesses the cached version of a web page until the cached web page is expired from the cache.</td>
</tr>
<tr>
<td>Secured caching</td>
<td>Secured caching is used on sites where the item security features are used to grant different users access to different Web pages and components based on the groups they belong to.</td>
</tr>
</tbody>
</table>
Table 15. Advanced caching types (continued)

<table>
<thead>
<tr>
<th>Advanced caching type</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Personalized caching</td>
<td>Personalized caching is used to cache web pages of users who have the same &quot;personalization profile&quot;. This means that users who have selected the same personalization categories and keywords, and who belong to the same group, share a single cache.</td>
</tr>
</tbody>
</table>

Default web content caching versus custom caching

Cache parameters in connect tags and URL requests can be used to override your server’s default advanced web content caching settings allowing you to set custom cache settings for individual web pages or components.

In most cases, basic, site and session caching would only be used as your server’s default web content cache. User, secured and personalized caching would mostly be used when using custom caching in connect tags and URL requests.

Note: If basic caching is used as your default web content cache, custom caching cannot be used.

Cache comparisons

Table 16. Basic caching versus advanced caching

<table>
<thead>
<tr>
<th>Function</th>
<th>Basic caching</th>
<th>Advanced caching</th>
</tr>
</thead>
<tbody>
<tr>
<td>Memory usage per item:</td>
<td>Medium</td>
<td>High</td>
</tr>
<tr>
<td>Performance improvement:</td>
<td>Very High</td>
<td>High</td>
</tr>
<tr>
<td>Custom caching available:</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Connect tag processing:</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Web Content Viewer Portlet:</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Caching Personalization components:

Web content caching can sometimes be used with Personalization components but will depend on the conditions set in the personalization rule, or the resources used to determine the rule results. Cache testing will be required to determine if the content returned by your personalization component can be cached using web content caching.

Caching versus pre-rendering

Content displayed in rendering portlets and through IBM Web Content Manager can be cached. An alternative to caching is the use of the pre-rendering feature. View the differences between each strategy.

A pre-rendered site can be viewed in two ways:

Using a web server

Viewing a pre-rendered site through a web server is similar to using basic caching because the displayed content is static and custom caching cannot be used.
Using Web Content Manager

Viewing a pre-rendered site through Web Content Manager is similar to using advanced caching because content can be dynamic and custom caching can be used.

Basic caching versus a pre-rendered site delivered with a web server

At first glance, the pre-rendering feature and basic caching do the same thing. There are however, some major differences that will determine which feature is the best for you.

The main difference between the two features is that the pre-rendering feature takes a snapshot of the entire site each time it is run. Basic caching only caches on a page-by-page basis. If performance is your main issue, then pre-rendering might be the answer. If not, then basic caching might be a better option.

Table 17. Basic caching versus a pre-rendered site delivered with a web server

<table>
<thead>
<tr>
<th>Function</th>
<th>Basic caching</th>
<th>Pre-rendered site delivered with a web server</th>
</tr>
</thead>
<tbody>
<tr>
<td>Performance:</td>
<td>Very fast</td>
<td>Extremely fast</td>
</tr>
<tr>
<td>Connect tag processing:</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Custom caching:</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Memory requirements:</td>
<td>Low to Medium</td>
<td>Memory requirements depends on the web server being used.</td>
</tr>
<tr>
<td>Disk requirements:</td>
<td>Low to Medium</td>
<td>Potentially very high as the entire site must be able to fit on disk.</td>
</tr>
<tr>
<td>Unexpected broken links:</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>As some pages may be cached at different times, there is a small chance that not all the links on a cached page will be currently valid.</td>
<td>The site is pre-rendered in a single batch, greatly reducing the chances of inconsistencies in the site.</td>
</tr>
</tbody>
</table>

Advanced caching versus a pre-rendered site delivered using Web Content Manager

These options are very similar. You may have to test both strategies before deciding which is best for your site.

Table 18. Advanced caching versus a pre-rendered site delivered using Web Content Manager

<table>
<thead>
<tr>
<th>Function</th>
<th>Advanced caching</th>
<th>Pre-rendered site delivered through Web Content Manager</th>
</tr>
</thead>
<tbody>
<tr>
<td>Performance:</td>
<td>Fast when cached, but slower if the requested page has expired from the cache. (As tag processing has a cost, this depends on how many connect tags a page contains.)</td>
<td>Fast, but as tag processing has a cost, this depends on how many connect tags a page contains.</td>
</tr>
</tbody>
</table>
Table 18. Advanced caching versus a pre-rendered site delivered using Web Content Manager (continued)

<table>
<thead>
<tr>
<th>Function</th>
<th>Advanced caching</th>
<th>Pre-rendered site delivered through Web Content Manager</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connect tag processing:</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Custom caching:</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Memory requirements:</td>
<td>Medium to high.</td>
<td>Medium to high.</td>
</tr>
<tr>
<td>Disk requirements:</td>
<td>Medium to high.</td>
<td>Medium to high.</td>
</tr>
<tr>
<td>Unexpected broken links:</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>
|                           | As some pages may be cached at different times, there is a small chance that not all the links on a cached page will be currently valid. | The site is pre-rendered in a single batch, greatly reducing the chances of inconsistencies in the site.

**Expiring strategies**

Like caching strategies, a server’s default expiring strategies can be set in the WCM WCMConfigService service using the WebSphere Integrated Solutions Console. Custom expiring parameters can also be set in connect tags and URL requests to override a server’s default expiring strategies.

**Note:** If basic caching is used as your default web content cache, custom expiring cannot be used.

In most cases the expiry schedule is based around how often the source content is updated. So, if the source content is updated hourly, then each cache would be expired hourly. If the source content is updated daily, then each cache would be expired daily.

Beyond these examples, a different expiry schedule would be used. If your web pages were only updated weekly, or monthly, you would still schedule your caches to expire daily. Otherwise, when your source content was updated, it could take up to a week for it to appear on your site.

**Caching expiries versus workflow expiries**

The expires parameter in a workflow is not related to the Expires parameter in IBM Web Content Manager caching. A page that is set to expire at midnight as part of a workflow will only do so if it has not already been saved in a cache. The page will remain in the cache until expired by the Web Content Manager application regardless of the Expires setting in a workflow.
You use the "expires" parameter in IBM Web Content Manager tags and URLs to specify how long to maintain data in the cache before it is expired. Once data expires from a cache, the next request for the data will be retrieved from the original server. The expires parameter is not mandatory.

Related tasks:
Setting service configuration properties
IBM WebSphere Portal comprises a framework of services to accommodate the different scenarios that portals need to address. Services are available for both WebSphere Portal and IBM Web Content Manager. You can configure some of these services.

Web content cache configuration
You can tailor the caching behavior of your web content environment by changing configuration settings such as the default cache type and expire settings.

You define and manage web content cache options in the WCM WCMConfigService service using the WebSphere Integrated Solutions Console.

Setting the default web content cache type
The default web content caching environment for your web content server is specified by the following properties:
- connect.businesslogic.defaultcache
- connect.moduleconfig.ajpe.contentcache.defaultcontentcache

Table 19. Caching parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>defaultcache value</th>
<th>defaultcontentcache value</th>
</tr>
</thead>
<tbody>
<tr>
<td>No caching:</td>
<td>false</td>
<td>None</td>
</tr>
<tr>
<td>Basic cache:</td>
<td>true</td>
<td>Not specified</td>
</tr>
<tr>
<td>Site caching:</td>
<td>false</td>
<td>Site</td>
</tr>
<tr>
<td>Session caching:</td>
<td>false</td>
<td>Session</td>
</tr>
<tr>
<td>User caching:</td>
<td>false</td>
<td>User</td>
</tr>
<tr>
<td>Secured caching:</td>
<td>false</td>
<td>Secured</td>
</tr>
<tr>
<td>Personalized caching:</td>
<td>false</td>
<td>Personalized</td>
</tr>
</tbody>
</table>

Additional default web content cache parameters
Web content cache configuration settings are specified by the following properties in the WCM WCMConfigService service.

Table 20. Cache properties per cache type

<table>
<thead>
<tr>
<th>Cache Type</th>
<th>Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic cache:</td>
<td>connect.businesslogic.defaultcacheexpires</td>
</tr>
<tr>
<td></td>
<td>connect.businesslogic.defaultcache</td>
</tr>
<tr>
<td>Advanced cache:</td>
<td>connect.moduleconfig.ajpe.contentcache.defaultcontentcache</td>
</tr>
<tr>
<td></td>
<td>connect.moduleconfig.ajpe.contentcache.contentcacheexpires</td>
</tr>
</tbody>
</table>
Table 20. Cache properties per cache type (continued)

<table>
<thead>
<tr>
<th>Cache Type</th>
<th>Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>Advanced cache: Session</td>
<td>connect.sessioncacheconfig.memcachesize</td>
</tr>
<tr>
<td>cache only</td>
<td></td>
</tr>
</tbody>
</table>

Table 21. Cache properties details

<table>
<thead>
<tr>
<th>Cache Property</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>contentcacheexpires</td>
<td>This sets the default expiry for all advanced caches. It can be either a relative period or an absolute date and time.</td>
</tr>
<tr>
<td>defaultcache</td>
<td>If true, basic caching is enabled. If false or missing, advanced caching is enabled.</td>
</tr>
<tr>
<td>defaultcacheexpires</td>
<td>This sets the default expiry for the basic cache. It can be either a relative period or an absolute date and time.</td>
</tr>
<tr>
<td>defaultcontentcache</td>
<td>If the advanced cache is enabled, the default advanced cache is set here.</td>
</tr>
<tr>
<td>resourceserver.browserCacheMaxAge</td>
<td>The maximum time an item will be stored in a web browser cache.</td>
</tr>
<tr>
<td>resourceserver.maxCacheObjectSize</td>
<td>The maximum size of objects that can be cached in kilobytes. By default this is set to 300.</td>
</tr>
</tbody>
</table>

Cache expire time formats

When setting the cache expire settings listed in Table 3, you can specify either a relative time, or absolute time:

- REL [integer-value][units]
- ABS [date-format-string]

[units] =
  - d|D for days
  - m|M for months
  - s|S for seconds
  - h|H for hours

[date-format-string] =
  - Mon, 06 Nov 2000 09:00:00 GMT
  - Monday, 06-Nov-00 09:00:00 GMT
  - Mon Nov 6 09:00:00 2000
  - 6 Nov 2000 9:00 AM

Note: The last two formats assume GMT.

Examples:
- contentcacheexpires="REL 300S"
- contentcacheexpires="ABS Mon, 06 Nov 2000 09:00:00 GMT"
IBM WebSphere Portal comprises a framework of services to accommodate the different scenarios that portals need to address. Services are available for both WebSphere Portal and IBM Web Content Manager. You can configure some of these services.

**Data cache configuration**

Data caching is used to cache data retrieved by the IBM Web Content Manager application from external sources using connect tags or by requests made through URLs.

You define and manage data cache options in the WCM WCMConfigService service using the WebSphere Integrated Solutions Console.

Specify the following properties for data cache options:

- `connect.connector.httpconnector.defaultcache`
  - Used when no cache is specified in a request for data. Possible values are `true` or `false`. If true, the data will be stored in the site cache.

- `connect.connector.httpconnector.defaultcachefiles`
  - The expiry date/time for items added to a cache (site or session) if the expiry date/time is not specified in the request.

- `connect.connector.sqlconnector.defaultcache`
  - Determines whether to cache data by default or not. Possible values are `true` or `false`.

Related tasks:

**Setting service configuration properties**

IBM WebSphere Portal comprises a framework of services to accommodate the different scenarios that portals need to address. Services are available for both WebSphere Portal and IBM Web Content Manager. You can configure some of these services.

**Pre-rendering options**

You can enable pre-rendering so that content can be viewed either through a IBM Web Content Manager application or as a standalone site that is accessed through a web server.

You define and manage pre-rendering options in the WCM WCMConfigService service using the WebSphere Integrated Solutions Console.

**Start pre-rendering automatically**

Although you can manually pre-render a website through the URL interface, you can also configure pre-rendering to run automatically when the server starts.

1. Click **Resources > Resource Environment > Resource Environment Providers**.
2. Click **WCM WCMConfigService**.
3. Under **Additional Properties**, click **Custom Properties**.
4. Edit the `connect.businesslogic.module` property, and append `cacher` to the value. For example:

```text
web, mail, default, ajpe, federatedProxy, ajpecomms, memberfixer, workflowenablement, itemdispatcher, plutouploadfile, plutodownloadfile, synd, subs, syndication, refreshallitems, unlocklibrary, custom, data, clearversions, clearhistory, reseteventlog, cacher
```
5. Save your changes and restart the server.

**Enable pre-rendering for sites viewed using Web Content Manager**

This option is used when you are accessing the pre-rendered site through Web Content Manager. This will increase performance as static content is accessed from the pre-rendered site, but dynamic content will still be rendered through Web Content Manager.

To enable users to access the pre-rendered site through a Web Content Manager application, specify the `connect.businesslogic.module.default.class` property in the WCM WCMConfigService service using the WebSphere Integrated Solutions Console.

- Property name: `connect.businesslogic.module.default.class`
- Value: `com.aptrix.cacher.CacherModule`

**Note:** You cannot use the local rendering portlet (Web Content Viewer) when pre-rendering is set as the default module.

**Enable pre-rendering for standalone sites**

This option is used when you are using Web Content Manager to generate a pre-rendered site, but are not using Web Content Manager to view the pre-rendered site. You will need to use a web server to view the pre-rendered site.

Specify the `connect.businesslogic.module.cacher.class` property in the WCM WCMConfigService service using the WebSphere Integrated Solutions Console.

- Property name: `connect.businesslogic.module.cacher.class`
- Value: `com.aptrix.cacher.CacherModule`

Specify the following properties to configure caching. Default values are listed, although you can tailor these values as needed. Unless you explicitly set a value for a property, the default value is used.

**connect.moduleconfig.cacher.destdir**

Value: `${USER_INSTALL_ROOT}/PortalServer/wcm/ilwwcm/cacher`

Base directory under which each site cache will be created. There will be one subdirectory created for each site.

**Important:** If the prerenderer is run with the `connect.moduleconfig.cacher.overwritecache` property set to true, any files in the `connect.moduleconfig.cacher.destdir` path that were not written in the last run of the prerenderer will be deleted. For this reason, ensure that the `connect.moduleconfig.cacher.destdir` path is only used for storing rendered content and that it does not contain any other data that cannot be recreated.

**connect.moduleconfig.cacher.tempdir**

Value: `${USER_INSTALL_ROOT}/PortalServer/wcm/ilwwcm/cacher/temp`

The temporary directory that is required to build the site cache prior to moving the data over to the base directory specified by the `connect.moduleconfig.cacher.destdir` property.
connect.moduleconfig.cacher.delay
Value: 1
This is used to set the time, in seconds, between requesting a page while caching.

connect.moduleconfig.cacher.busydelay
Value: 5
This is used to set the time, in seconds, of the busy delay setting. This is used if executing within the busy start to busy end period. Otherwise the delay setting is used.

connect.moduleconfig.cacher.busystart/connect.moduleconfig.cacher.busyend
Value: 9:00 am/5:00 pm
These settings determine the times between which the busy delay setting will be used. Enter an absolute time as shown.

connect.moduleconfig.cacher.overwritecache
true  The prerenderer will overwrite files in the destdir cache directory (then delete unneeded files). This results in a progressive change in site content as seen by the user. This is the default value.
false  The first time a site is pre-rendered, the cached site files will be added to the destination directory. As changes are made to the site through the authoring portlet, the new version of the site will gradually be cached in the temporary directory and the old site will remain in the destination directory. After the cacher has finished caching the site completely, the contents of the temporary directory are moved to the destination directory which will then contain both old and new versions of the cached site.

Note: A value of false should not be used if a web server is used to display the pre-rendered data because some web servers lock the data directories.

connect.moduleconfig.cacher.rendereruser
Value: Anonymous.
This determines the user to be used to render the Web Content Manager content. Either type Anonymous or Administrator or a specific user or group name.
The site is pre-rendered based on this user's security rights. If the user specified here does not have access to a particular component it will not be pre-rendered.

connect.moduleconfig.cacher.task.cacherurl
Value: http://${WCM_HOST}:${WCM_PORT}/${WCM_CONTEXT_ROOT}/connect/
The full URL to be used as the replacement for the connect servlet in pre-rendered pages. The URL should end with the string specified in connect.moduleconfig.cacher.task.servletpath property if it is not blank.
The context of cacherurl is used when generating a URL with pre-rendering. This property is not used when a page belongs to a site that has not already been pre-rendered at a site level by the scheduled task or through a SRV=cacheSite request.

connect.moduleconfig.cacher.task.servletpath
Value: /connect
The path of the substituted connect servlet defined in the `connect.moduleconfig.cacher.task.cacherurl` property. This property can remain blank if the cacherurl context should be used unchanged.

**connect.moduleconfig.cacher.defaultcontentname**
- Value: `index.html`

This sets the name of the default or home file used when accessing the pre-rendered site. This normally would be `index.html`.

**connect.moduleconfig.cacher.task.siteareas**
- Value: `LibraryA/SiteAreaA,LibraryB/SiteAreaB,SiteAreaC`

The site areas within a Web Content Manager environment to cache are entered here, separated by commas. This property provides the option of specifying the library in addition to the site area. If the library is specified, the pre-renderer looks for the site area in that library. If no library is specified, the default library is used, as specified in the `defaultLibrary` property.

**Note:** If any of your site area names contain commas, you must create separate parameters for each site area using this format:

`connect.moduleconfig.cacher.task.siteareas.N`

N represents a different integer for each parameter. For example, if you want to pre-render a site area named "SiteArea,Red" and a site named "Site,Yellow", you would need to create the following parameters:

`connect.moduleconfig.cacher.task.siteareas.1=MyLib/SiteArea,Red`
`connect.moduleconfig.cacher.task.siteareas.2=Site,Yellow`

**connect.moduleconfig.cacher.task.interval.recurrence**
**connect.moduleconfig.cacher.task.interval.startdelay**

The CacherModule can be set to run after a recurring number of minutes.

**recurrence:**
- Value: `10`

The recurring period in minutes for a recurring task.

**startdelay:**
- Value: `1`

The delay in minutes prior to starting the first recurring task.

**Note:** If you do not configure pre-rendering to start automatically when the server starts, pre-rendering at intervals does not work until you manually trigger the cacher module.

**connect.moduleconfig.cacher.task.scheduled.times**
- Value: `3:00 am`

Alternately, the CacherModule can be set to run at certain times. Enter a series of absolute times, separated by commas.

**Important:** When specifying time values, be sure you conform to the format `H:MM am|pm`, including the use of the colon (:) character and the space. Incorrectly specified values prevent pre-rendering from functioning properly.
Note: If you do not configure pre-rendering to start automatically when the server starts, pre-rendering at scheduled times does not work until you manually trigger the cacher module.

Pre-rendering resources

`connect.moduleconfig.cacher.useTieredResourceFolders`

Value: false

All resources, such as images and file resources, are stored under the following folder:

`CACHER_DIR\LIBRARY\SITEAREA\resources`

By default, each individual resource is saved under its own folder. For example, a resource with the ID of "7961d78049717f29bc57fee5670e9d7b" will be stored under this folder:

`CACHER_DIR\LIBRARY\SITEAREA\resources\7961d78049717f29bc57fee5670e9d7b`

You can change this behavior so that resources are stored under a tiered set of sub-folders based on the first two characters of the resource ID by changing the value of `connect.moduleconfig.cacher.useTieredResourceFolders` to true. For example, a resource with the ID of "7961d78049717f29bc57fee5670e9d7b" will be stored under this folder:

`CACHER_DIR\LIBRARY\SITEAREA\resources\7\9`

All other resources that whose IDs begin with "79" will also be stored under this folder. This is done to reduce the number of sub-folders under the "resources" folders.

Related tasks:

- Setting service configuration properties

IBM WebSphere Portal comprises a framework of services to accommodate the different scenarios that portals need to address. Services are available for both WebSphere Portal and IBM Web Content Manager. You can configure some of these services.

Disabling the site toolbar on a delivery server

The site toolbar provides access to editing features for managed pages, including adding and editing pages and web content. Although essential for an authoring server, it is recommended that you disable the site toolbar on a delivery server. You can disable the toolbar for an entire portal or for specific virtual portals.

The site toolbar function is not typically needed on a delivery server, and disabling the site toolbar can improve performance on the delivery server.

1. Log in to the WebSphere Integrated Solutions Console as an administrator.
2. Click Resources > Resource Environment > Resource Environment Providers.
3. Click WP VirtualPortalConfigService.
4. Update the appropriate configuration properties, depending on whether you want to affect the entire portal or a specific virtual portal.
   - To affect the entire portal, complete the following steps:
     a. Click Custom properties.
     b. Edit the `global.toolbar.enabled` property, and set the value to false.
        This setting disables the site toolbar for all virtual portals.
   - To affect a specific virtual portal, complete the following steps:
a. Click Custom properties.

b. To disable the site toolbar for the default virtual portal, edit the default.toolbar.enabled property, and set the value to false.

c. For each virtual portal other than the default where you want to disable the site toolbar, specify the following properties.

   context.virtual_portal_context.property.toolbar.enabled
   Set the value to false. Replace virtual_portal_context with the context of the target virtual portal (for example, context.vpl.property.toolbar.

   hostname.virtual_portal_hostname.property.toolbar.enabled
   Set the value to false. Replace virtual_portal_hostname with the host name of the target virtual portal (for example, hostname.vp.example.com.property.toolbar.enabled.

   If defined, the global.toolbar.enabled property acts as a fallback setting for virtual portals that have no values defined.

   For more information about prefixes, placeholders, and the order in which properties are evaluated, see Virtual Portal Configuration Service

Related reference:

Virtual Portal Configuration Service

The Virtual Portal configuration service (WP VirtualPortalConfigService) enables you to specify properties for the default virtual portal and for specific virtual portals.

Reserved authoring portlet

When working with the web content viewer or Web content pages, some scenarios involve web content authoring tasks accomplished with authoring tools components. Such authoring tasks are performed through a special instance of the authoring portlet that is reserved specifically for these tasks and is installed on page that is hidden from the page navigation available to typical users.

The following tasks use the reserved authoring portlet:

- Selecting a web content folder when creating or editing the properties of a web content page.
- Configuring the web content viewer, such as selecting the content item to display.
- Performing inline editing using authoring tools components rendered in the web content viewer.

Typically authoring tasks are performed in a separate window that opens from the current page, but you can configure the behavior of authoring tools components to redirect users to the hidden page containing the reserved authoring portlet.

Ensuring the availability of the reserved authoring portlet

If either the authoring portlet instance or the hidden portal page are not available or if the user lacks the permission to access either of them, the authoring tasks requiring the reserved authoring portlet will fail, causing web content pages and the web content viewer to be unusable. For this reason, you must be careful when administering the reserved authoring portlet and the hidden portal page.

The following conditions are essential for the proper function of the reserved authoring portlet:


• Users performing any of the previous authoring tasks must have the User role on the hidden portal page.
• Users performing any of the previous authoring tasks must have the User role on the reserved authoring portlet.
• The reserved authoring portlet must be the only portlet located on the hidden portal page.
• The unique name of the hidden portal page must be `com.ibm.wps.hiddenpage.wcm.Authoring_Portlet`.
• The unique name of the portlet window of the authoring portlet instance on the hidden portal page must be `com.ibm.wps.hiddenpage.wcm.control.Authoring_Portlet`.

Availability problems related to the reserved authoring portlet or the hidden portal page are usually identified by the following symptoms:

• The SystemOut.log file for the portal server contains error messages referencing the authoring portlet or hidden page. For example:
  
  `EJPDB0124E: The specified string [com.ibm.wps.hiddenpage.wcm.Authoring_Portlet] can neither be deserialized as an object ID nor resolved as a unique name. EJPDB0124E: The specified string [com.ibm.wps.hiddenpage.wcm.control.Authoring_Portlet] can neither be deserialized as an object ID nor resolved as a unique name.`

• When a separate window is launched from the current page to perform the authoring task, the new window displays the following message:
  
  `Error 400: EJPPH0006E: The resolution of a URI failed. Refer to the stack trace for more detailed information.`

• When a separate window is launched from the current page to perform the authoring task, the new window is empty.

• When the user is redirected to another portal page to perform the authoring task, the user is redirected to the default portal page instead of the page containing the reserved portlet.

• When the user is redirected to another portal page to perform the authoring task, the user is redirected to an empty page.

If any of these problems occur, verify that the conditions for proper operation of the reserved authoring portlet and hidden portal page are fully implemented.

**Note:** If the reserved authoring portlet or the hidden portlet page are removed inadvertently, you can deploy them again using the action-install-wcm-hidden-authoring configuration task.

**Configuring the reserved authoring portlet**

The reserved authoring portlet is essential to the proper operation of web content pages and the web content viewer, so it is important that the configuration of the reserved authoring portlet reflect similar settings for performing authoring tasks as the configuration of other instances of the IBM Web Content Manager authoring portlet.

1. Log in to the portal as an administrator.
2. Click **Administration** in the tool bar.
3. Under **Portal User Interface** in the navigation tree, click **Manage Pages**.
5. Click the Edit Page Layout icon (small pencil) for the page.
6. Select Edit shared settings from the portlet menu, and specify any settings for the reserved authoring portlet. The available settings and the process for updating them is the same for the reserved authoring portlet as it is for any other instance of the authoring portlet.

   **Note:** Changes made to the reserved authoring portlet with the Edit shared settings mode affect only the reserved authoring portlet and no other instances of the authoring portlet. To ensure a consistent authoring experience, you can make the same changes to other authoring portlet instances using the Edit shared settings mode for each instance. Alternatively, you could make the same changes to every instance of the authoring portlet by using the Configure mode from a single instance. Changes you make in the Configure mode also affect the reserved authoring portlet.

7. Save your changes.

**Related concepts:**

"Controlling the behavior of authoring tools components" on page 271

**Related reference:**

Defining authoring tools

You can format an authoring tool element’s look and feel in different ways, including displaying authoring tools as text-based links, or image based links.

### Additional configuration options

These configuration options are available to address installation requirements for additional deployment scenarios.

### Controlling access to hosts specified in a URL

By default, you can specify any host name in a URL used to retrieve content. However, you can restrict access to a specified list of host names by modifying the configuration of the WCM WCMConfigService service.

1. Log in to the WebSphere Integrated Solutions Console.
2. Click Resources > Resource Environment > Resource Environment Providers > WCM WCMConfigService > Custom properties.

   **Cluster note:** If you are using this web content server as part of a cluster, ensure that you use the WebSphere Integrated Solutions Console for the deployment manager when manipulating configuration properties.

3. Update the configuration to block access from unknown hosts. Specify the following property:

   - Property name: connect.connector.httpconnector.denyunknownhosts
   - Value: true

4. For each host name for which you want to grant access, add a new property. Use the following format for new properties:
5. Optional: Specify a default cache expiration value for the host name you added by adding a new property. Use the following format for new properties:

- Property name:
  connect.connector.httpconnector.hosts.<host_name>.defaultcacheexpires,
  where <host_name> is the fully qualified host name of the server for which you want to permit access. For example:
  connect.connector.httpconnector.hosts.www.example.com.defaultcacheexpires
- Value: expiration_time. For example: REL 9000s

6. Optional: Specify a default cache setting for the host name you added by adding a new property. Use the following format for new properties:

- Property name:
  connect.connector.httpconnector.hosts.<host_name>.defaultcache,
  where <host_name> is the fully qualified host name of the server for which you want to permit access. For example:
  connect.connector.httpconnector.hosts.www.example.com.defaultcache
- Value: true or false

Related concepts:

"Cache expire parameters” on page 238

You use the "expires" parameter in IBM Web Content Manager tags and URLs to specify how long to maintain data in the cache before it is expired. Once data expires from a cache, the next request for the data will be retrieved from the original server. The expires parameter is not mandatory.

Related tasks:

Setting service configuration properties

IBM WebSphere Portal comprises a framework of services to accommodate the different scenarios that portals need to address. Services are available for both WebSphere Portal and IBM Web Content Manager. You can configure some of these services.

Web content substitution variables

IBM Web Content Manager uses several substitution variables defined in the configuration for IBM WebSphere Application Server.

If you need to modify these variables, use the WebSphere Integrated Solutions Console for the application server. If you are working with a managed cell or cluster, use the WebSphere Integrated Solutions Console for the deployment manager when making changes.

Table 22. Web content substitution variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WCM_CONTEXT_ROOT</td>
<td>The context root for the enterprise application for Web Content Manager.</td>
</tr>
<tr>
<td></td>
<td>Example: wps/wcm</td>
</tr>
<tr>
<td>WCM_HOST</td>
<td>The fully qualified host name of the machine running the portal.</td>
</tr>
<tr>
<td></td>
<td>Example: <a href="http://www.example.com">www.example.com</a></td>
</tr>
<tr>
<td>Variable</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td>WCM_ILWWCM_HOME</td>
<td>This is the directory where the Web Content Manager application is installed. Example: <code>PortalServer_root/wcm</code></td>
</tr>
<tr>
<td>WCM_PORT</td>
<td>The port number used to access the portal. Example: 10038</td>
</tr>
<tr>
<td>WCM_SCHEMA</td>
<td>The database schema name of the JCR domain configured for use with IBM WebSphere Portal. Example: jcr</td>
</tr>
<tr>
<td>WCM_SEARCHSEED_CONTEXT_ROOT</td>
<td>The context root for the Search Seed portlet. Example: <code>wps/wcmsearchseed</code></td>
</tr>
<tr>
<td>WCM_WEB_APP_HOME</td>
<td>The directory path where the ilwwcm.war file is located. Example: <code>was_profile_root/wp_profile/installedApps/node_name/wcm.ear/ilwwcm.war</code></td>
</tr>
<tr>
<td>WCM_WPS_CONTEXT_ROOT</td>
<td>The context root or base URI for the portal. All URLs beginning with this path will be reserved for the portal. Example: wps <code>http://hostname.example.com:10038/wps/portal</code></td>
</tr>
<tr>
<td>WCM_WPS_DEFAULT_HOME</td>
<td>The default portal page. This is the page for users who are not logged in. Example: portal <code>http://hostname.example.com:10038/wps/portal</code></td>
</tr>
<tr>
<td>WCM_WPS_PERSONALIZED_HOME</td>
<td>The portal page for users who have already logged in to the portal. This page cannot be accessed by anonymous users. Example: myportal <code>http://hostname.example.com:10038/wps/myportal</code></td>
</tr>
</tbody>
</table>

### Enabling connect tags

Enable connect tags to reference web content components and apply customized caching to the components.

1. Log in to the WebSphere Integrated Solutions Console.
2. Click Resources > Resource Environment > Resource Environment Providers > WCM WCMConfigService > Custom properties.
**Cluster note:** If you are using this web content server as part of a cluster, ensure that you use the WebSphere Integrated Solutions Console for the deployment manager when manipulating configuration properties.

3. Specify `connect.businesslogic` properties to process connect tags from any host or from specific hosts.

**Process connect tags from any host**
Add the following property:
- Property name: `connect.businesslogic.processunknownhosts`
- Value: `true`

**Process connect tags from specific hosts**
Add the following property:
- Property name: `connect.businesslogic.processunknownhosts`
- Value: `false`

For each host for which you want to enable processing, add a new property:
- Property name: `connect.businesslogic.hosts.hostname`
- Value: `true`

4. Restart the server or cluster.

**Related tasks:**
- [Setting service configuration properties](#)

IBM WebSphere Portal comprises a framework of services to accommodate the different scenarios that portals need to address. Services are available for both WebSphere Portal and IBM Web Content Manager. You can configure some of these services.

**Remove authoring configuration task**
The remove authoring configuration task will uninstall the authoring portlet and related portal pages.

**Running the configuration task:**

To remove the Authoring portlet:
1. Stop the server.
2. Open a command prompt.
3. Open a telnet session to the portal server.
4. Run the `remove-wcm-authoring` task from the `wp_profile_root/ConfigEngine` directory.

**Windows**
```
ConfigEngine.bat remove-wcm-authoring -DWasPassword=encryption
```

**UNIX**
```
./ConfigEngine.sh remove-wcm-authoring -DWasPassword=encryption
```

**IBM i**
```
ConfigEngine.sh remove-wcm-authoring -DWasPassword=encryption
```

**z/OS**
```
./ConfigEngine.sh remove-wcm-authoring -DWasPassword=encryption
```

**Enabling email**
To use the email workflow action you must configure Web Content Manager to use your SMTP server.
1. Log in to the WebSphere Integrated Solutions Console.
2. Click **Resources > Resource Environment > Resource Environment Providers > WCM WCMConfigService > Custom properties.**

   **Cluster note:** If you are using this web content server as part of a cluster, ensure that you use the WebSphere Integrated Solutions Console for the deployment manager when manipulating configuration properties.

3. Specify `connect.connector.mailconnector` properties to use your SMTP server. Add the following properties:

   **Default SMTP server**
   - Property name: `connect.connector.mailconnector.defaultsmtpserver`
   - Value: `mail.yourmailserver.com`

   **Default email address for "from" field**
   - Property name: `connect.connector.mailconnector.defaultfromaddress`
   - Value: `admin@yourmailserver.com`

   **Default email address for "reply-to" field**
   - Property name: `connect.connector.mailconnector.defaultreplytoaddress`
   - Value: `admin@yourmailserver.com`

4. If you use a secured SMTP server, you will also need to specify a user name and password to access the SMTP server: Add the following properties:

   **Default user name**
   - Property name: `connect.connector.mailconnector.defaultusername`
   - Value: `username`

   **Default password**
   - Property name: `connect.connector.mailconnector.defaultpassword`
   - Value: `password`

5. Save your changes.
6. Restart the portal for the new settings to take effect.

**Related tasks:**
- **Setting service configuration properties**

**Configuring managed pages**

When you perform a new installation of IBM WebSphere Portal 8.0, managed pages are enabled by default. However, you can also manually disable and enable the feature as needed.

**Migration note:** If you migrate from a previous version, managed pages are disabled by default, but you can enable the feature after migration.
Enabling managed pages

By default, support for managed pages is enabled for the default virtual portal. However, you can also manually enable this support if managed pages are disabled.

Important: Do not attempt to enable managed pages on a server where managed pages are already enabled. If you previously disabled managed pages and want to re-enable the feature, you must ensure that the Portal Site library is empty first. If you fail to remove page artifacts from the previous configuration, the resulting portal might not work properly.

When support for managed pages is enabled for a virtual portal, all pages in the virtual portal are copied into the Portal Site library in IBM Web Content Manager. However, the following pages are not treated as managed pages and are not copied:
- Administration pages, as identified by the label ibm.portal.Administration and its children
- Private pages

Each virtual portal has its own Portal Site library.

Note: To take advantage of the features available to managed pages in the user interface, your pages must use the Portal 8.0 theme.

1. Start the portal server.
2. To enable support for managed pages, run the enable-managed-pages task from the \wp_profile_root\ConfigEngine directory.
   - Windows
     ConfigEngine.bat enable-managed-pages -DPortalAdminPwd=password -DWasPassword=password
   - AIX Linux Solaris
     ./ConfigEngine.sh enable-managed-pages -DPortalAdminPwd=password -DWasPassword=password
   - IBM i
     ConfigEngine.sh enable-managed-pages -DPortalAdminPwd=password -DWasPassword=password
   - z/OS
     ./ConfigEngine.sh enable-managed-pages -DPortalAdminPwd=password -DWasPassword=password

After running the enable-managed-pages task for the first time, the property \managed\_pages is created in the WP WPConfigService configuration service. The value of the property is set to true.

3. Restart the portal server.
4. To populate web content libraries with information about virtual portals in the system, run the create-virtual-portal-site-nodes task from the \wp_profile_root\ConfigEngine directory. For each virtual portal, this task creates a library and a site area called lost-found for resources that cannot be properly located. If the library or site area exist, the task exits. By default, the task runs on all virtual portals in the system.
   - Windows
     ConfigEngine.bat create-virtual-portal-site-nodes -DPortalAdminPwd=password -DWasPassword=password
5. To populate web content libraries with information about the portal pages in the system, run the `create-page-nodes` task from the `wp_profile_root/ConfigEngine` directory.

This task can also be used when portal pages and managed pages artifacts in Web Content Manager are not synchronized. In this case, the task attempts to resynchronize the portal artifacts and web content artifacts, giving precedence to the portal artifacts.

**Performance note:** Depending on the amount of information in the system, the `create-page-nodes` task can take a long time to run. Because of the database load of the task, it is not recommended that you run the task frequently. The initial run of the task requires the most time, while subsequent runs typically require less time.

Windows

```
ConfigEngine.bat create-page-nodes -DPortalAdminPwd=password -DWasPassword=password
```

AIX Linux Solaris

```
./ConfigEngine.sh create-page-nodes -DPortalAdminPwd=password -DWasPassword=password
```

IBM i

```
ConfigEngine.sh create-page-nodes -DPortalAdminPwd=password -DWasPassword=password
```

z/OS

```
./ConfigEngine.sh create-page-nodes -DPortalAdminPwd=password -DWasPassword=password
```

By default, this task is performed on all pages in all virtual portals. To limit this task to a specific virtual portal, identify the virtual portal by adding one of the following parameters to the command line. Each parameter requires the prefix `-D` on the command line.

**VirtualPortalHost**

Specify the host name of the virtual portal. For example, `vp.example.com`.

**Important:** If the host name of the virtual portal is the same as the host name of the default virtual portal, you must also specify the `VirtualPortalContext` property. You can specify the `VirtualPortalHost` property by itself only if the host name is unique.

**VirtualPortalContext**

Specify the virtual portal context that identifies the virtual portal. For example, `vp1`.

You can customize the task with the following optional parameters on the command line. Each parameter requires the prefix `-D` on the command line.

**RunParallel**

Indicate whether you want the task to run with multiple threads. A value of `false` indicates a single thread and is the default setting.
A value of true indicates multiple threads, as specified by the work manager \texttt{wpsJcrSyncWorkManager} in the WebSphere Integrated Solutions Console. Each thread requires a database connection. For optimal performance, ensure that your database connection pool supports at least as many connections as there are threads in the pool.

**Excluded**

Specify a list of unique names of page nodes to exclude from the creation process. Excluding a page also excludes its child pages. By default, the portal administration pages (\texttt{ibm.portal.Administration}) are excluded.

6. Optional: If you used web content pages before enabling managed pages, you can transfer the content that is associated with those pages to the Portal Site library. For details on performing this transfer, see \textit{Transferring content associations to the Portal Site library}.

**Related tasks:**

“Transferring content associations to the Portal Site library” on page 104

When you enable manage pages, any web content pages that you have are converted to managed pages and added to the Portal Site library. However, the content that is associated with the web content pages remains in the original libraries. You can transfer this associated content to the Portal Site library with the \textit{internalize-content-mappings} task.

**Disabling managed pages**

Disable support for managed pages by running the \texttt{disable-managed-pages} configuration task.

Disabling managed pages has the following effects:

- By default, each virtual portal has its own specific workspace where content is stored. When you disable managed pages, only a single workspace for the default virtual portal is available. The workspaces of other virtual portals are still there, but you can no longer access them. Any system associations between pages in those virtual portals and their respective Portal Site libraries no longer work.

**Important:** To preserve content in the other virtual portals, you must import or syndicate the libraries into the default virtual portal before disabling managed pages.

- You can still access the Portal Site library for the default virtual portal, but the library is no longer automatically synchronized with the page structure.

- Pages are no longer managed in IBM Web Content Manager, with the following implications:
  - No page drafts can be created.
  - No new versions of pages can be created.
  - Pages are no longer syndicated.
  - Access control changes that you perform in the portal interface are no longer applied to the portal page site area.
  - If you delete a page from the portal interface, the corresponding portal page site area is not deleted.

- If you create a page with either the \texttt{Basic} or \texttt{Articles} page template, the page has no web content association. This missing association can cause errors if you attempt to add content from the \texttt{Web Content} category of the \texttt{Content} tab in the
To use the sample web content items when managed pages are disabled, create a web content association on the page before attempting to add content.

1. Run the disable-managed-pages task from the ConfigEngine directory.

Windows

```
ConfigEngine.bat disable-managed-pages -DPortalAdminPwd=password
-DSwasPassword=password
```

AIX Linux Solaris

```
./ConfigEngine.sh disable-managed-pages -DPortalAdminPwd=password
-DSwasPassword=password
```

IBM i

```
ConfigEngine.sh disable-managed-pages -DPortalAdminPwd=password
-DSwasPassword=password
```

z/OS

```
./ConfigEngine.sh disable-managed-pages -DPortalAdminPwd=password
-DSwasPassword=password
```

After running the disable-managed-pages task for the first time, the property "managed.pages" is created in the WP WPConfigService configuration service. The value of the property is set to false.

2. Restart the portal server.

### Transferring content associations to the Portal Site library

When you enable manage pages, any web content pages that you have are converted to managed pages and added to the Portal Site library. However, the content that is associated with the web content pages remains in the original libraries. You can transfer this associated content to the Portal Site library with the internalize-content-mappings task.

**Note:** Administration pages are not intended to be managed pages and so are not included when you enable managed pages.

When you transfer the content association for a page to the Portal Site library, several things happen:

- The content that is referenced by the default content association for the page is copied to the portal page site area for the page. Only the default content association is affected; other content associations for the page are ignored.

**Note:** Nested pages are not copied. Nested site areas are not copied in the following cases:
- The nested site area is referenced by the default association of another page.
- The nested site area has the same name as an existing site area for the same page.

- Template mappings and content elements that exist in the associated site area are copied over into the portal page. If the template mapping or element already exists for the page, the copy is not performed.
- The default content setting for the portal page is modified to reference the copied content.
- The configuration of any web content viewers on the page is updated to reference the content that is stored in the portal page site area. However, viewer configurations that use content paths are not affected.
To transfer content associations, run the internalize-content-mappings task from the `wp_profile_root/ConfigEngine` directory.

**Windows**
```
ConfigEngine.bat internalize-content-mappings
-PortalPage=target_page -IncludeDescendants=true_or_false
-DSynchronous=true_or_false -PortalAdminPwd=password
-DWasPassword=password
```

**AIX Linux Solaris**
```
./ConfigEngine.sh internalize-content-mappings
-PortalPage=target_page -IncludeDescendants=true_or_false
-DSynchronous=true_or_false -PortalAdminPwd=password
-DWasPassword=password
```

**IBM i**
```
./ConfigEngine.sh internalize-content-mappings
-PortalPage=target_page -IncludeDescendants=true_or_false
-DSynchronous=true_or_false -PortalAdminPwd=password
-DWasPassword=password
```

**z/OS**
```
./ConfigEngine.sh internalize-content-mappings
-PortalPage=target_page -IncludeDescendants=true_or_false
-DSynchronous=true_or_false -PortalAdminPwd=password
-DWasPassword=password
```

The following properties must be specified either on the command line or in the `wkplc.properties` file.

**PortalPage**
The object ID or the unique page name of the page for which you want to transfer content. If the target page is contained in a virtual portal, you must identify the virtual portal by specifying either the `VirtualPortalContext` parameter or `VirtualPortalHost` parameter.

**IncludeDescendants**
Specify true to transfer content for the target page and any child pages. To transfer content only for the target page, specify false. If not specified, the default value is true.

**Synchronous**
Specify true to perform the transfer synchronously. To perform the transfer asynchronously, specify false. If not specified, the default value is true.

**Verbose**
Specify true to output additional information to the log. To generate basic log information, specify false. If not specified, the default value is false.

**VirtualPortalContext**
Specify the virtual portal context that identifies the virtual portal. For example, vp1.

**VirtualPortalHost**
Specify the host name of the virtual portal. For example, vp.example.com.

**Important**: If the host name of the virtual portal is the same as the host name of the default virtual portal, you must also specify the `VirtualPortalContext` property. You can specify the `VirtualPortalHost` property by itself only if the host name is unique.

**PortalAdminPwd**
The administrator password for WebSphere Portal.
WasPassword

The administrator password for WebSphere Application Server.

Example commands:

- **Windows:** ConfigEngine.bat internalize-content-mappings
  -DPortalPage=example.page -DIncludeDescendants=true -DSynchronous=true
  -DPortalAdminPwd=password -DWasPassword=password

- **AIX Linux Solaris:** ./ConfigEngine.sh internalize-content-mappings
  -DPortalPage=example.page -DIncludeDescendants=true -DSynchronous=true
  -DPortalAdminPwd=password -DWasPassword=password

- **IBM i:** ConfigEngine.sh internalize-content-mappings
  -DPortalPage=example.page -DIncludeDescendants=true -DSynchronous=true
  -DPortalAdminPwd=password -DWasPassword=password

- **z/OS:** ./ConfigEngine.sh internalize-content-mappings
  -DPortalPage=example.page -DIncludeDescendants=true -DSynchronous=true
  -DPortalAdminPwd=password -DWasPassword=password

---

**Syndication properties**

You can tailor the syndication behavior of your web content environment by changing configuration settings such as the syndication interval and automatic syndication.

You define and manage syndication options in the `WCM WCMConfigService` service using the WebSphere Integrated Solutions Console.

**Changing the syndication interval**

Although the frequency of syndication is set by default during installation, you can change the syndication interval to better suit the needs of your environment.

For example, you might shorten the interval in an active authoring environment where users must collaborate heavily and rely on timely replication. Similarly you might lengthen the interval to avoid excessive replication of data that does not change often.

**Note:** The syndication interval applies to all syndication operations and cannot be specified separately for different syndicator-subscriber pairs.

To change the syndication interval, modify the `deployment.itemChangedTaskDelay` property. By default, the syndication interval is set to 30 seconds. Specify the number of seconds to use as the syndication interval, with a minimum of 0 seconds and a maximum of 65536 seconds. A value of 0 will prevent syndication from occurring. If you set the value to so short an interval that syndication cannot complete before the interval expires, syndication begins again when the previous syndication completes.

**Disabling automatic syndication**

In some cases you might choose to rely only on manual syndication to have complete control over when syndication occurs. To do this, you must disable automatic syndication. When automatic syndication is disabled, the syndication interval setting is ignored. This property should be set to the same value on both the syndicator and the subscriber.
To disable automatic syndication, specify the following property:

- Property name: `connect.moduleconfig.syndication.inittasks`
- Value: `false`

**Configuring a subscriber-only server**

A syndicator server uses several processes to gather and queue content for syndication. These processes can sometimes impact server performance when run. However, a subscriber-only server does not require these processes, so you can improve performance on the subscriber-only server by disabling the processes.

To do this, ensure that `deployment.subscriberOnly` property is set to `true`.

**Enabling secure syndication using SSL**

In order to enable and use SSL for syndication, the following properties must be changed in the `WCM WCMConfigService` to use the "https" protocol and the appropriate port.

- `deployment.itemDispatcherUrl`
- `deployment.syndicatorUrl`
- `deployment.subscriberUrl`

**Related tasks:**

- Setting service configuration properties

IBM WebSphere Portal comprises a framework of services to accommodate the different scenarios that portals need to address. Services are available for both WebSphere Portal and IBM Web Content Manager. You can configure some of these services.

---

**Enabling search for web content**

You use Portal Search to search for text displayed in web sites created by IBM Web Content Manager.

**Indexing web content**

In order to search for web content, your content must first be indexed by the WebSphere Portal search engine. Once the content has been indexed, you can run searches using the search center or using a search component. If you search for documents in the WebSphere Portal search center, be aware that you see search results for published documents only. Unpublished pending changes in a project are not included in the results.

**Creating a content source for a site area**

The WebSphere Portal search engine defines content sources that index your web content. All the child site areas and content items of the selected site area will be included in the index. Related content sources are grouped together in a search collection.

1. Go to **Administration > Search Administration > Manage Search**.
2. Select or create a new collection. The default search collection named `WebContentCollection` is provided by default.
3. Click **New Content Source**.
4. Select **WCM site** as the content source type.

5. Enter a name in the **Content Source Name** field.

6. Enter the following URL in the **Collect documents linked from this URL** field:

   **For a stand-alone server:**
   
   ```
   http://hostname:port_number/wps/seedlist/myserver?SeedlistId=library/sitearea1/childsitearea2
   &Source=com.ibm.workplace.wcm.plugins.seedlist.retriever.WCMRetrieverFactory&Action=GetDocuments
   ```

   You will need to replace **hostname**, **port_number**, **library** and **site area** with values appropriate for your site. If your library name or site area names contain spaces, you will need to replace the spaces with a "+" symbol. For example, the path `library one/site area one` would be instead be defined as `library+one/site+area+one`

   **For a cluster:**
   
   In this case you to use the host and port of the HTTP server:
   
   ```
   http://httpserver:port_number/wps/seedlist/myserver?SeedlistId=library/sitearea1/childsitearea2
   &Source=com.ibm.workplace.wcm.plugins.seedlist.retriever.WCMRetrieverFactory&Action=GetDocuments
   ```

   You will need to replace **httpserver**, **port_number**, **library** and **site area** with values appropriate for your site. If your library name or site area names contain spaces, you will need to replace the spaces with a "+" symbol. For example, the path `library one/site area one` would be instead be defined as `library+one/site+area+one`

   **For a virtual portal configured to use the URL Context as its access point:**
   
   ```
   &Source=com.ibm.workplace.wcm.plugins.seedlist.retriever.WCMRetrieverFactory&Action=GetDocuments
   ```

   You will need to replace **httpserver**, **port_number**, **virtualPortalContext**, **library** and **site area** with values appropriate for your site. If your library name or site area names contain spaces, you will need to replace the spaces with a "+" symbol. For example, the path `library one/site area one` would be instead be defined as `library+one/site+area+one`

   **For a virtual portal configured to use a different hostname as its access point:**
   
   ```
   http://vphostname:port_number/wps/seedlist/myserver/?SeedlistId=library/sitearea1/childsitearea2
   &Source=com.ibm.workplace.wcm.plugins.seedlist.retriever.WCMRetrieverFactory&Action=GetDocuments
   ```

   You will need to replace **vphostname**, **port_number**, **library** and **site area** with values appropriate for your site. If your library name or site area names contain spaces, you will need to replace the spaces with a "+" symbol. For example, the path `library one/site area one` would be instead be defined as `library+one/site+area+one`

   **Note:** The seedlist ID can be any of the following:
   
   - library
   - library/site area
   - library/site area/sub-site area/...
   - the JCRID of a site area

7. If the content to be indexed is secured, go to the **Security** tab and enter the user name and password of the user that will be used to access the secured site. You must then click **Create** on the search tab itself.

8. If your site uses remote actions, you will need to filter these out of your search index. Go to the **Filter** tab:
a. Type a name in the Rule Name field
b. Select Apply rule while Collecting documents
c. Select the rule type of Exclude
d. Select the rule basis of URL text
e. Type *\&wcmAuthoringAction=* in the URL text field
f. Click Create in the Filter tab

9. Click Create.

If you have multiple parent site areas and want your searches to run across all site areas, you can create a content source for each of them in the same collection. If you don’t want your searches to run across all parent site areas, create a separate collection for each parent site area or group of related parent site areas.

**Searching web content in a virtual portal**

Search services and search collections are separate for individual virtual portals and are not shared between individual virtual portals. You set up an individual search service and separate search collections for each virtual portal. These collections can be used to crawl and search the same set of documents.

If you are using a website that is shared across virtual portals, then to search that website in a virtual portal environment you must:

1. Create a new search collection for the virtual portal. You can create a new content source by copying the URL from your original search collection.
2. Create a new search component, or copy an existing search component, and configure it to use the new virtual portal search collection created in step 1.
3. Create a new search form, using an HTML component, configured to use the search component created in step 2.
4. Create a new content item to display the HTML component created in step 3.

You must perform these steps for each virtual portal in your system.

**Configuring Web Content Manager search options**

You can edit the following search options to manage how the search service works with Web Content Manager search forms

1. Edit the SearchService.properties file.
   File location:
   
   **Windows**  \wp_profile_root\PortalServer\wcm\shared\app\config\wcmservices
   **UNIX** \wp_profile_root\PortalServer\wcm\shared\app\config\wcmservices
   **IBM i**  \wp_profile_root\PortalServer\wcm\shared\app\config\wcmservices
   **z/OS** \wp_profile_root\PortalServer\wcm\shared\app\config\wcmservices
   
   2. Specify values for the search parameters.
Table 23. Parameters for the search service in Web Content Manager.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>SearchService.DateFormatString</td>
<td>This is used to set the date format when entering dates in search forms and for displaying search results. Enter a supported Java date format string. If this property is not set, then the default format is \textit{MMM dd yyyy HH:mm:ss z}.</td>
</tr>
<tr>
<td>SearchService.RecrawlInterval</td>
<td>This is the &quot;recrawl interval&quot; in hours.</td>
</tr>
<tr>
<td>SearchService.BrokenLinksExpirationAge</td>
<td>This is the default &quot;Broken links expiration age&quot; in days.</td>
</tr>
</tbody>
</table>
| SearchService.MetaFields           | This is used to specify additional elements to crawl when searching for Metadata. The format for this parameter value is: elementName,key1. To specify more than one Metadata field maps, use the following format: elementName1,key1;elementName2,key2;elementName3,key3. For example, to crawl for Metadata in a text element named metaText: 
  • SearchService.MetaFields=metaText,meta
  • elementName is the name of element you would like to search for Metadata. Any valid element with that name in a searchable site area or content item will be crawled.
  • key is the "key" that is specified in an element tag used as part of a search element design. In the previous example, the key of "meta" has been used. To render the content of the metaText element in a search element design, you would use the following tag: <Element context="autoFill" type="content" key="meta"/> |
| Note:                             | Only text elements and short text elements can be searched. Only site areas that have been configured to be searchable will be crawled. |
| SearchService.SearchSeed.ExcludeFileAttachments | Set this to "true" to prevent file resource component attachments from being included in the search results. If set to false, the files stored in file resource elements in content items can also be searched. Files stored in file resource elements in a site area can also be searched so long as a default content item has been selected. |

4. Restart the portal for the new settings to take effect.
**Configuring Search Center to search for web content**

You can use the Search Center to search for web content by adding a web content search collection to the Search Center.

1. Go to Administration → Search Administration → Manage Search → Search Scopes.
2. Click New Scope.
3. Click Select Locations and select your web content search collection.
4. Complete the search scope and click OK.

**Crawling web content with search seedlists**

Portal Search supports the use of seedlists to make crawling websites and their metadata more efficient and to provide content owners fine-grained control over how content and metadata are crawled. You can configure the portal to use seedlist support when crawling content generated with IBM Web Content Manager.

By default Portal Search is configured to use seedlist format 1.0 when indexing content for search collections. When used with web content, seedlist format 1.0 makes it possible to use the web content page type to render content found in the search results on the corresponding web content page. You can also include custom metadata fields from a web content item that will appear in the search seedlist but not in the HTML source.

Search seedlist 1.0 can make access control information available in a way that makes pre-filtering of contents possible. Pre-filtering provides the fastest filtering approach because it takes place in the search index level. An additional advantage of pre-filtering is that remote secured content sources can be searched from the portal. The filtering mode is defined as part of the search service configuration parameters.

**Note:** Support for generic seedlist 1.0 crawling is only available with IBM OmniFind® Enterprise Edition Version 9.1 and later.

**Using the search seedlist 1.0 format**

As of version 6.1.5, Portal Search is configured to support the IBM Web Content Manager search seedlist 1.0 format by default. Versions before 6.1.5 use Web Content Manager search seedlist format 0.9.

Search seedlist 1.0 provides several features:

- You can use the web content page type to render content found in the search results on the corresponding web content page.
- You can include custom metadata fields from a web content item that appear in the search seedlist but not in the HTML source.
- You can search within a specific library or site area, across all web content libraries, or across a list of libraries.
- You can perform incremental crawling of libraries for faster seedlist processing. With incremental crawling, when a crawl requests new items, only items that have been added, changed, or deleted since the previous crawl are retrieved.

**Important:** The syntax of the seedlist URL has changed with seedlist format 1.0. Older search collections created using seedlist format 0.9 cannot be reused or migrated to the new format. Be sure that you index all your content again after updating the Web Content Manager seedlist format from 0.9 to 1.0.
Search seedlist 1.0 can make access control information available in a way that makes pre-filtering of contents possible. Pre-filtering provides the fastest filtering approach because it takes place in the search index level. An additional advantage of pre-filtering is that remote secured content sources can be searched from the portal. The filtering mode is defined as part of the search service configuration parameters.

**Enabling support for search seedlist 1.0:**

If you want to use Portal Search to crawl your web content and leverage features like web content pages you must enable seedlist 1.0 support for the Portal Search crawler.

1. Log in to the portal as an administrator.
2. Click **Administration** in the tool bar.
3. Create a new search collection.
   a. Click **Manage Search > Search Collections**.
   b. Create a new search collection for your web content. Be sure that the new search collection uses the portal search service edited in the previous steps.
4. Add the following custom properties to the **WP ConfigService** resource environment using the WebSphere Integrated Solutions Console:
   a. `wcm.config.seedlist.version=1.0`
   b. `wcm.config.seedlist.servletpath=/seedlist`
   c. `wcm.config.seedlist.metakeys=<metakey1>,<metakey2>` This is an optional step and is only required if you want to specify your own metadata.

**Related tasks:**

[Setting service configuration properties]

IBM WebSphere Portal comprises a framework of services to accommodate the different scenarios that portals need to address. Services are available for both WebSphere Portal and IBM Web Content Manager. You can configure some of these services.

**Using the custom metadata field search support:**

With the search seedlist 1.0 support, custom metadata fields specified on content items are added to the search seedlist as metadata information, without requiring the metadata to appear in the HTML source for the content items.

1. Log in to the WebSphere Integrated Solutions Console.
2. Click **Resources > Resource Environment > Resource Environment Providers**.
3. Click **WP ConfigService**.
4. Under **Additional Properties**, click **Custom Properties**.
5. Click **New**, and enter the property name `wcm.config.seedlist.metakeys`, and set the string value to a comma-delimited list of your own metadata (for example, `<metakey1>,<metakey2>`).

Add the names of the text element from your content that should be included in the search results to the `wcm.config.seedlist.metakeys` property. If you want to add more than one text element, separate them with commas. The name of the text element on your content item that should be included in the search seedlist must match the name configured for this configuration key.

For example, set `wcm.config.seedlist.metakeys=language,region` in the WP ConfigService resource environment provider, and add an IBM Web Content Manager text component as an element with the name `language` to a content item or authoring template. In your content item you can enter the value
german into the text component for the language. After saving the content item, the search crawler will add the value german into a metadata field called language within the search seedlist. Then you can filter the search results based on your metadata information.

6. Click OK, and save the changes to the master configuration.

7. Restart the portal.

**Seedlist 1.0 REST service API:**

The IBM Web Content Manager API for retrieving application content through a seedlist is based on the REST architecture style. To obtain seedlist content, third party crawlers or administrator applications need to construct and send only HTTP requests to the application servlet.

All REST API requests are synchronous calls. The order of the parameters in the requests does not matter. The parameter names are case-sensitive and must be entered in the format described here. An HTTP error response (for example, status code 404) is generated in the following situations:

- An unknown or unsupported parameter is submitted as part of the request.
- Web Content Manager cannot resolve the site area path or ID.
- Web Content Manager cannot find any items.
- The search seedlist enterprise application (Seedlist_Servlet) is not running.

The request is a standard HTTP GET command. The URL is formed by combining the seedlist servlet host name, port number, and path, followed by a collection of input parameters separated by ampersand (&) characters. The input parameters are entered as name-value pairs.

For example:

```
```

- **library_list**
  
  One or more web content libraries, separated by commas. If no value is specified, all libraries are used.

- **action**

  The action to perform on the request. The following actions are available:

  - **GetDocuments**
    
    Retrieves a list of content items with their associated information.

  - **number_of_entries**

    For each seedlist page that is returned, this value specifies the number of entries in the list of content items. If no value is specified, 100 items are returned.

**Examples**

In these examples, replace the following variables with values that are appropriate for your environment:

- **host_name**
- **virtual_portal_host_name**
- **http_server**
- **port_number**
- **library**
- **site_area**
For the **SeedlistId** parameter, you can specify the value in the following formats:

- No value
- A specific library (for example, *library1*)
- A specific site area (for example, *site_area1*)
- A list of libraries, separated by commas (for example, *library1,library2,library3*)
- The JCRID of a site area

**Retrieve a maximum of 100 items from a stand-alone server using the path to the site area**

```
http://host_name:port_number/wps/seedlist/
myserver?SeedlistId=library/site_area
&Source=com.ibm.workplace.wcm.plugins.seedlist.retriever.WCMRetrieveFactory&Action=GetDocuments
```

**Retrieve a maximum of 200 items from a stand-alone server using the ID of the site area**

```
http://host_name:port_number/wps/seedlist/
myserver?SeedlistId=site_area_id
&Source=com.ibm.workplace.wcm.plugins.seedlist.retriever.WCMRetrieveFactory&Action=GetDocuments&Range=200
```

**Retrieve a maximum of 100 items from a specific library**

```
http://host_name:port_number/wps/seedlist/
myserver?SeedlistId=library
&Source=com.ibm.workplace.wcm.plugins.seedlist.retriever.WCMRetrieveFactory&Action=GetDocuments
```

**Retrieve a maximum of 100 items from all libraries**

**Note:** To use all libraries, leave SeedlistId value empty.

```
http://host_name:port_number/wps/seedlist/myserver?SeedlistId=
&Source=com.ibm.workplace.wcm.plugins.seedlist.retriever.WCMRetrieveFactory&Action=GetDocuments
```

**Retrieve a maximum of 100 items from a specified list of libraries**

```
http://host_name:port_number/wps/seedlist/
myserver?SeedlistId=library1,library2
&Source=com.ibm.workplace.wcm.plugins.seedlist.retriever.WCMRetrieveFactory&Action=GetDocuments
```

**Retrieve a maximum of 100 items from a cluster**

**Note:** When referencing a cluster, specify the request with the host name and port number of the HTTP server.

```
http://http_server:port_number/wps/seedlist/
myserver?SeedlistId=library/site_area
&Source=com.ibm.workplace.wcm.plugins.seedlist.retriever.WCMRetrieveFactory&Action=GetDocuments
```

**Retrieve a maximum of 100 items from a virtual portal configured to use the URL context as the access point**

```
http://http_server:port_number/wps/seedlist/myserver/
virtual_portal_context?SeedlistId=library/site_area
&Source=com.ibm.workplace.wcm.plugins.seedlist.retriever.WCMRetrieveFactory&Action=GetDocuments
```

**Retrieve a maximum of 100 items from a virtual portal configured to use a different host name as the access point**

```
http://virtual_portal_host_name:port_number/wps/seedlist/
myserver?SeedlistId=library/site_area
&Source=com.ibm.workplace.wcm.plugins.seedlist.retriever.WCMRetrieveFactory&Action=GetDocuments
```
Important: You can access the REST API for the Web Content Manager search seedlist 1.0 with a secured connection (HTTPS) or with an unsecured connection (HTTP). Depending on the method, ensure that you use the correct port. However, if you access this REST API with an unsecured connection, you are automatically redirected to a secured connection.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SeedlistID</td>
<td>No default; must be specified.</td>
<td>Identifies the seedlist. This parameter can be specified in the following ways:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• An empty value causes all libraries to be used.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• A specific library (for example, library1)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• A specific site area (for example, site_area1)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• A list of libraries, separated by commas (for example, library1,library2,library3)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• The JCRID of a site area</td>
</tr>
<tr>
<td>Start</td>
<td>0</td>
<td>Defines the start number for currently returned section.</td>
</tr>
<tr>
<td>Range</td>
<td>100</td>
<td>Defines the number of returned entries for current section.</td>
</tr>
<tr>
<td>Date</td>
<td>No default. If not specified, all applicable results are returned.</td>
<td>Indicates that entries (documents) that were updated after this date are retrieved. The date format (compliant to standard ISO 8601) is the following: dateTtimezone, where date is yyyy-MM-dd, time is HH:mm:ss, and timezone is ±hhmm. This format includes time zone information, which is critical if the client and server are in different time zones. Important: Proper HTML URL encoding must be performed (for example, represent the plus symbol + as %2B).</td>
</tr>
<tr>
<td>Parameter</td>
<td>Default Value</td>
<td>Description</td>
</tr>
<tr>
<td>----------------</td>
<td>---------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Action</td>
<td>GetDocuments</td>
<td>Defines requested action to execute.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- GetDocuments retrieve all underlying documents.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- GetNumberOfDocuments returns the number of all underlying documents, typically for debug purposes. This value must be the same as the number of all documents returned from an appropriate GetDocuments request.</td>
</tr>
<tr>
<td>Format</td>
<td>ATOM</td>
<td>Defines the output format: ATOM / HTML / XML.</td>
</tr>
<tr>
<td>Timestamp</td>
<td>No default.</td>
<td>Indicates the content provider timestamp from a previous crawling session. The timestamp represents a snapshot of the content and allows the crawler to get only the content changes on the next crawling. This parameter is used for incremental crawling.</td>
</tr>
</tbody>
</table>

**Using the search seedlist 0.9**

Although Portal Search is configured to support the search seedlist 1.0 format by default, you can reconfigure the portal to use the standard seedlist 0.9 format when searching for web content with the Search Center. For example, you might choose to use seedlist format 0.9 because you want to make use of older search collections or because you retrieve the seedlist 0.9 contents using the seedlist URL, which uses a different syntax from the URL used with the search seedlist 1.0 format.

**Important:** With IBM WebSphere Portal Version 7, search seedlist format 0.9 is deprecated. Although you can still use seedlist format 0.9, it is recommended that you transition to seedlist format 1.0 to ensure future compatibility.

To use the seedlist format 0.9, you essentially disable the default support for the seedlist format 1.0.

1. Log in to the portal as an administrator.
2. Click **Administration** in the tool bar.
3. Disable search seedlist 1.0 support for IBM Web Content Manager.
   b. Click **Resources > Resource Environment > Resource Environment Providers**.
   c. Click **WP ConfigService**.
   d. Under **Additional Properties**, click **Custom Properties**.
   e. Remove the property named `wcm.config.seedlist.version`.
   f. Remove the property named `wcm.config.seedlist.servletpath`. 
g. If it exists, remove the property named `wcm.config.seedlist.metakeys`.

h. Click **OK**, and save the changes to the master configuration.

i. Restart the portal.

---

**Managing tagging and rating for web content**

When using tagging and rating with web content, the web content viewer provides additional scope options for the filtering of tagging and rating results. Because changes in the web content system can affect the accuracy of the tagging and rating information used by the portal, it is important to keep the scope information up to date by synchronizing the scopes on a regular basis.

**Related tasks:**

“Tagging and rating plug-ins for web content” on page 244

Just as you can tag and rate portal resources like pages and portlets, you can also tag and rate content items generated with IBM Web Content Manager and displayed with the web content viewer. Two plug-in components are available to support the tagging and rating of content items in your web content system. You can add the `[Plugin:tags]` component and `[Plugin:ratings]` component in a presentation template to quickly integrate tagging and rating widgets into the current content item.

**Using tagging and rating scopes with web content**

Scoping is typically used to filter the tag cloud or ratings overview according to hierarchical metadata attached to the resources being tagged. When applying tagging and rating to web content, you can scope these display components according to authoring template, category, or content item parent.

You can configure the advanced settings of the web content viewer to limit results to show only tags or ratings associated with one or more of the following scopes:

- The parent of the content item being displayed (for example, a site area).
- The authoring template that is used to generate the content item or site area being displayed.
- The categories used to profile the content item being displayed. In this way, you can manage scopes from within your web content system simply by defining taxonomies for your content items.
Related tasks:

“Synchronizing scopes for web content”
When users are tagging or rating web content, the web content viewer provides the tagging or rating information to the portal, where it is stored. If information in the web content system changes, this can cause the tagging and rating information stored in the portal to be out of sync. This can happen, for example, if content items are moved or category information changes. To ensure the tagging and rating information is current, synchronize the scopes used for web content. You can set up automatic synchronization according to different conditions or perform a manual synchronization as needed.

Related reference:

Advanced options
The Advanced Options settings specify link broadcast behavior between web content viewers and whether context processor plug-ins are applied.

Synchronizing scopes for web content
When users are tagging or rating web content, the web content viewer provides the tagging or rating information to the portal, where it is stored. If information in the web content system changes, this can cause the tagging and rating information stored in the portal to be out of sync. This can happen, for example, if content items are moved or category information changes. To ensure the tagging and rating information is current, synchronize the scopes used for web content. You can set up automatic synchronization according to different conditions or perform a manual synchronization as needed.

Related information:

Setting service configuration properties

Synchronizing scopes when items change
To automatically perform scope synchronization whenever an item changes in the web content system, specify the tagging.syndication.enableItemModificationSynchronization property in the Web Content Manager configuration service.

Note: This type of synchronization only works for individual item changes. For example, this type of synchronization is not automatically performed when an entire site area or folder is moved. To synchronize scopes after such a change, you can perform synchronization manually.

1. Log in to the WebSphere Integrated Solutions Console (http://hostname.example.com:10027/ibm/console).
2. Click Resources > Resource Environment > Resource Environment Providers.
3. Click WCM WCMConfigService.
5. Add the tagging.syndication.enableItemModificationSynchronization property.
   a. Click New, and enter the property name tagging.syndication.enableItemModificationSynchronization.
   b. Set the string value to true.
6. Click OK, and save the changes to the master configuration.
7. Restart the portal.
Synchronizing scopes after syndication
To automatically perform scope synchronization whenever syndication occurs, specify the tagging.syndication.enableTagSynchronization property in the Web Content Manager configuration service.

1. Log in to the WebSphere Integrated Solutions Console (http://hostname.example.com:10027/ibm/console).
2. Click Resources > Resource Environment > Resource Environment Providers.
3. Click WCM WCMConfigService.
5. Add the tagging.syndication.enableTagSynchronization property.
   a. Click New, and enter the property name tagging.syndication.enableTagSynchronization.
   b. Set the string value to true.
6. Click OK, and save the changes to the master configuration.
7. Restart the portal.

Scheduling scope synchronization
You can schedule scope synchronization to be performed at specific times by defining the schedule with the XML configuration interface.

1. Verify whether any scheduled synchronizations have already been defined for the portal.
   a. Create an export file that you can use with the xmlaccess command. Here is an example of a request you can use to query the current configuration:

   ```xml
   <?xml version="1.0" encoding="UTF-8"?>
   <request type="export" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xsi:noNamespaceSchemaLocation="PortalConfig_8.0.0.xsd">
     <portal action="locate">
        <task action="export" name="com.ibm.portal.cp.SynchronizationTask"/>
     </portal>
   </request>
   
   b. Run the xmlaccess command, specifying the export file. The resulting output file contains any scheduled synchronization times that are defined in the portal.

2. Set the synchronization schedule.
   a. To set a time for a scheduled synchronization, create an XML request document. For example, to schedule a synchronization to occur at 15:36 hours every day, you would use a request like this:

   ```xml
   <?xml version="1.0" encoding="UTF-8"?>
   <request type="update" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xsi:noNamespaceSchemaLocation="PortalConfig_8.0.0.xsd">
     <portal action="locate">
        <task action="create" name="com.ibm.portal.cp.SynchronizationTask">
           <startTime>15:36</startTime>
        </task>
     </portal>
   </request>
   
   For each scheduled synchronization, create a separate task element and specify the time with a startTime element.
b. Run the `xmlaccess` command, specifying the file containing the scheduling request. Scope information for the web content system will then be synchronized automatically according to the schedule you defined.

3. Optional: If you want to set a minimum time before subsequent synchronizations are performed, specify the `tagging.syndication.minimumTagSynchronizationTimeInterval` property in the Web Content Manager configuration service.
   b. Click Resources > Resource Environment > Resource Environment Providers.
   c. Click WP ConfigService.
   d. Under Additional Properties, click Custom Properties.
   e. Click New, and enter the property name `tagging.syndication.minimumTagSynchronizationTimeInterval`.
   f. Set the string value to the number of seconds between synchronizations.
   g. Click OK, and save the changes to the master configuration.
   h. Restart the portal.

Related information:
- Setting service configuration properties
- Working with the XML configuration interface
- XML configuration reference

Synchronizing scopes manually

If you have not enabled automatic synchronization of the scopes used for web content or if you want to perform synchronization outside of a scheduled synchronization period, you can manually start the synchronization process.

To manually perform synchronization, run the `cp-sync` configuration task or submit an XML request to the portal by using the XML configuration interface.

- To perform synchronization with a configuration task, run the following task from the `wp_profile_root/ConfigEngine` directory:
  - Windows: `ConfigEngine.bat cp-sync -DWasPassword=password -DPortalAdminPwd=password`
  - UNIX: `./ConfigEngine.sh cp-sync -DWasPassword=password -DPortalAdminPwd=password`
  - IBM i: `ConfigEngine.sh cp-sync -DWasPassword=password -DPortalAdminPwd=password`
  - z/OS: `ConfigEngine.sh cp-sync -DWasPassword=password -DPortalAdminPwd=password`

- Create an XML request file and submit it using the `xmlaccess` command. Here is an example of a request you can use to start synchronization:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<request type="update" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="PortalConfig_7.0.0.xsd">
  <portal action="locate">
    <task action="create" name="com.ibm.portal.cp.SynchronizationTask"/>
  </portal>
</request>
```

Related information:
- Working with the XML configuration interface
Chapter 5. Setting up a site

Setting up a site includes customizing the site’s look-and-feel, creating pages, and adding content to the site. You can also add wikis and blogs to your site and let users tag and rate content on your site.

Site toolbar

The site toolbar provides access to regularly used features for managed pages from a central location. Key tasks that you can perform include working with projects, adding and editing content, changing page properties, and customizing style and layout.

Note:
• The site toolbar requires the Portal 8.0 theme.
• Depending on the authorization level of the user, some toolbar options might not be displayed. For example, for users without edit access to the page, the toolbar does not include the links to specify view or edit mode.

View mode

The toolbar is initially displayed in view mode and provides basic information about the page and access to authoring tasks.

1. Links to primary areas of the portal are included, enabling quick access to applications like the authoring portlet.

2. With the Edit Mode and View Mode links, you can specify the mode for the page. View mode shows the draft state of the currently selected project. Edit mode enables you to modify the page and its content as part of the currently selected project.

3. The project menu displays the currently selected project. Changes that you make to the page or content on the page are made as draft changes that are associated with the selected project. The project menu provides several additional features:
   • To quickly change projects, the menu lists the five most recent projects that are associated with the page and that are still active.
- The **Published site** option clears the current project and displays the published version of the page. Although you can make edits while the **Published site** option is selected, do your editing while in a project. Edits made to the published site are effective immediately and do not benefit from the workflow safeguards that are provided by projects.

- You can create projects. When you create a project, the new project is automatically selected as the current project.

- If you want to use a project that is not listed as a recent project, you can search or browse all available projects.

- You can perform management tasks for the project. Such tasks include working with a list of all project items or changing project properties.

**Note:**
- If a user has a higher access control permission than a privileged user for the page, the user can create or select a project when editing the page. The project menu is not available for users that have only privileged rights for a page, because their edits are private customizations of the page.

- If managed pages are disabled, the project menu is not displayed. Instead, a toggle button is displayed that enables the user to switch from view mode to edit mode.

**Edit mode**

When you click **Edit Mode**, the toolbar expands to display additional information about the page and the selected project and to provide access to editing features. The following figure shows the site toolbar in edit mode.

![Figure 2. Site toolbar expanded in edit mode.](image)

1. The active project is indicated by the project menu. Any edits that you make to the page and its content are saved as drafts in the current project.

2. Authoring actions are available in edit mode, including saving page changes as a draft or discarding changes that are not saved.

**Note:** Users can revert actions that they have done to the page by using the **Revert** button. Users can revert the following actions:
- Adding content
- Changing the style
- Changing the Layout
• Moving a portlet
• Hiding a portlet
• Portlet drag and drop
• Deleting a portlet

Users cannot revert the following actions:
• Creating a page
• Moving a page
• Deleting a page

When the user saves the page by clicking Save, all edit actions are committed and can no longer be reverted.

3. The expanded area of the toolbar provides tabs that show information about the page and project and enable you to customize the page. The tabs include:

   **Overview**
   Displays page and project information:
   • Page information includes properties like name, friendly URL, and assigned workflow. Web content associations and community associations are also displayed. If you have sufficient access rights, you can edit the properties and associations by selecting that section of the toolbar and clicking edit.
   • Project information includes a list of content items and pages that are part of the current project. You can edit the items and pages by selecting that section of the toolbar and clicking read or goto. If no project is selected, no project information is displayed.

   **Content**
   Provides categories to organize components that you can add to the page, such as portlets, iWidgets, and web content.

   **Style**
   Provides predefined themes that you can select to change the appearance of the page.

   **Layout**
   Provides predefined formats that you can select to change the layout of the portlets and widgets on the page.

4. You can maximize and minimize the toolbar while remaining in edit mode.

**Previewing as another user**

You can preview changes to your website without logging out and logging on again as another user. This preview capability enables you to quickly verify that users with different access levels see only content that they are authorized to see. You can preview changes as a specific user or as an unauthenticated user.

While in edit mode, click More, and then click either As User or As Unauthenticated User. The page is updated according to the user that you are impersonating.

To stop previewing, click Stop Previewing.

You can modify the preview function to tailor preview behavior in the following scenario:
• User1 creates a project and a page within the project. The new page contains a portlet.
• User1 then attempts to preview the page as User2. In this case, User2 has the permissions to view the page but does not have the permissions to view the portlet that is on the page.

• User1 receives the following message: You are not authorized to use this portlet. This message displays to make it clear that a portlet is rendered here, if the user has the required permissions.

You can suppress the message in this scenario.

For more information about the access permissions that are required for previewing as another user, see Access control for managed pages.

1. Log on to the WebSphere Integrated Solutions Console as an administrator.
2. Click Resources > Resource Environment > Resource Environment Providers.
3. Click WP ConfigService, and then click Custom properties.
4. Click portlets.unauthorized.visible.project. Set the value to false to suppress the message. The default value is true.

Related concepts:
“Access control for managed pages” on page 162

Access control for managed pages provides more capabilities than access control for standard portal pages. In addition to the access control features available for pages through portal administration, you can also apply IBM Web Content Manager features, like workflow and syndication, to access control.

Libraries

Create a set of libraries to store different types of web content items for different websites or different teams.

You can create libraries with the Web Content Libraries administration portlet.

Important: When creating a web content library, ensure that the name of the library does not match the URL context of any virtual portals on the same server. If the name of a library and the URL context of a virtual portal have the same value, incorrect rendering behavior can result.

Web content libraries

Your web content system can contain multiple libraries. The number of libraries required is determined by the type of website you are creating, and the types of users who require access to each library.

In most systems you will need a minimum of two libraries:

1. A design library where you store all the items required for the web content system itself
2. A content library used to store the content developed by your content creators

Separating your site into these libraries enables you to better control the access to each library, and also allows you to setup different syndication strategies for each library.
Example:

Human resource and marketing content are stored in separate libraries.

A third library is used to store brand-related content (images, presentation templates, brand related text and HTML components).

Content from the branding library can be accessed and used by the human resource and marketing but it is read-only. Only users with editor access or higher to the branding library can edit that library.

Related tasks:
Creating web content libraries
You create web content libraries in the WebSphere Portal administration portlet.

Web content library default items
When you create a web content library, you can choose to include a set of default web content items in the new library. These can be used as a starting point for your Web Content Manager system and web site.

Default items
The following items are created when you select Include default items in the new library when you create a library.

Workflow items:
- A workflow named Express Workflow with a single workflow stage named Publish Stage using the publish action named Publish.
- A workflow named Three Stage Workflow using the following workflow stages:
  - Draft Stage
  - Publish Stage using the publish action named Publish.
  - Expire Stage using the expire action named Expire.

Authoring template:
The authoring template is named Article and contains a single rich text element named Body and uses the Express Workflow as the default workflow for content items created using this authoring template.
Presentation template:
The presentation template is named Simple Article Layout.

Site area and content items:
- The content items named Sample Article and Sample Article 2 are stored in the site area named Articles.
- The site area named Articles contains a template map between the authoring template is named Article and the presentation template named Simple Article Layout.

Components:
- The authoring tool named Article Toolbar is used to add New and Edit functions to the rendered page. It is referenced in the presentation template named Simple Article Layout.
- The menu named Articles List is used to display a list of content items on the rendered page. It is referenced in the presentation template named Simple Article Layout.

Access controls
As the web content library default items are configured to inherit their access settings from the library they are stored in, users are not able to access these items until you have configured the access settings of the library.

Using the default items
The default items are best displayed using a web content viewer portlet:
1. Go to Administration > Manage Pages.
2. Select Create New Page From.
3. Under Web Content Mappings, select the site area named Articles from your web content library.
4. Complete the rest of the form and click OK.
5. Edit the page layout of the new page and add a Web Content Viewer (JSR 286) portlet to the page.
6. The content item named Sample Article is displayed on the page.

Pages
A page is an organization element that contains content. There are different types of pages in portal. There are also different interfaces for creating and managing pages.

The portal theme allows you to perform the following tasks:
- Create, reorder, delete, and edit the properties of pages, labels, and URLs
- Reorder pages, labels, and URLs
- Assign access to pages, labels, and URLs
- Move pages to a new location in the portal hierarchy

Both administrators and users with appropriate access can create and delete pages. Users can delete only the pages they create or the pages for which they have at least Manager access.
Page creation and navigation

When you create a page, you create it in relationship to existing pages, as either a sibling or child. The page hierarchy affects the generated navigation.

Page creation

There are two user interfaces for creating pages, the Manage Pages portlet and the site toolbar. Pages created from either user interface are created as managed pages. Managed pages are stored in a web content library called Portal Site. They also have other benefits such as workflow. If you disable managed pages, you cannot create pages from the site toolbar.

Navigation

From any page, you can create a child or sibling page using the site toolbar. You can move the page if you need to.

If you create a page from the administration Manage Pages portlet, you must select the content root and find the right place in the hierarchy.

Friendly URLs and URL mapping

When creating a URL mapping, creating a page, or modifying a page, the URL mappings and friendly URLs cannot match, partially overlap, or interfere with each other. For example, do not use strings such as home, ibm, ibm.com. Also do not use strings that are used as URL mappings or friendly URLs in your portal already. Conflict can cause browser redirect loops. Sometimes the loop occurs without an error message.

To determine such strings, use the XML configuration interface to create an export from your portal. Then scan the exported file for the string that you want to use for your URL mapping or for your friendly URL.
Related concepts:

“Managed pages” on page 152
Managed pages streamline site management in your portal by simplifying how you create pages and add content. Because page information and content are stored in web content libraries, you can more easily coordinate and publish changes with syndication.

Related tasks:

Creating page templates
To simplify the creation of pages, define pre-configured pages that can be used as templates when creating new pages.

“Disabling managed pages” on page 103
Disable support for managed pages by running the disable-managed-pages configuration task.

Managing pages
Manage Pages allows you to create, edit, activate, order, and delete pages as well as external Web pages and labels. Available tasks depend on which item is selected. Each page can contain multiple pages. All pages on which you have the User or greater role are displayed in a navigation menu. You must expand pages to access nested pages. The options that you see are dependent upon your access level.

Creating a page from the site toolbar

A page displays content, such as portlets and other pages. Pages organize your site information. As you create pages, you also create new navigational elements to the site. You can create a page under an existing page or you can create a page that is a peer to an existing page. When you create a page you can also reference an existing page, apply a layout, and select supported markups.

Your access role determines the kind on page you can create.

Public pages
You must have the Administrator, Manager, or Editor role assignment to create a public page.

Custom page (derived page)
You must have the Editor and Privileged User role assignment to create a custom page.

If you have Editor role on the derived page, you can change anything except markups.

If you have a Privileged User role on the derived page, you can change the title, skins, or layout on the derived page. This role is restricted for a layout, by the derivation parent page.

If you reference an existing page, layout, supported markups, locks, skins, portlet list, and locale-specific titles are predetermined by the existing page you reference. Any changes to the original page results in the same change to all pages that are referenced.

When you create a page, you can give the page a friendly URL. Friendly URLs must be unique and the portal validates that the name you enter is unique. How

1. Click More > New Child Page or New Sibling Page, depending on where you want the page to appear in the hierarchy.
2. Type a unique URL in **Friendly URL Name** or use the name that was generated for you. This name creates a custom address for your page that is easy to remember and share.

**Note:** When creating a URL Mapping or creating or modifying a page, make sure it is unique. For example, do not use strings such as home, ibm, ibm.com. Also do not use strings that were already used as URL Mappings or friendly URLs in your portal. Otherwise browser redirect loops might occur, sometimes without an error message. To determine such strings, use the XML configuration interface to export your portal. Then scan the exported file for the string that you want to use for your URL Mapping or friendly URL. By default, the portal ensures that the friendly URL name that you enter is unique. However, this enforcement does not include derived pages with an inherited friendly name and siblings that are moved in by a personalization rule.

3. You can select a Page Template by clicking **Change**. The default Basic Page Template provided with portal creates an empty portal page.

4. Click **Create** to create the page and add new content. Click **Cancel** if you want to quit without creating the page.

### Page templates

Portal includes templates that are available for immediate use, Basic and Articles. You can also create templates and add them to the site toolbar.

Both the Basic and Articles templates are managed pages page. Managed pages are stored in a IBM Web Content Manager library called Portal Site.

#### Basic portal page

When you create a page, you can select a page template. The default page template is Basic for the basic portal page. When you create a page using this template, you can add portlets, content, and more to the page.

#### Articles

The articles template has two portlets that display content, Articles and List of Articles. This type of portlet is referred to as a content viewer portlet.

There is also a content association between the two portlets. The List of Articles portlet displays a list of the articles stores in the page site area. You can select an article from the list and it displays in the Article portlet.
 Managed pages streamline site management in your portal by simplifying how you create pages and add content. Because page information and content are stored in web content libraries, you can more easily coordinate and publish changes with syndication.

**Related concepts:**
“Managed pages” on page 152

**Related tasks:**
Creating page templates
To simplify the creation of pages, define pre-configured pages that can be used as templates when creating new pages.

### Creating page from a template using the site toolbar

Create pages quickly by using a page template. Pages created from page templates contain pre-configured portlets and settings.

1. Navigate to the page where you want to add the new page.
2. Click **Edit Mode** in the site toolbar.
3. Click **More**, and then click either **New Child Page** or **New Sibling Page**, depending on where you want to create the page.
4. Enter the page name.
5. Type a unique URL in the **Friendly URL Name** field. The friendly URL creates a custom address for your page that is easy to remember and share.

**Note:** When creating a URL Mapping or creating or modifying a page, make sure that URL Mappings and friendly URLs in your portal do not match, partially overlap, or otherwise interfere with each other. For example, do not use strings such as home, ibm, ibm.com, and do not use strings that are used as URL Mappings or friendly URLs in your portal already. Otherwise infinite browser redirect loops might occur, sometimes without an error message. To determine such strings, create an export from your portal by using the XML configuration interface. Then scan the exported XML result output file for the string that you want to use for your URL Mapping or for your friendly URL.

6. In the **Page Template** section, click **Change** to select the page template you want to use for the new page. The default template is the **Basic** template.

**Note:** If you are creating a public page, you must have at least User role access to the template page you are using.

7. Click **Create**.

### Page layout and style

Page layout defines how information (content, applications, and so on) is organized on your page. Page style defines the color scheme that is applied to the page.

Portal includes layouts and styles that are available for immediate use. You can easily select a layout and style from the site toolbar. In addition to the included layouts and styles you can create additional ones as needed.

### Changing page style

Change the appearance of a page by selecting a style from the list of predefined themes. When you select a theme, it is previewed on the page. You can add custom styles to the list of page themes that are available for selection.
To change the appearance of a page, proceed as follows:
1. Navigate to, or select the page you want to change and click **Edit Mode**.
2. Click the **Style** tab.
3. Select a category from the menu on the side.
4. Filter down the results displayed by typing search terms into the input box.
5. Page through the content by using the scrollbar.
6. To preview a style, click that style.
7. To commit a style change to the server, click **Save Draft**.
8. Close the toolbar by clicking the **Minimize** button or by clicking the **Edit Mode** toggle button to return to **View Mode**.

**Note:** The default style available in the default portal installation resets the page to have no custom style sheet applied. Therefore a page that has the default style applied simply inherits the styles applied to its ancestor pages.

**Changing page layout**

You can select a page layout from a list of predefined formats. Portlets and widgets are automatically positioned into the new layout. You can add custom layouts that you define to the list of page formats available for selection.

**Note:** The portal theme does not support locked containers.
1. Navigate to the page where you want to change the layout and click **Edit Mode** to open the toolbar.
2. Click to the **Layout** tab.
3. To select a layout, click the layout item. No changes in the content are displayed at this time.

**Note:** You can filter the layouts shown by typing space separated search terms into the input box in the toolbar.
4. To commit a layout change to the server, click **Save Draft**. The page automatically refreshes to render the layout change.

**Moving a page**

You can move a page and its children to another location in the portal hierarchy.
1. Navigate to the page you would like to move and click **Edit Mode**.
2. Click **More > Move Page**.
3. From the dialog, select the location to move the page.
4. Click **Save** to complete the move.

An alternative to the inline move page dialog is the Manage Pages portlet.

**Comparison of features support by types of portal pages**

Learn about considerations which types of portal pages and themes you can use for which purposes and what they support. The following table compares which type of page supports which portal features.

<table>
<thead>
<tr>
<th>Portal feature to be supported</th>
<th>Portal pages from before portal Version 8.0</th>
<th>Page builder page starting with portal Version 8.0</th>
</tr>
</thead>
</table>
Table 24. Support of portal features by different types of pages (continued)

<table>
<thead>
<tr>
<th>Portal feature to be supported</th>
<th>Portal pages from before portal Version 8.0</th>
<th>Page builder page starting with portal Version 8.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use IBM portlet</td>
<td>yes</td>
<td>• server side aggregation: yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• client side aggregation: no</td>
</tr>
<tr>
<td>Use standard portlet JSR186 or JSR268</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Use IWidget Version 2.1</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>Portlet consumed by using WSRP</td>
<td>yes</td>
<td>yes</td>
</tr>
</tbody>
</table>

**Communications**

<table>
<thead>
<tr>
<th>Portlet-to-portlet communication</th>
<th>yes</th>
<th>yes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Widget-to-widget communication</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>Widget-to-portlet and portlet-to-widget communication</td>
<td>no</td>
<td>• server side aggregation: no</td>
</tr>
<tr>
<td>Add or remove components</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Change container layout</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>Change style</td>
<td>no</td>
<td>no</td>
</tr>
</tbody>
</table>

**Explicit page derivation:**
- portal pages derived from portal pages
- page builder pages derived from page builder pages

| Add or remove components        | yes | yes |
| Change container layout         | yes | yes, You need to assign a different template. |
| Change style                    | no  | yes |

**Infrastructure**

| Dynamic user interface          | yes | no  |
| Other portlet services         | yes | • server side aggregation: yes |
|                                |     | • client side aggregation: depends |

Related concepts:

Page Builder themes and skins
The Page Builder theme is still supported but it is not the recommended approach for themes in version 8.

Projects and workflows

Projects allow you to change a set of items on your site and ensure that they are published together at the same time. A new project has a default workflow that allows content to be reviewed and approved before it is published.
Projects overview

Projects allow you to change a set of items and ensure that they are published together at the same time.

- You can either create a project directly, or base it on a previously created project.
- Once created, you can both add existing items to the project and create new items.
- All items are added to a project with a status of draft.
- The project itself cannot be published until all items in the project are ready to be published.
- Each item in a project can go through a separate change approval process.
- Projects themselves can also be processed through an approval process.

Related tasks:

Creating a project

You use a project to manage changes to a set of items.

Project states

When working with a project, the project progresses through a series of states.

Active

A project which has draft items in it is considered "active". These items can be individually approved until they reach a state of "pending".

Syndicating

If "All items" or "Live and project" syndication has been enabled for a library, a status of "Syndicating" is displayed on the project until all items on both the syndicator and subscriber have reached a state of pending.

Review

If any approvers have been assigned to a project, the project can be submitted for review once all items in the project are in a pending state. If approved, the project then progresses to pending.

Pending

A project which contains only items in a "pending" state, or has been approved, is itself considered "pending". If the project publish option is set to automatic this state is skipped. When the project is published manually or when the publish date is reached, the project moves to the "publishing" state.

Publishing

This is the state where all items in the project move from pending to published.

Failed

Indicates that one or more project items failed to publish.

Published

Once all items are published the project achieves a state of "published".

Project review state

If you select at least one approver for a project, the project must be submitted for review before it can be published.

To enforce a review, a user with editor access or higher clicks Submit for Review from the toolbar once all the items in the project are "pending". A project can either require the approval of only a single approver or all approvers. Only when the project is approved will the project itself become pending. The project is then either automatically or manually published depending on how the project has been configured.
Once a project is in the review state:

- Users with editor access or higher can return a project to an active state by clicking **Withdraw from Review**.
- Approvers can approve a project by clicking **Approve Project**. A project can either require the approval of only a single approver or all approvers.
- Approvers can decline a project by clicking **Reject Project**. The project is returned to the active state.
- If joint approval is specified, approvers can withdraw an approval while the project is still in review by clicking **Withdraw Approval**. The user's approval is withdrawn, but the project remains in a state of review.

**Ways to publish a project**
Publishing a project refreshes the live web site with your changes.

There are three publishing methods used by projects:

**Date**  When **Date** is selected, all items are published as soon as all the items in the project reach a state of "pending" and the publish date selected in the project is reached. You can also click **Publish** in the project form before the date being reached once all items in the project reach a state of "pending".

**Manual**  When **Manual** is selected all items remain in a state of "pending" until the project is manually published by clicking **Publish** in the project form. The **Publish** button is not activated until all items in the project reach a state of "pending". Only users with editor access or higher to a project can publish a project.

**Automatic**  When **Automatic** is selected all items are published as soon as all the items in the project reach a state of "pending".

Once the project is published, you can continue to review the status and version history of the items in the project by opening the project form.

**Note:** You and other users can update the same items in multiple projects. Messages are displayed when you move an item to the publish state to warn you when this has occurred.

**Projects and syndication**
Projects are included in syndication, the method used by IBM Web Content Manager to replicate data from a web content library on a syndicator server to a web content library on a subscriber server. Although projects will be syndicated with other items in a library that is being syndicated, you cannot use the subscriber copy of the project to update or publish your project. Work with projects on the syndicator server only.

When a project is syndicated, the publish method on the subscriber is automatically changed to "Deferred to Syndication", and the following actions are disabled on the subscriber:

- Publish project
- Add to project
- Remove from project
- Mark for deletion
- Cancel deletion
This means that the project on the subscriber cannot be updated or published unless it receives updates from the syndicator.

**Projects and custom workflow actions**

Each workflow stage contains default actions, but you can also create custom workflow actions by creating a custom workflow plug-in.

You can assign custom workflow actions to run when a project enters a specific state in a project. For example, you could create custom workflow action that could:

- verify web standards compliance for all assets within the project including items that are not traditionally workflowed, such as components and presentation templates.
- automatically reject a project if it is in review for a certain period of time.
- automatically delete a project when it is successfully published.

**Workflow and change management**

You can manage changes to web content items either by creating drafts, using workflows or adding items to projects.

**Related tasks:**
- Creating a publish action
  A publish action changes the status of an item from "draft" to "published".
- Creating an expire action
  An expire action changes the status of an item from "published" to "expired".
- Creating email actions
  An email action sends an email to a set of users or groups.
- Creating a custom action
  A custom action is used to run a custom workflow action based on a Java class you have previously created and added to your system.
- Creating a version action
  A version action causes a new version of an item to be created when run.
- Creating a scheduled move action
  A scheduled move action moves an item to the next workflow stage at a specified date and time.
- Creating a workflow stage
  A workflow stage is composed of a set of selected workflow actions.
- Creating a workflow
  You select the workflow stages to comprise a workflow in a workflow form.

**Item status**

There are three major status levels that an item can be in at any one time; **Draft**, **Published** or **Expired**. The current state of an item indicates where the item exists within a change management process, and where that item can be viewed and accessed.

**Status types**

Draft  This indicates that the item is currently being updated.

Pending published
  An item that uses a workflow can appear in a state of pending published. This indicates that the item has entered a workflow stage that includes a publish action but the action has yet to be processed.
Published
A published status indicates that an item is ready to be rendered in the live site.

Pending expired
An item that uses a workflow can appear in a state of pending expired. This indicates that the item has entered a workflow stage that includes an expire action but the action has yet to be processed.

Expired
An expired status indicates that an item is ready to be expired from the live site.

Changing status
The status of an item can only change in a linear fashion:

- Draft to Published.
- Published to Expired.
- Expired to Published.
- Published to Draft.

You cannot change an item’s status from Expired to Draft.

The process of publishing and expiring items

When the status of an item changes to published or expired, this does not mean that the item will have been added or removed from the rendered site. A status of published or expired means that the process of publishing or expiring an item has begun.

The actual time a published item will appear on a website, or the time an expired item is removed from a website, also depends on:

- how long it takes to syndicate updates to the delivery server
- how long it takes for the current cache to expire

Draft items:

Creating a draft of an item allows you to work on changes to that item without changing the published version of the item. Draft items can either be stand-alone items, or form part of a workflow. Once the changes are completed, you can choose to either publish the item, or discard the changes by cancelling the draft. You can create multiple drafts of a single item.

Note: Draft items are only displayed in the Authoring Portlet and are not rendered within the published website.

Working with draft items not using a workflow

You can directly create a draft of any non-workflowed items.

Creating a draft item

To create a new draft item:

- open a non-workflowed published item in edit mode and click Save as Draft. This will create a draft copy of the published item without removing the published item from the live site.
• open a non-workflowed published item in read mode and click **Create Draft**. This will create a draft copy of the published item without removing the published item from the live site.

• open a non-workflowed published item in read mode and click **Change to Draft**. This will change the state of the item from "published" to "draft" and the item will no longer be visible on the live site. You cannot change an item to draft if it is being referenced by any other items.

**Saving a draft item**

Once the draft has been created:

• to save the draft item, click **Save, Save and Close** or **Save and Read**.

• to publish the draft item, click **Save and publish**.

Once saved, your draft item will be displayed alongside other items in your item views but will be displayed with a status of draft. If you have created a draft of a previously published item, both the draft and published versions of your item will appear in your item views. You can edit and save drafts multiple times before publishing your changes.

**Publishing a draft item**

Any changes made to a draft item will not appear in the live site until you publish the draft item. When your draft is ready to publish you can either:

• select an item in a view and click **Publish item**.

• open an item in read-mode and click **Publish item**.

• open an item in edit-mode and click **Save and publish**.

**Canceling a draft**

Canceling a draft is essentially the same as deleting a draft as all the changes made to the item are discarded. To cancel a draft open a draft item and click **More Actions > Cancel draft**. You need editor access or higher to cancel a draft of items not using a workflow.

**Working with draft items in a workflow**

Within a workflow, items have a status of draft until they enter a workflow stage where a publish action is executed. Draft items in a workflow are displayed alongside other items in your item views but are displayed with a status of draft. If you have created a draft of a previously published item, both the draft and published versions of your item will appear in your item views. You can edit and save drafts multiple times before submitting your draft to the next stage in the workflow.

You use the following buttons to work with items during a workflow.

Table 25. Approving and declining access controls.

**Note:** This table assumes contributor access or higher to the library.

<table>
<thead>
<tr>
<th>Actions</th>
<th>Function</th>
<th>Item Access</th>
<th>Role access to library resources</th>
</tr>
</thead>
<tbody>
<tr>
<td>More Actions &gt; Approve</td>
<td>When joint approval is set, button is used to approve an item in the All Items and My Items views.</td>
<td>Approver or administrator.</td>
<td>Contributor access or higher to the item type.</td>
</tr>
</tbody>
</table>
Table 25. Approving and declining access controls (continued).

Note: This table assumes contributor access or higher to the library.

<table>
<thead>
<tr>
<th>Actions</th>
<th>Function</th>
<th>Item Access</th>
<th>Role access to library resources</th>
</tr>
</thead>
<tbody>
<tr>
<td>More Actions &gt; Cancel draft</td>
<td>This removes a draft item from a workflow altogether. This displays only after an item has been published and a new draft has been created.</td>
<td>Manager access or higher.</td>
<td>Contributor access or higher to the item type.</td>
</tr>
<tr>
<td>More Actions &gt; Decline</td>
<td>Used when an item is rejected during a workflow. Executes any actions defined in the reject stage, and then sends the item back to the first stage of a workflow.</td>
<td>Approver access.</td>
<td>Contributor access or higher to the item type.</td>
</tr>
<tr>
<td>More Actions &gt; Next Stage</td>
<td>Used to approve an item and send it to the next stage in a workflow.</td>
<td>Approver access.</td>
<td>Contributor access or higher to the item type.</td>
</tr>
<tr>
<td>More Actions &gt; Previous Stage</td>
<td>The previous stage button returns an item to the stage previous to the current stage. When an item is moved to the previous stage, the entry workflow actions on the previous stage are executed, but the exit workflow actions on the current stage are not.</td>
<td>Manager access or higher. Approver access on workflow stages where approvers have been granted access to the previous stage button.</td>
<td>Contributor access or higher to the item type.</td>
</tr>
<tr>
<td>More Actions &gt; Process now</td>
<td>Manually processes an item if its status is pending. All actions in the stage are processed.</td>
<td>Administrator access</td>
<td>Not required.</td>
</tr>
<tr>
<td>Submit for review</td>
<td>Used to move a draft item into a draft stage for approval.</td>
<td>Approver access.</td>
<td>Contributor access or higher to the item type.</td>
</tr>
<tr>
<td>Publish</td>
<td>Used to move a draft item into a stage with a publish action.</td>
<td>Approver access.</td>
<td>Contributor access or higher to the item type.</td>
</tr>
</tbody>
</table>

Disabling multiple drafts:

When creating a workflow you can choose to disable the creation of multiple drafts for any item using that workflow. This allows you to assign a workflow to items where you only want to manage one change at a time.
Working with draft items in a project

You use draft items within a project in the same way you use drafts outside of a project, regardless of whether the draft item is participating in a workflow, or not participating in a workflow. The only difference being that when the draft is ready to be published, it will remain in a pending state until all items in the project are ready to be published. Draft items in a project are displayed alongside other items in your item views, but are displayed with a status of draft. If you have created a draft of a previously published item, both the draft and published versions of your item will appear in your item views.

Managing drafts

There are various ways you can manage your draft items:

- If viewing a published item with only one draft item, you can view the draft item by clicking More Actions > Go to draft.
- When viewing a draft item, you can open the published version of an item by clicking More Actions > Go to published.
- If a published item has more than one draft you click More Actions > Manage Drafts to view a list of drafts for the current item. All drafts are displayed, but you can only read, edit and cancel drafts that you have sufficient access to.

Published items:

Items are not visible on a website until they are published. Once published, an item can be expired, or returned to a draft state.

Working with published items not using a workflow

To create a new draft item:

- open a non-workflowed published item in edit mode and click Save as Draft. This will create a draft copy of the published item without removing the published item from the live site.
- open a non-workflowed published item in read mode and click Change to Draft. This will change the state of the item from "published" to "draft" and the item will no longer be visible on the live site.

Working with published items using a workflow

You use the following access control options to work with items during a workflow.

<table>
<thead>
<tr>
<th>Actions</th>
<th>Function</th>
<th>Item Access</th>
<th>Role access to library resources</th>
</tr>
</thead>
<tbody>
<tr>
<td>Approve</td>
<td>When joint approval is set, use this option to approve an item in the All Items and My Items views.</td>
<td>Approver or administrator.</td>
<td>Contributor access or higher to the item type.</td>
</tr>
</tbody>
</table>
Table 26. Published items functions and access controls (continued).

**Note:** This table assumes contributor access or higher has been set on the library.

<table>
<thead>
<tr>
<th>Actions</th>
<th>Function</th>
<th>Item Access</th>
<th>Role access to library resources</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create draft</td>
<td>Creates a draft copy of a published item. The published item is locked on the rendered site, while a copy of the item is sent through a workflow. Once the draft item has been approved for publishing, it replaces the published item.</td>
<td>Manager access or higher, or approver access.</td>
<td>Editor access or higher to the item type.</td>
</tr>
<tr>
<td>Next Stage</td>
<td>Used to approve an item and send it to the next stage in a workflow.</td>
<td>Approver access.</td>
<td>Contributor access or higher to the item type.</td>
</tr>
</tbody>
</table>
| Previous Stage| The previous stage option returns an item to the stage previous to the current stage.  
• If the current stage contains a publish action, the item status will revert to draft when returned to the previous stage. The published version is removed and is no longer visible on the live site.  
• When an item is moved to the previous stage, the entry workflow actions on the previous stage are executed, but the exit workflow actions on the current stage are not.  
**Note:** The previous stage option is not enabled:  
• on published items that have children.  
• on published items where a draft already exists. | Manager access or higher, or on workflow stages that have been configured to enable approvers access to the previous stage option. | Contributor access or higher to the item type. |
Table 26. Published items functions and access controls (continued).

**Note:** This table assumes contributor access or higher has been set on the library.

<table>
<thead>
<tr>
<th>Actions</th>
<th>Function</th>
<th>Item Access</th>
<th>Role access to library resources</th>
</tr>
</thead>
<tbody>
<tr>
<td>Process now</td>
<td>Manually processes an item if its status is pending. All actions in the stage are processed.</td>
<td>Administrator access</td>
<td>Not required.</td>
</tr>
</tbody>
</table>
Table 26. Published items functions and access controls (continued).

Note: This table assumes contributor access or higher has been set on the library.

<table>
<thead>
<tr>
<th>Actions</th>
<th>Function</th>
<th>Item Access</th>
<th>Role access to library resources</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Restart workflow</strong></td>
<td>Sends an item back to the first stage of a Workflow without activating the reject stage. The item no longer appears as published and is removed from the rendered site. <strong>Note:</strong> When you restart a workflow, any actions set to run on entering the first stage will not be executed. <strong>Note:</strong> When you restart a workflow with multiple stages, a draft item is created in the first stage. In addition to this, the item will also appear in the deleted items view so that the last published version of the item can be restored. Once the draft is republished, the version in the deleted items view is removed. <strong>Note:</strong> Items with references cannot be restarted in the workflow. References includes link components, link elements, and embedded links in rich text or HTML fields. You will need to remove these references before you can restart the workflow of the item. Use the view references dialog to see if references exist. This restriction is to prevent broken links appearing in your site.</td>
<td>Manager access or higher, or approver access.</td>
<td>Editor access or higher to the item type.</td>
</tr>
</tbody>
</table>

Expired items:
Expired items are items that were once published but have since been removed from the live website. Only items that use a workflow can be expired.

You use the following access control options to work with expired items within a workflow.

Table 27. Expired items functions and access controls.

Note: This table assumes contributor access or higher to the library.

<table>
<thead>
<tr>
<th>Actions</th>
<th>Function</th>
<th>Item Access</th>
<th>Role access to library resources</th>
</tr>
</thead>
<tbody>
<tr>
<td>Approve</td>
<td>When joint approval is set, use this option to approve an item in the All Items and My Items views.</td>
<td>Approver or administrator.</td>
<td>Contributor access or higher to the item type.</td>
</tr>
<tr>
<td>Create draft</td>
<td>Creates a draft copy of an expired item.</td>
<td>Manager access or higher, or approver access.</td>
<td>Editor access or higher to the item type.</td>
</tr>
<tr>
<td>Next Stage</td>
<td>Used to approve an item and send it to the next stage in a workflow. For expired items, additional workflow stages may include further actions that will be run after an item has been expired.</td>
<td>Approver access.</td>
<td>Contributor access or higher to the item type.</td>
</tr>
</tbody>
</table>
| Previous Stage | The previous stage option returns an item to the stage previous to the current stage.  
  • If the current stage contains an expire action, the item status will revert to published when returned to the previous stage and will be visible on the live site.  
  • When an item is moved to the previous stage, the entry workflow actions on the previous stage are executed, but the exit workflow actions on the current stage are not. | Manager access or higher, or on workflow stages that have been configured to enable approvers access to the previous stage option. | Contributor access or higher to the item type. |
Table 27. Expired items functions and access controls (continued).

Note: This table assumes contributor access or higher to the library.

<table>
<thead>
<tr>
<th>Actions</th>
<th>Function</th>
<th>Item Access</th>
<th>Role access to library resources</th>
</tr>
</thead>
<tbody>
<tr>
<td>Process now</td>
<td>Manually expires an item if its status is pending expired. All actions in the stage are processed.</td>
<td>Administrator access</td>
<td>Not required.</td>
</tr>
</tbody>
</table>
Table 27. Expired items functions and access controls (continued).

Note: This table assumes contributor access or higher to the library.

<table>
<thead>
<tr>
<th>Actions</th>
<th>Function</th>
<th>Item Access</th>
<th>Role access to library resources</th>
</tr>
</thead>
</table>
| Restart workflow  | Sends an item back to the first stage of a Workflow without activating the reject stage. The item no longer appears as published and is removed from the rendered site.  
  Note: When you restart a workflow, any actions set to run on entering the first stage will not be executed.  
  Note: When you restart a workflow with multiple stages, a draft item is created in the first stage. In addition to this, the item will also appear in the deleted items view so that the last published version of the item can be restored. Once the draft is republished, the version in the deleted items view is removed.  
  Note: Items with references cannot be restarted in the workflow. References includes link components, link elements, and embedded links in rich text or HTML fields. You will need to remove these references before you can restart the workflow of the item. Use the view references dialog to see if references exist. This restriction is to prevent broken links appearing in your site. | Manager access or higher, or approver access. | Editor access or higher to the item type. |
Workflow stages and actions

You use workflows to control the access to, verification and eventual approval of items. Only if an item is approved at all stages up to a published stage can it be viewed on your website.

A workflow must have at least one stage, but typically has more, and it always flows in a linear pattern. You can use a workflow to:

- review the accuracy of content.
- review content for any legal implications.
- review content to ensure it meets accessibility guidelines.
- ensure that no malicious code such as cross scripting attacks have been added to content.

A reject stage may be specified, which is a stage that is executed when a document is declined, before moving it to the first stage of the workflow. If the item is rejected at any stage, someone with editor access needs to correct or amend the item and resubmit it into the selected workflow (for approval). All items that are rejected (regardless of the stage they are at in the approval process) are sent back to the first (creation) stage of the workflow.

You can also specify that a comment must be entered on every move a document makes in the workflow or only on specific stages. This comment is added to the document’s history section.

Workflow stages:

Workflow stages are the building blocks of a workflow. You need to create at least one stage before you can create a workflow.

Workflow stages

Stages determine:

- What actions to execute when entering or exiting a workflow stage
- The access levels of users or groups within that stage.

In most cases, actions are run when entering a stage. For example, you add a scheduled move action to run on entering a stage so that it is enabled as soon as an item enters that stage. However, if you set a scheduled move action to run on leaving a stage, it will never run. The most common type of actions to run on leaving a stage are email actions, when you want to notify users that an item has exited a workflow stage, or custom workflow actions that have been designed to run a task when an item leaves a stage.

Note: Some actions need to be run in a specific order. For example:

- A scheduled move action must always be the final action in a workflow stage, because any actions scheduled after a scheduled move action will not be run.
- You cannot run a version action before a publish action because you cannot save versions of draft items.
- If using a custom action, you may want to run the custom action before executing an email action so that the draft content item is in a state ready to be reviewed by an approver.

Note: The access settings that are defined in the properties section of the workflow stage form are the security settings applied to items during a workflow,
not the Security section of a workflow stage. The Security section only defines who has access to the workflow stage item itself.

Reject stages

In addition to the workflow stages that make up a workflow, there are also workflow stages that are used as part of rejecting an item. When an item is rejected, a reject stage can be triggered that executes pre-defined actions. Once the actions have been executed, the item is returned to the first stage of the workflow.

Workflow actions:

Each workflow stage contains sets of actions; those that are executing when entering the stage and those executed when exiting the stage. The exit actions are restricted to non-scheduled actions, since they must be executed immediately.

The following table describes the actions you can choose for workflow stages.

<table>
<thead>
<tr>
<th>Action</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Publish</td>
<td>Changes an item's Status from Draft to Published. This means the item is available on the rendered site. An item will only be published once it has entered a workflow stage containing a publish action, and when the selected published date and time has been reached.</td>
</tr>
<tr>
<td>Expire</td>
<td>Changes an item's Status from Published to Expired. This means the document is no longer available on the site. An item will only be expired once it has entered a workflow stage containing an expire action, and when the selected expired date and time has been reached.</td>
</tr>
<tr>
<td>Email</td>
<td>This sends emails when executed. You can create new email actions and specify who the recipients will be. You can select to email approvers, authors and owners. You can also create a list of other users or groups to email. A link to the Item to be reviewed is included in the email.</td>
</tr>
<tr>
<td>Scheduled Move</td>
<td>Performs a scheduled move to the next stage on a specified date. A list-box will allow you to select one of four date types that are entered on each individual document, or you specify a static date.</td>
</tr>
<tr>
<td>Version</td>
<td>This will create a version of an item when executed.</td>
</tr>
<tr>
<td>Custom</td>
<td>You can also create custom workflow actions by creating a custom workflow plug-in. These can be used and scheduled within a workflow like other workflow actions.</td>
</tr>
</tbody>
</table>

Accessing Items during a workflow:

If an item is participating in a workflow, the creator is given manager access to the item only in the first workflow stage. As the item progresses through a workflow, the item access is determined by the combined workflow and system defined access levels.
Table 29. Access levels

<table>
<thead>
<tr>
<th>Access level</th>
<th>1st workflow stage</th>
<th>Additional workflow stages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read</td>
<td>• System defined</td>
<td>• System defined</td>
</tr>
<tr>
<td></td>
<td>• Workflow defined</td>
<td>• Workflow defined</td>
</tr>
<tr>
<td>Edit</td>
<td>• System defined</td>
<td>• System defined</td>
</tr>
<tr>
<td></td>
<td>• Workflow defined</td>
<td>• Workflow defined</td>
</tr>
<tr>
<td>Delete</td>
<td>• User who created item</td>
<td>• System defined</td>
</tr>
<tr>
<td></td>
<td>• System defined</td>
<td>• Workflow defined</td>
</tr>
<tr>
<td></td>
<td>• Workflow defined</td>
<td></td>
</tr>
<tr>
<td>Approve</td>
<td>• Workflow defined</td>
<td>• Workflow defined</td>
</tr>
</tbody>
</table>

Joint approval:

Joint approval is used in cases where approval from multiple users is required before moving the document to the next stage.

- You specify which stages you want to be jointly approved.
- If joint approval is active, then all the approvers specified for this stage must approve the document.
- The exception is an administrator, who can force a document to the next stage.
- If a group is specified as an approver, only one user per group is required to approve the document.
- If that user is a member of more than the Group that has been given Approve access, then all the groups will be considered as having approved the document.

Workflow example:

This example describes the steps required to create a four stage Workflow.

Actions

The actions required for this workflow are:

Table 30. Actions

<table>
<thead>
<tr>
<th>Action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Publish</td>
<td>Makes a document visible as a published document on the website.</td>
</tr>
<tr>
<td>Scheduled Move</td>
<td>This scheduled move action detects when a document has passed its expire date and moves the document into the Expired stage.</td>
</tr>
<tr>
<td>Expire</td>
<td>Stops a document from being visible on the web site.</td>
</tr>
<tr>
<td>Email</td>
<td>Sends emails to selected users. In this example, you select &quot;Email Stage Approvers&quot;.</td>
</tr>
</tbody>
</table>
**Stages**

The following stages make up the Workflow:

*Table 31. Stages*

<table>
<thead>
<tr>
<th>Workflow Stage</th>
<th>Actions on Entering</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stage 1, Draft</td>
<td></td>
<td>All content will begin the Workflow in this Stage. All content authors will be authorized to use the Draft stage.</td>
</tr>
<tr>
<td>Stage 2, Review</td>
<td>Email</td>
<td>Content will require a review stage before being published. A small group of people will be authorized to approve content in this stage and move it to the Publish stage. The email action will send an email to each approver.</td>
</tr>
<tr>
<td>Stage 3, Publish</td>
<td>Publish, Scheduled Move</td>
<td>This is the stage where content from the workflow is published. The Publish action is triggered when the Publish Date is reached. At this point the content will be visible on the website. The Scheduled Move action will also be triggered when the Expire Date is reached.</td>
</tr>
<tr>
<td>Stage 4, Expired</td>
<td>Expire</td>
<td>The Scheduled Move action will move the content into this stage, where the Expire action will stop the document from being visible on the website.</td>
</tr>
</tbody>
</table>

**Progressing through the workflow**

*Table 32. Progressing through the workflow*

<table>
<thead>
<tr>
<th>Process</th>
<th>Stage</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>An item is first created and saved:</td>
<td>Stage 1, Draft</td>
<td>Draft</td>
</tr>
<tr>
<td>Once the Item Creator is ready, they move it to the next stage.</td>
<td>Stage 2, Review</td>
<td>Draft</td>
</tr>
</tbody>
</table>
### Table 32. Progressing through the workflow (continued)

<table>
<thead>
<tr>
<th>Process</th>
<th>Stage</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>An Approver would then access the document and review it. If Approved, the item would be moved to the next stage.</td>
<td>Stage 3, Publish</td>
<td>Draft pending Published</td>
</tr>
</tbody>
</table>

The Status is Draft pending Published. This is because the time specified in the Publish Date field of the item has not yet been reached.

Once the Publish Date is reached, the item’s Status changes to Published. The item is still in Stage 3, Publish, but the Status has changed.

Once the Expire Date is reached, the document is moved to the final stage. As both the Scheduled Move and Expire actions are using the Expired date in this example, the item’s Status immediately changes to Expired.

If the Scheduled Moved Action used a General Date prior to the Expire Date, then the item’s Status would remain as Published pending Expire until the Expire Date was reached even though it had entered Stage 4, Expire.

---

**Using workflows and access to workflow items:**

Users do not require access to a workflow’s actions or stages to participate in a workflow. Actions are performed using system access and are not determined by the access of the user who approved/rejected the item.

**Adding workflow to managed pages**

The default workflow used for managed pages has only a draft and approved state. However, if the default workflow is not sufficient, you can define custom workflows and use them with managed pages.

Before you perform this task, ensure that you add the Portal Site library to the list of libraries that you can edit with the authoring portlet. In the authoring portlet, click Preferences, and then click Configure or Edit Shared Settings.
To use a custom workflow with a managed page, you must create a page template and specify the custom workflow on the template. When you create a managed page based on the template, the custom workflow is used for the page.

1. In the authoring portlet, create any workflow stages that you require and then create the custom workflow.

2. Create a project to use when creating the page template.

   Note: Because you can set a workflow only on draft items, you must create the template as a draft in the context of a project. After you add your custom workflow to the draft, publish the draft to make the page template available.

3. In the project menu, select the new project as the current context.

4. Click Administration > Portal User Interface > Page Templates.

5. Click New Page, and create the page template.

6. Click Save Draft.

7. Add the workflow to the template.

   a. In the site toolbar, click Edit in the page properties section.
   b. In the Page Properties window, click Security.
   c. In the Workflow section, click Select in the Workflow field.
   d. Select the custom workflow, and save your changes.

8. Approve and publish the project to make the page template available for use.

   a. In the project menu, click Manage in the project actions section.
   b. Select the page template in the list of project items, and click More > Approve.
   c. Click Publish Project.

After completing this task, you can select the new page template when creating a page in a project, and the custom workflow is automatically used.

**Best practices for projects**

Use these tips and guidelines to develop and publish projects more effectively.

**Use separate projects for different parts of the site structure**

Coordinating projects is an important part of ensuring that changes to your website are published as expected. For example, you might have two authors working on the same part of the site structure but with each author editing pages in a different project. Because the authors are working in different projects, they cannot see draft changes that the other author makes until the projects are published. If both authors insert a new page between the same two published pages, the resulting page order might not be what was intended.

To prevent these situations, avoid editing the same part of the site structure with multiple projects at the same time. Instead, when using multiple projects at the same time, ensure that you use each project for a different part of the site structure.

**Coordinate changes to a project across users**

When multiple users are working in the same project, one user might not be aware of the changes that another user makes. If you change the state of the project, such as deleting it or publishing it, any outstanding updates from other users are not saved. To ensure that you do not inadvertently discard changes from other users, coordinate with users of the project before changing the state of the project.
Coordinate unique names across projects

When multiple projects are used for the same site, similar unique names might be introduced in different projects. For example, you might have two authors working on the same site but with each author editing in a different project. Because the authors are working in different projects, they cannot see draft unique names that the other author uses until the projects are published. If both authors insert a similar unique name, the result might not be what was intended.

To prevent these situations, avoid introducing new unique names with multiple projects at the same time. Instead, when using multiple projects at the same time, ensure that you use different unique names within each project.

Managed pages

Managed pages streamline site management in your portal by simplifying how you create pages and add content. Because page information and content are stored in web content libraries, you can more easily coordinate and publish changes with syndication.

Managed pages are portal pages that are stored in IBM Web Content Manager. By managing portal pages from within Web Content Manager, you can apply features like workflow, version control, and syndication to portal pages.

When you perform a new installation of IBM WebSphere Portal, managed pages are enabled by default. However, if you migrate from an earlier version or previously disabled the feature, ensure that managed pages are enabled, as described in Configuring managed pages.

Related tasks:

- Configuring managed pages on page 100

When you perform a new installation of IBM WebSphere Portal 8.0, managed pages are enabled by default. However, you can also manually disable and enable the feature as needed.

WebSphere Portal artifacts affected by managed pages

You can manage all types of portal pages, including pages that use the Portal 8.0 theme, dynamic pages, and static pages.

To take advantage of all the features available to managed pages in the user interface, you must use pages that use the WebSphere Portal 8.0 theme.

The following portal artifacts are affected by managed pages support:

- Portal pages, including page properties, metadata, and layout settings
- Access control that you grant to portal pages
- Public wires connecting portlets on the same page or on different pages
- Portlet preferences made in the Edit Shared Settings mode

The following portal artifacts are not affected by managed pages support and so are not available for versioning or syndication:

- Composite applications
- Explicitly derived pages
- Private pages
• Personalization rules that are defined in Personalization and not in Web Content Manager
• Tags and ratings
• Themes and skins
• URL mappings
• Artifacts that are stored in the WebDAV file store by portlets or iWidgets
• WSRP Producers

Related tasks:

Drafts and projects for changes to managed pages

You can create, update, and approve pages in a draft state, without affecting the published site. When working with managed pages, you can edit pages as drafts, manage drafts with approval workflow, and then syndicate to publish changes.

The page structure is replicated in the site structure of a web content library. Because of this integration, you can add web content directly to a page with the authoring portlet. Syndication then publishes the managed page, ensuring that changes to the page and to its associated web content are published at the same time. Update pages in a draft state first. Drafts are organized within projects in Web Content Manager. When you publish draft changes to the published site, a project coordinates the updates and ensures that all drafts are published at the same time.

The following changes to a page result in a draft that is not visible on the published site and is only visible within the project:
• Any action performed by using the toolbar, such as adding content to page, changing the page style, or changing the page layout.
• Changing the access control granted to a page.
• Changing the page properties.
• Creating a child page.
• Moving a page.
• Deleting a page.
• Changing the community or web content associations for a page.
• Creating or modifying portlet wires for communication between portlets on the same page or on different pages.
• Changing the portlet configuration with either the Configure mode or the Edit Shared Settings mode of the portlet.

Changes to page draft elements result in either a new draft or an update to an existing draft. A page draft consists of the following elements:
• The page item itself, including title, description, metadata, and properties
• The access control of the page
• The page layout
• Any public page wires that are connected to portlets on the same page or on another page
• Any portlet preferences that are defined in Edit Shared Settings mode
Related concepts:

“Projects overview” on page 133

Projects allow you to change a set of items and ensure that they are published together at the same time.

Scope of edits for managed pages

When you edit managed pages in a project, you use workflow for approving changes before publishing them until they are approved. If you edit ages without selecting a project, your changes are published immediately.

The recommended way to edit managed pages is to work within a project. When you are working in a project, any changes that you make to a page affect only the view within the project. After the project is approved and published, the changes are then available to all users.

Note: Changes to private pages and community pages are not part of a project. When you are working in a project, actions that personalize or create a private page are not visible.

When no project is selected, you are modifying the published site. Changes to public pages affect all users and are available immediately. While working on the published site, changes that other users are making within active projects are not visible.

Default workflow for managed pages in projects

To make the page changes available on the published site, you must submit the page for review and approval. The steps for the workflow depend on what is defined for the page. By default, a simple workflow is assigned to page or URL mapping drafts. The default workflow (Express Workflow) has only two states: draft and approved. For pages using this workflow, submitting the draft moves the page to the publish pending state. When all other changes in the project are published, the pending draft page is also published. After all items in the project reach the publish pending state, you can publish the project by clicking Publish in the project menu.

Note: If you use the authoring portlet to edit the access settings for a page that is part of a workflow, you cannot modify the approvers. To change the approvers for the page, you must set the approvers on the workflow stage that currently contains the page.

However, you can change the assigned workflow in the user interface to include additional review and approval stages on the project level in Web Content Manager. After the draft changes in the project are approved and published, the updated pages are visible to all users. For details on using a custom workflow, see Adding workflow to managed pages.
The default workflow used for managed pages has only a draft and approved state. However, if the default workflow is not sufficient, you can define custom workflows and use them with managed pages.

**Edit mode and administration pages**

Because the administration pages are not intended to be edited, those pages are excluded from the edit mode features provided by the site toolbar.

When you navigate to an administration page, the site toolbar automatically exits edit mode. The edit mode features are suppressed by the page parameter `theme.disable.edit.mode` on the Administration page label (unique name wps.Administration). Setting the parameter on the top-level administration page also causes the child pages to be affected.

You can set the `theme.disable.edit.mode` parameter on any page where you want to disable edit mode in the site toolbar. Edit the properties of the page, and add the `theme.disable.edit.mode` parameter with a value of true.

** Managed pages and site management**

Using managed pages is the preferred method for site management. The Resource Manager portlet is deprecated in Version 8.0 but can still be used for environments where managed pages are not available.

For information about site management with the Resource Manager portlet, see Version 7 product documentation: Managing your site.

**Managed pages and the authoring portlet**

When you create a managed page in the portal, a corresponding page item is created in the Portal Site library. The page items are represented as portal page site areas in the web content library.

By default, the display name of the portal page site area is based on the title of the portal page. Web Content Manager assigns a unique name in the library for each portal page site area. This unique name enables you to have pages with the same title organized in separate portal page site areas.

Although you can view the page items in the authoring portlet, the items are read-only. To change, move, or delete the page items, you must use the portal user interface. However, you can use the authoring portlet to add site areas and content items in the site structure for the managed page. In this case, the content that you add is automatically rendered by components, such as menus or navigators, and by web content viewers.

**Note:** If you delete a portal page, the portal page site area is deleted from the web content library. If the portal page site area contains any other site areas or content items, they are also deleted.
System content associations

System content associations are used to associate a portal page with its corresponding artifacts in IBM Web Content Manager. A system content association is an extension of the standard content association.

A standard content association maps a web content page or web content viewer to content in a web content library. For a system content association, there is an additional system flag that distinguishes the association from a content association. Like web content associations, system content associations point to objects in a web content library. However, the objects are associated with a managed page rather than content. System content associations are managed by the portal.

The system flag is a private, read-only flag and cannot be modified programmatically through the public API or the REST API. However, you can use these interfaces to query a content association to determine whether it is a system content association or standard content association.

Best practices for managed pages

Use these tips and guidelines to develop and deploy managed pages more effectively.

Create links in web content to portal pages

When you enable managed pages, you can create links to portal pages from within the authoring portlet in IBM Web Content Manager. You can create links to portal pages in two ways:

- By editing a content item in the rich text editor and inserting a link.
- By creating a link component.

To select the portal page, click Browse content in the Link field, and navigate to the page in the Portal Site library.

Referential integrity applies for links to portal pages. You cannot delete a portal page if a link pointing to that page exists. You can view or remove such link references in the following ways:

- Edit the page properties in the portal user interface and select View References.
- Select the page item in the authoring portlet in Web Content Manager and click More > View References.

When users click a link, the link is resolved according to the system content association for the portal page item in the Portal Site library. Based on the system content association, the appropriate portal page is displayed.

Important: You cannot change system content associations through typical operations with the user interface. However, it is possible to change system content associations through programmatic interfaces, like the XML configuration interface (xmlaccess command), or other low-level database operations. If a system content association is changed or corrupted through such a method, the link can no longer be resolved.

Use unique friendly URLs with managed pages

When creating managed pages, it is not possible to programatically enforce uniqueness of friendly URLs. Because of this behavior, it is possible to create
multiple pages that have the same friendly URL, which can produce unexpected results. To prevent potential confusion, ensure that all friendly URLs that you create are unique.

**Use transaction processing with the XML configuration interface**

Because managed pages are stored in the Portal Site library in Web Content Manager, each page has corresponding objects in the JCR database. You must be aware of this relation when you create, update, or delete managed pages with the XML configuration interface. If xmlaccess processing is interrupted, it can result in a mismatch between the page state and database state.

To ensure that page and database information for a managed page remain synchronized, use the transaction-level attribute of the request element in the XML file. For more information about using the transaction-level attribute, see XML configuration reference.

Example:

```
<request
    type="update"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="PortalConfig_8.0.0.xsd"
    transaction-level="resource">
```

**Troubleshooting managed pages**

When using managed pages, you might encounter problems that are related to projects, access rights, or other issues.

**User A cannot view project X**

Ensure that User A has the following access rights:

- User on the project. Specify this access by editing the project and adding User A to the User list in the Access section.
- User on the WCM_REST_SERVICE virtual resource. Specify this access in the portal by clicking Administration > Access > Resource Permissions and editing the WCM_REST_SERVICE resource.

**User A cannot modify a published or draft page**

Check the following issues as potential causes:

- If the page is part of a workflow, ensure that User A has Editor access to the current workflow stage.
- Is access control inheritance enabled for the portal page site area? The setting is enabled by default but can be disabled as needed. Verify the setting by editing the portal page site area and viewing the Access section of the properties to determine whether the Inheritance setting is selected.
  
  If access control inheritance is disabled, ensure that User A has Editor access to the portal page site area.

**Why does User A receive the message "You are customizing this page. Changes are only visible to you"?**

This message is generated because User A has Privileged User access to the page. This access level is the default access for a newly created user, and changes that the user makes are visible only to this user.
If you want the changes that User A makes to be visible to all users of that page, User A requires Editor access on the page.

**User A is in a project and receives the message "You are customizing this page. Changes are only visible to you" rather than creating a draft**

This message is generated because User A has Privileged User access to the page. This access level is the default access for a newly created user, and changes that the user makes are visible only to this user.

To create a draft in a project, User A requires either of the following access:
- Editor rights on the page.
- User access on the page and Approver access on the corresponding web content page item.

**User A cannot drag content from the site toolbar onto the page**

When using the site toolbar and attempting to add content from the Content category in the toolbar, the following access is required for User A:
- Editor rights on the page.
- User rights on the web content viewer portlet that User A wants to add.

**Portal pages are not synchronized with portal page site areas in Portal Site library**

Typically, the pages in the portal and their corresponding portal page site areas in the Portal Site library are automatically synchronized. However, in some cases, these artifacts can become unsynchronized. For example, this situation can occur when restoring data from a backup or from errors when creating the portal page site area after the portal page is created.

You can resynchronize the Portal Site library based on the current portal page structure that is stored in the portal database. When performing this synchronization, the portal database acts a master repository. Any portal page site areas in the Portal Site library that do not correspond with existing portal pages are removed from the Portal Site library. Any content site areas or content items within the affected portal page site areas are also removed.

To perform this resynchronization, run the create-page-nodes configuration task, as described in *Enabling managed pages*.

**Important:** This task also removes any draft pages that are not found in the Portal Site library.

**Known issues for managed pages**

You can review known issues for managed pages.

**Remote portlet entities are not aware of projects**

PortletEntity objects stored in the Release domain of the portal database are project-aware. Changes to such a portlet entity, such as setting and modifying
preferences, are reflected as a change limited to the active project. Changes to a portlet made in the active project are not visible on the published site until the changes are syndicated.

This ability to modify a portlet in a project does not apply to remote portlets that are produced with WSRP. As the remote system, the WSRP Producer is responsible for managing the portlet entities. However, because the WSRP Producer is not project-aware, the Producer cannot differentiate between the following changes:

- Changes made to the remote portlet entity directly on the published site.
- Changes made to the remote portlet entity when viewing and interacting with a project.

Because of this limitation, changes made in a project are displayed immediately on the published site through the remote portlet entity.

**Authoring portlet issues**

**Search results for page items in the authoring portlet**

When searching in the authoring portlet with the Titles or Descriptions filter, no results are returned for page items. To ensure that search results are returned, use the All attributes filter. Page items are included in the search results because the title and description attributes are stored in an XML document in the page item.

**Sorting order of managed pages**

When displayed in the authoring portlet, managed pages are listed according to the page IDs rather than the page titles. If you attempt to sort pages by title, this behavior causes the pages to display in an unexpected order. This sorting behavior also applies to any window or view, such as the Manage Project window, that is based on the authoring portlet.

**Changing the order of managed pages in the portal**

You can modify the ordering and hierarchy of managed pages in the portal only with the site toolbar in the portal interface. If you move the portal page site area in the authoring portlet, the page order in the portal is not affected.

**Personalization rules**

- Personalization rules that you create using the Personalization editor are not managed in Web Content Manager and so are not available for versioning or included in syndication. These rules must be published using the pznload command or by publishing with Personalization.
- Personalization rules are not aware of projects and the status of items in a project. Because of this characteristic, rules operate only on published content and do not include draft items.

**Limited support for derived pages**

Explicitly derived pages that are in the release domain can be managed by Web Content Manager. However, if you modify an explicitly derived page, that change does not generate drafts for all of the derived pages. If you want the change to occur for all of the derived pages, you must edit each derived page separately. For more information, see the documentation about derived pages.
**Portlet configuration settings**

**Configure mode**
If you change the configuration settings for a portlet in **Configure** mode, these changes are global and are not limited to the page. Because the changes are global, the changes cannot be managed in Web Content Manager and so cannot be syndicated to another server. To transfer these changes to another server, use the XML configuration interface (**xmlaccess** command).

**Edit Shared Settings mode**
If you change the configuration settings for a portlet in **Edit Shared Settings** mode, these changes are part of the page. Because the changes are part of the page, the changes are managed in Web Content Manager and are automatically syndicated to other servers.

**New Site wizard not supported**
The New Site wizard for creating virtual portals is not supported when managed pages are enabled.

**Automatic publishing and deleted items**
When you specify automatic publishing for a project, the project is published as soon as all the items in the project reach a state of "pending." Deletions do not go through an explicit approval stage and are available for publishing immediately. If your project consists of only deletions, automatic publishing of the project can occur prematurely.

To prevent this automatic publishing, you can complete the following steps:

- Ensure that the project contains new pages or changes to pages, which require approval before publishing.
- Set the project to use manual publishing.

**Syndication and versioning**

- You cannot syndicate the Portal Site library between servers that have different product offerings installed. For example, you cannot syndicate the Portal Site library from a server with WebSphere Portal Enable installed to a server with WebSphere Portal Express installed.
- The versioning feature of Web Content Manager also applies to managed pages and enables you to perform different tasks with page versions. Versioning tasks include saving, deleting, and restoring versions. The Page Properties window lists the versions of the page on the **Advanced** tab. However, if you create a page and syndicate the page for the first time, the version information is empty when you view the page properties on the delivery server. After subsequent syndication operations, the version information is listed.

**Related information:**
[Pages and page types: shared, derived, and hidden pages](#)

**Administering managed pages**
You can perform advanced administration tasks for managed pages, such as generating URLs for projects or working with projects by using scripts.
Project URL generation

You can redirect request processing to a specific project by generating URLs with the ProjectIdentificationService API, the REST API, or the Enabler API. Request processing operates either completely within the scope of a project or completely outside the scope of a project. You cannot switch projects during request processing.

When a request originates from within a project, the request URL contains a project identifier for that project. The project information is included only in the URL and is not bound to the session. The project identifier can be an object ID (OID), as used by the portal, or a universally unique identifier (UUID), as used by Web Content Manager. To direct request processing to a specific project, you must generate a URL for the project and then render the URL.

Java API

To generate URLs that target a project using the Java API in the portal, you can use the ProjectIdentificationService API with the StateManagerService API:

- The ProjectIdentificationService provides methods to create a ServerContext object, based on the identifier of the target project and the current ServerContext object.
- The project-specific ServerContext object can then be used to retrieve a URLFactory object from the state manager service. All URLs generated with this factory contain the project ID.

This example constructs a portal URL to the current navigational state for a new project:

```java
// construct a server context for the project
final ServerContext projectCtx = projectService.createServerContext(
    projectId, stateService.getServerContext());

// access the URL factory to create a URL
final URLFactory urlFct = stateService.getURLFactory(projectCtx);

// construct a URL to the current state
final EngineURL url = urlFct.newURL(StatesConstants.SMART_COPY);
url.writeDispose(out);

// done with URL generation
urlFct.dispose();
```

This example constructs a portal URL to a URI in a specific project:

```java
// construct a server context for the project
final PocServerContext projectCtx = projectService.createServerContext(
    projectId, pocService.getServerContext());

// access the URL factory to create a URL
final DisposablePocURLFactory urlFct = pocService.getPURLFactory(projectCtx);

// construct a URL to the current state
final PocURL url = urlFct.newURL(PocURLFactory.LATE_BINDING);
url.setMode(StatesConstants.VALUE_DOWNLOAD);
url.setURI(new URI("test:abc"));

// serialize
url.writeDispose(out);

// done with URL generation
urlFct.dispose();
```
REST API

If your application uses the Representational State Transfer (REST) architecture, you can use the remote APIs provided with the portal to construct project-specific URLs.

Enabler API

If you are using the Enabler API, you can pass the project identifier to the URL generation API as the parameter project in the parameters object. The project identifier can be either a serialized OID or a UUID.

```javascript
// get the current nav state
var state = com.ibm.mashups.enabler.model.state;
var navState = state.NavigationStateModelFactory.getNavigationStateModel();

// get the URL generator
var urlGen = state.UrlGeneratorFactory.getURLGenerator();
urlGen.getURL(navState, function(url) { alert(url); }, { "project": "UUID-of-Project" } );
```

Access control for managed pages

Access control for managed pages provides more capabilities than access control for standard portal pages. In addition to the access control features available for pages through portal administration, you can also apply IBM Web Content Manager features, like workflow and syndication, to access control.

When you create a managed page in the portal, a corresponding page item is created in a web content library. You can view and change access control settings for a managed page in two ways:

- By navigating to the page and using the site toolbar
- By opening the corresponding page item in the web content authoring portlet

Regardless of the method you use to change an access control setting, the corresponding element is automatically updated. This synchronization ensures that effective permissions are coordinated between the portal page and the web content page item.

Special considerations

Because managed pages integrate features from portal pages and Web Content Manager, there are special considerations that apply with access control for managed pages.

Unified set of applicable roles with different effective capabilities

With managed pages, portal pages and Web Content Manager are aware of the same roles; however, some roles are effectively ignored in Web Content Manager. For example, the roles of Privileged User and Markup Editor are used with portal pages to support features such as personalizing a page. In Web Content Manager, these roles have no effect on access control.

When you perform a web content action on a managed page, like previewing, publishing, or syndicating the page, Web Content Manager accounts for the portal roles. This awareness ensures that pages retain their appropriate permissions from the roles. For details on the portal roles, see Roles.

Virtual groups in Web Content Manager (authors, owners, creators)

In Web Content Manager you can grant access to virtual groups (authors, owners, creators) through the web content authoring portlet or as part of a
workflow stage. Portal pages do not provide an equivalent mechanism. When you grant permissions on a page item to users or groups with the virtual groups, direct role mappings are assigned on the portal page. These role mappings ensure that equal permissions are applied.

The owner virtual group, however, is limited to a single owner for page items in Web Content Manager. The owner of the portal page is automatically synchronized with the owner of the page item. This owner has the same set of allowed actions as the Manager role, as described in the Ownership section of Roles.

**Important**: If you are using author or creator groups for access control management in Web Content Manager, use only the authoring portlet to perform access control tasks. Do not use the site toolbar in the portal interface to revoke permissions, because doing so can lead to a potentially complex assignment of permissions.

**Traversal support for portal pages**

With portal pages, traversal support provides implicit permissions that enable users to navigate through a page hierarchy. For example, a user might have permission to access a child page but might not have permission to access the parent page. Because of traversal support, the user is permitted to navigate to the child page. See Roles for details on traversal support.

However, traversal support is not provided for web content items. Content authors that use the authoring portlet must be assigned the User role on all pages higher than the child page to navigate to editable content. Without this access permission, the editable content is not visible in the authoring portlet, even though the author can access the page. Typical page administration tasks can still be performed from the page.

**Permissions granted through virtual resources**

With traditional portal pages, you can grant permissions on the virtual resources PORTAL and CONTENT_NODES that inherit permissions to the complete page hierarchy. This inheritance is described in Resources. You can also specify a similar inheritance for web content libraries that inherit from the root node.

Because permissions for managed pages are synchronized between portal pages and page items in Web Content Manager, such inheritance is problematic. This inheritance can result in different effective permissions on portal pages and content items. Although you can manage permissions correctly either through the page or the authoring portlet, the preferred approach is through the page. If you grant permissions to the entire page hierarchy with inheritance, grant this permission on the root resource for the page hierarchy (wps.content.root page). As the permission on this page node is synchronized to the corresponding page item in Web Content Manager, the effective permissions are automatically synchronized throughout the hierarchy.

**Access control permissions managed by workflows**

When working with managed pages, you can apply access control to page items through workflow stages and actions, as described in Workflow and change management. In addition to permissions from the workflow, you can also modify permissions on the page with the site toolbar. Changes that you make with the site toolbar override the access permissions in effect.
with the current workflow stage. When the next workflow stage is entered, changes from the site toolbar are reset and the permissions specified by the workflow stage take effect.

**External security support**

You cannot use externalized roles or role mappings with managed pages. Pages cannot be externalized while being edited in a project. Similarly, externalized resources cannot be added to a project.

**Required permissions**

The following permissions are required for typical actions with managed pages.

<table>
<thead>
<tr>
<th>Action</th>
<th>Required permission</th>
</tr>
</thead>
<tbody>
<tr>
<td>Access a project view in the site toolbar</td>
<td>User on the WCM_REST_SERVICE virtual resource</td>
</tr>
</tbody>
</table>
| View a project in the site toolbar | • User on the WCM_REST_SERVICE virtual resource, in addition to the permissions that are required to view a specific project  
• User on the selected project |
| Create a project | • Contributor on the Portal Site library  
• User on the selected project |
| Create a draft of a published page by editing the page in a project | • Editor on the page  
• User on the selected project |
| Create a draft of a published page with the Create Draft action in the site toolbar. | • User on the page and Approver on the corresponding web content page item. For details, see "Approver role for creating draft pages" on page 166.  
• User on the selected project |
| Create a draft child page under a parent page in a project | • Contributor or Editor on the parent page  
• User on the selected project |
Table 33. Required permissions for typical actions with managed pages (continued)

<table>
<thead>
<tr>
<th>Action</th>
<th>Required permission</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preview a project</td>
<td>• Can Run As User on the USERS virtual resource</td>
</tr>
<tr>
<td></td>
<td>• The user that is impersonated requires at least User access to the current portal page. If an anonymous user does not have access to the page, the As Unauthenticated User preview option is not available in the site toolbar. In addition, if you select the As User preview option, you cannot select users that do not have access to the page.</td>
</tr>
<tr>
<td></td>
<td>• User on the selected project</td>
</tr>
<tr>
<td></td>
<td>By default only users and unauthenticated users that have explicit access to the project can preview the project. You can globally assign access for users or unauthenticated users to view all items in all libraries and projects in a specific virtual portal or the default virtual portal. To assign these rights, use the Set root access setting in the library administration portlet (Administration &gt; Portal Content &gt; Web Content Libraries).</td>
</tr>
</tbody>
</table>

Create web content by adding web content viewer to a page. The viewer is configured to create and render content from a web content library.

|                                                           | • Editor on the page                                                               |
|                                                           | • User on the viewer portlet                                                       |
|                                                           | • No library permissions are enforced.                                              |

Perform inline editing of content on a page

|                                                           | • Editor on the page                                                               |
|                                                           | • Appropriate permissions on the library that contains the content                 |

For the required permissions for portal pages and web content items, see Access permissions for portal pages and User roles and access for web content items.

The default set of access control permissions for anonymous users and for members of the All Authenticated Users group are described in Initial Access Control Settings. With managed pages, the following default permissions exist:

• Anonymous users can view projects and have User access to the Portal Site library.

• Members of the All Authenticated Users group can create new projects and have Editor access to the Portal Site library. This access ensures that users can perform inline editing tasks. You can restrict access as needed with the library administration portlet.

To modify a portal page or page item, you require only those permissions that are needed to perform the action from the user interface or programming API. You do not also require permissions for the underlying synchronization actions that take place automatically. These automatic updates are performed with system privileges.

For example, you might add a portlet to a page by using the site toolbar. In this case, you require sufficient permissions on the page that you are editing and on
Adding page content to a portal page

When you add a portlet to a portal page, you typically add content to that portlet. However, you may also need to modify that content. You can modify content in the following ways:

- Use a workflow to enable business users to create draft pages
- Use the Portal Site library to store managed pages

**Approver role for creating draft pages**

With managed pages, you can use a workflow to enable business users to create draft versions of pages that they are normally not allowed to edit. By using a workflow in this way, you accomplish two things:

- You provide business users with the ability to modify pages.
- You can still ensure that the drafts are reviewed and approved by technical users before the changes are published to the external site.

Typically a user with User access to a page has permission only to view the page. But if the user also has Approver access to the corresponding page item in the Portal Site library, the user can create page drafts. When a user has this access, the user can navigate to the portal page and use the site toolbar to create a draft.

To enable business users to create draft pages, complete the following steps:

1. In the Portal Site library, assign a workflow to the page items that correspond to the portal pages that you want users to modify. By default, page items are not managed in a workflow.
2. Edit the publish stage of the workflow, and update the access control properties to add the users to the Approver role.
3. Edit the initial draft stage of the workflow, and update the access control properties. Add the users to the roles that correspond to the permissions that the users require on the draft pages that they create.

**Contributor role for creating child pages**

Users with Contributor access to the published version of a page can create child pages under that page. When in edit mode on the parent page, contributors can use the site toolbar to create a child page.

**Related concepts:**

- "Workflow and change management" on page 135

Related information:
- Roles
- Resources
- Access permissions
- User roles and access
- Initial Access Control Settings

**Portal Scripting Interface and project support**

With the Portal Scripting Interface, you can create JACL or Jython scripts to automate the management of projects.

Using the Project bean with the Portal Scripting Interface, you can perform the following actions on projects:

- List all available projects
- Create and delete projects
- Retrieve information about a specific project
• Retrieve locale-specific attributes for projects
• Approve projects
• Publish projects
• Set active project

To run commands with the Project bean, you can use the Portal bean to set a project as the context for subsequent commands.

**List projects**

To retrieve a list of projects, use the listall method. This method returns the names of the projects.

- Jacl syntax: $Project listall
- Jython syntax: Project.listall()

**Jacl example:**

```
wsadmin>$Project listall
"TestProject1"
```

**Jython example:**

```
wsadmin>Project.listall()
"TestProject2" "TestProject1"
```

**Create projects**

To create a project, use the create method.

- Jacl syntax: $Project create "project_name"
- Jython syntax: Project.create("project_name")

**Jacl example:**

```
wsadmin>$Project create "TestProject1"
TestProject1
```

**Jython example:**

```
wsadmin>Project.create("TestProject1")
'TestProject1'
```

**Note:** If you create a project with the Portal Scripting Interface, the project is not listed with the recent projects in the project menu.

**Delete projects**

To delete a project, use the delete method.

- Jacl syntax: $Project delete "project_name"
- Jython syntax: Project.delete("project_name")

**Jacl example:**

```
wsadmin>$Project delete "TestProject1"
```

**Jython example:**

```
wsadmin>Project.delete("TestProject1")
```

**Retrieve project details**

Retrieve project details with the details method. This method returns the following information about the project: the Universally Unique Identifier (UUID), state, name, and title.

- Jacl syntax: $Project details "project_name"
- Jython syntax: Project.details("project_name")

Jacl example:
```jacl
$Project details "TestProject1"
uuid : bb571f0c-2143-4bcc-ba42-358a64d75116
state: ACTIVE
name : TestProject1
title: TestProject1
items:  
testpage1 (draft / new)
```

Jython example:
```python
wsadmin>print Project.details("TestProject1")
uuid : 255f129d-cc46-4d83-823d-80f9a28d13f5
state: ACTIVE
name : TestProject1
title: TestProject1
items:  
testpage1 (draft / new)
```

Retrieves translated attributes

If any project attributes are translated, such as the title or description, you can retrieve those attributes with the nlsget method. Specify the attribute with one of the following parameters:

- Title: `title` or `t`
- Description: `description`, `descr`, or `d`
- Jacl syntax: `$Project nlsget "project_name" attribute_parameter [locale]`
- Jython syntax: `Project.nlsget("project_name","attribute_parameter",["locale"])`

Jacl example:
```jacl
wsadmin>$Project nlsget "TestProject1" descr en
This is the description for TestProject1.
```

Jython example:
```python
wsadmin>print Project.nlsget("TestProject1", "descr", "en")
This is the description for TestProject1.
```

If you do not specify a value for the `locale` parameter, the currently selected locale is used.

Approve projects

To approve the drafts in a project, use the approve method. This method approves all draft items in the project at the same time. Before you can publish a project, all items in the project must be approved.

- Jacl syntax: `$Project approve "project_name"`
- Jython syntax: `Project.approve("project_name")`

Jacl example:
```jacl
wsadmin>$Project approve "TestProject1"
wsadmin>$Project details "TestProject1"
uuid : bb571f0c-2143-4bcc-ba42-358a64d75116
state: PENDING
name : TestProject1
title: TestProject1
items:  
testpage1 (draft / new / publish pending)
```
Publish projects

To publish a project, use the publish method. Before you can publish a project, all items in the project must be approved.

- Jacl syntax: $Project publish "project_name"
- Jython syntax: Project.publish("project_name")

Jython example:

```python
wsadmin>Project.publish("TestProject1")
```

```
wsadmin>print Project.details("TestProject1")
```

```
uuid : 255f129d-cc46-4d83-823d-80f9a28d13f5
state: PENDING
name : TestProject1
title: TestProject1
items:
    testpage1 (draft / new / publish pending)
```

Set active project

For commands that you want to run within a project, use the setproject method of the Portal bean to specify the project. When invoking the setproject method, you identify the active project with the name of the project. If you invoke the setproject method without specifying a project name, the active project is cleared. When you set the project during a session, the project is active immediately.

To set the active project, you must establish a user session with the portal by using the login command of the Portal bean.

- Jacl syntax: $Project setproject "project_name"
- Jython syntax: Portal.setproject("project_name")

Jython example:

```python
wsadmin>Portal.setproject("TestProject1")
```

Examples

These examples demonstrate a typical command sequence to create a page within a specific project. Each example script performs the following operations:
- Establishes a user session.
- Creates a project (myproject).
- Retrieves the details for the myproject project.
- Sets the active project to the myproject project.
- Locates the Home page in the portal.
- Creates the testpage1 page as a child page of the Home page. This operation takes place within the context of the active project.
- Clears the active project.
- Terminates the user session.

**Jacl example:**
```jacl
$Portal login
set myproject [$Project create "My new project"
$Project details $myproject
$Portal setproject $myproject
$Content find any un ibm.portal.Home select
$Content create page testpage1 html shared public
$Portal setproject
$Portal logout
```

**Jython example:**
```python
Portal.login()
myproject = Project.create("My new project")
Project.details(myproject)
Portal.setproject(myproject)
Content.find("any", "un", "ibm.portal.Home", "select")
Content.create("page", "testpage1", "html", "shared", "public")
Portal.setproject()
Portal.logout()
```

### Portal Scripting Interface and web content libraries

With the Portal Scripting Interface, you can create Jacl or Jython scripts to automate the management of web content libraries. Using the DocumentLibrary bean with the Portal Scripting Interface, you can create and delete libraries, retrieve a list of libraries, and retrieve library attributes.

### Create libraries

To create a library, use the create method.

- Jacl syntax: `$DocumentLibrary create "library_name"`
- Jython syntax: `DocumentLibrary.create("library_name")`

**Jacl example:**
```jacl
wsadmin>$DocumentLibrary create "Library1"
"library1"
```

**Jython example:**
```python
wsadmin>DocumentLibrary.create("Library1")
"library1"
```

### Delete libraries

To delete a library, use the delete method.

- Jacl syntax: `$DocumentLibrary delete "library_name"`
- Jython syntax: `DocumentLibrary.delete("library_name")`

**Jacl example:**
```jacl
wsadmin>$DocumentLibrary delete "Library1"
```

**Jython example:**
```python
wsadmin>DocumentLibrary.delete("Library1")
```
Jython example:
wsadmin>DocumentLibrary.delete("Library1")

List libraries

To retrieve a list of libraries, use the listall method.

- Jacl syntax: $DocumentLibrary listall
- Jython syntax: DocumentLibrary.listall()

Jacl example:
wsadmin>$DocumentLibrary listall
"template page content" "wiki template v70" "blog template v70" "web resources v70"
"portal site" "web content templates" "blog solo template v70" "web content"
"library1"

Jython example:
wsadmin>DocumentLibrary.listall()
"template page content" "wiki template v70" "blog template v70" "web resources v70"
"portal site" "web content templates" "blog solo template v70" "web content"
"library1"

Retrieve library details

Retrieve library details with the details method. This method returns the following information about the library: the Universally Unique Identifier (UUID), state, name, and title.

- Jacl syntax: $DocumentLibrary details "library_name"
- Jython syntax: DocumentLibrary.details("library_name")

Jacl example:
wsadmin>$DocumentLibrary details "Library1"
uuid : 64fa541a-a189-4ed6-8a6f-4c3dcc148295
name : library1
title : Library1
description: enabled : true
deletion prohibited: false

Jython example:
wsadmin>print DocumentLibrary.details("Library1")
uuid : 13b06eb0-52c7-415b-9a93-4195966a2a3
name : library1
title : Library1
description:
enabled : true
deletion prohibited: false

XML configuration interface and managed pages
You can use the XML configuration interface (xmlaccess command) to manipulate managed pages just as you can for other portal resources.

Project scope

When using the XML configuration interface with managed pages, processing occurs either completely within a project or completely outside a project.

When you import a published page and specify a project scope, the page is created as a draft page in the project.
**Important:** You cannot export a draft page and then import that page as a draft in another project.

To run the `xmlaccess` command from within a project, specify the project either with the project name or with the object ID of the project. The command uses the following format, depending on how you identify the project:

```
xmlaccess -in input_file -url http://hostname:port_number/wps/config/$project/project_name
xmlaccess -in input_file -url http://hostname:port_number/wps/config/$project/project_object_id
```

For example:

- `xmlaccess -in Export.xml -url http://www.example.com:10039/wps/config/$project/myproject`
- The following command must be entered all in one line:
  ```
  xmlaccess -in Export.xml -url
  http://www.example.com:10039/wps/config/$project/
  Z6QReDeN9E6046P9CGJMK633P8JMG6J1P8MM47MPD6MMCC63P13L6GP63R46J1
  ```

**AIX Linux Solaris note:** You might need to precede the percent sign (%) with a backslash (\) to prevent `$project` from being interpreted as an environment variable. For example:

```
xmlaccess -in Export.xml -url http://www.example.com:10039/wps/config/$project/myproject
```

**Use transaction processing with the XML configuration interface**

Because managed pages are stored in the Portal Site library in Web Content Manager, each page has corresponding objects in the JCR database. You must be aware of this relation when you create, update, or delete managed pages with the XML configuration interface. If `xmlaccess` processing is interrupted, it can result in a mismatch between the page state and database state.

To ensure that page and database information for a managed page remain synchronized, use the `transaction-level` attribute of the `request` element in the XML file. For more information about using the `transaction-level` attribute, see XML configuration reference.

Example:

```
<request
  type="update"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="PortalConfig_8.0.0.xsd"
  transaction-level="resource">

Lost-found site area

When Web Content Manager cannot determine the proper location of a page item in the Portal Site library, the page item is stored in the **lost-found** site area. This site area ensures that you can recover pages and any content stored beneath the pages in the event of a problem.

The following causes are typical reasons that a page might be stored in the lost-found site area:

- The synchronization process between the portal and Web Content Manager finds page items in the Portal Site library that do not have corresponding pages in the portal.
- You move a managed page beneath an unmanaged page in the portal page hierarchy. In this case, the page item for the managed page is stored in the lost-found site area.
Web content items

When creating a new website there are a set of core web content items that must be created including templates and site framework items.

Creating authoring templates

An authoring template is used to define the default authoring settings for site areas and content items. This includes the design of the authoring form, what fields and elements appear on the authoring form, and default values for fields and elements. You must create authoring templates before creating site areas and content items.

Related concepts:

“Working with authoring templates”

When working with authoring templates you define settings for the authoring template itself, as well as defining default settings for the items created using the authoring template.

Working with authoring templates

When working with authoring templates you define settings for the authoring template itself, as well as defining default settings for the items created using the authoring template.

Authoring template types

Site area templates

Site area authoring templates are used to define the default settings of site areas.

Default site area template:

A default site area is installed with IBM Web Content Manager. This can be used to maintain the behavior of site areas migrated from previous releases. You can disable the creation of new site areas using the default site area template by changing the following configuration parameter to "false" in the WCM WCMConfigService service using the WebSphere Integrated Solutions Console.

wcm.authoringui.defaultSiteAreaTemplateEnabled=false

Content templates

Content authoring templates are used to define the default settings for content items. You create render content in two different ways:

Content rendering

If you configure a content authoring template to render as content, then the content items you create will be standard content items. These are designed to store elements that can be rendered within presentation templates.

Resource rendering

If you configure a content authoring template to render as a resource, then the content items you create will be based on a file stored in a file resource element. When a resource content item is rendered, the file stored in the selected file resource element is saved.
rendered on the web page. No presentation template is used when the file is rendered, only the content of the file itself. This is useful when you want to store a file, such as a PDF file, and render it directly on a page but would also like to have the PDF file listed in navigational components such as menus and navigators.

**Defining authoring template properties**

**Element selection**
When creating an authoring template you add elements to the template to determine what types of content will be stored within the item. When constructing the authoring template, you can select more than one element field of the same element type. For example, you might add three text element fields, two rich text element fields, and four image element fields to the same authoring template.

**Default values**
You can specify default values for each field and element in the authoring template. Well selected default values can make it easier and more efficient for an author to create new items and can streamline the item creation process.

**Simplified form layout**
The authoring template provides features that help you simplify the presentation of the authoring form.

**Authoring form layout options**
You can control the general layout of the fields on the authoring form by specifying an authoring form layout option. Depending on the layout option you select, this can reduce the vertical space required to display the elements on the authoring form.

**Hidden fields**
In addition to organizing an authoring form with a layout option for the fields, you can further simplify the form presented to the item author through the use of hidden fields. With the exception of those fields that are required for an authoring form, you can designate any other field in the authoring template to be hidden. A field marked to be hidden in the authoring template is not displayed on the authoring form, thereby streamlining the form’s visual appearance. Note, however, that although a hidden field is not displayed on the authoring form, the information defined in the field is still associated with the authoring form and will be processed with the form. This is particularly useful when used in conjunction with a default value for a field because it enables you to specify a setting for a field and then hide the field on the authoring form to ensure that the field’s value cannot be changed by the item author. For example, you might want to set access control levels for item generated from the authoring template in the Access Control section of the template and then hide that section on the resulting authoring form. When an item is generated from the template, the access control levels for the item will be derived from the default values in the template.

**Custom help text**
To further help tailor the content form for a item author, Web Content Manager provides the capability of adding customized help text to the authoring template.
You can define help text for the entire authoring form that is generated from the authoring template. For example, this help text can be used to describe the purpose of the form. You should include whatever specific information you feel would be of use to the authors using the form.

In addition to the HTML text you can add to describe the entire authoring form, you can also specify in-line help text that is displayed with each element on the form. This help text can provide targeted information for a particular field on the form, explaining possible values or noting special conditions related to the field.

Labeling elements

The names of element labels in different items must be the same if an element reference in a presentation template is to change depending on the current context. This is an important consideration if two authoring templates will be using the same presentation template. The element types however, do not have to be consistent.

<table>
<thead>
<tr>
<th>Site area</th>
<th>Element label</th>
<th>Element type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Business</td>
<td>Heading</td>
<td>Image</td>
</tr>
<tr>
<td>Personal</td>
<td>Heading</td>
<td>Rich Text</td>
</tr>
<tr>
<td>Features</td>
<td>Heading</td>
<td>Text</td>
</tr>
<tr>
<td>News</td>
<td>Heading</td>
<td>Text</td>
</tr>
</tbody>
</table>

Related tasks:
“Creating authoring templates” on page 173

An authoring template is used to define the default authoring settings for site areas and content items. This includes the design of the authoring form, what fields and elements appear on the authoring form, and default values for fields and elements. You must create authoring templates before creating site areas and content items.

Customizing elements using JSP

A “custom JSP” field is available on some element types when added to an authoring template. You use this to reference a JSP file to use instead of the element default view in the user interface. You can write JSP to control the look and feel of an element, and to restrict the values that can be entered into an element.

Storing JSP Files:

JSP files can be located:
- within the `was_profile_root`/installedApps/node-name/wcm.ear/ilwwcm.war directory of your server.
- within any other web application running on portal. When referencing JSP files in another web application, use the following path: `contextPath;jspPath`

For example: `/wps/customapplication;/jsp/editor.jsp`
Custom bean and EditorBean API:

The CustomBean and EditorBean API can be found under
com.ibm.workplace.wcm.api.authoring in the Javadoc located under the
was_profile_root\installedApps\nodename\wcm.ear\ilwwcm.war\webinterface\folder.

Referencing jsp files

When referencing a JSP file in the custom JSP field of the element properties view,
you can use the following formats.

When located within the ilwwcm.war directory of your server use this format

was_profile_root\installedApps\node-name\wcm.ear\ilwwcm.war

Note: The JSP page is also stored in the client war directory of the local
rendering portlet or of the servlet or portlet that calls the JSP, if using the
Web Content Manager API. For example, to render a JSP page on a local
rendering portlet, you would also need to store a copy of the JSP file under
was_profile_root\installedApps\node-name\PA_WCMLocalRendering.ear/
ilwwcm-localrende.war

When located within any other web application running on portal

contextPath;jspPath
   Specifies an edit mode version of the field where the JSP is located
   in another application. For example: /wps/customapplication;/jsp/editor.jsp

jspPath
   Specifies an edit mode version of the field where the JSP is located
   in same application as Web Content Manager.

editmode=contextPath;jspPath
   Specifies an edit mode version of the field where the JSP is located
   in another application.

editmode=jspPath
   Specifies an edit mode version of the field where the JSP is located
   in same application as Web Content Manager.

readmode=contextPath;jspPath
   Specifies a read mode version of the field where the JSP is located
   in another application.

readmode=jspPath
   Specifies a read mode version of the field where the JSP is located
   in same application as Web Content Manager.

readmode=contextPath;jspPath,editmode=contextPath;jspPath
   Specifies an edit mode and read mode version of the field where
   the JSPs are located in another application.

readmode=jspPath,editmode=jspPath
   Specifies an edit mode and read mode version of the field where
   the JSPs are located in same application as Web Content Manager.

Text element example

<%@ taglib uri="/WEB-INF/tld/portlet.tld" prefix="portletAPI" %>
<%@ page import="com.ibm.workplace.wcm.api.authoring.CustomItemBean" %>
<portletAPI:init />

```<%
CustomItemBean customItem =
    (CustomItemBean) request.getAttribute("CustomItemBean");
customItem.setSubmitFunctionName("myoptionsubmit");
String fvalue = (String)customItem.getFieldValue();
fvalue = fvalue.replaceAll("\\\", ",quot;").replaceAll("\\\", \\"#39;\")
%
<script language='Javascript'>
function myoptionsubmit()
{
    document.getElementById('<%=customItem.getFieldName()%>').value =
    document.getElementById('<%=customItem.getFieldName()%>_mycustomoption').value;
}
</script>
<INPUT id='<%=customItem.getFieldName()%>_mycustomoption' value='<%=fvalue%>'>
```

**Rich Text element example**

```<%@ page import="com.ibm.workplace.wcm.app.ui.portlet.widget.EditorBean"%>
<%@ taglib uri="/WEB-INF/tld/wcm.tld" prefix="wcm" %>

<%
    EditorBean editor = (EditorBean) request.getAttribute("EditorBean");
%
<script language='Javascript'>
function setHtml(id, html)
{
    document.getElementById(id + "_rte").value = html;
}
function getHtml(id)
{
    return document.getElementById(id + "_rte").value;
}
function setRichTextValue(theText)
{
    document.getElementById('<%= editor.getName()%>_rte').value = theText;
}
</script>
<textarea cols="85" rows="15" id="<%= editor.getName()%>_rte"></textarea>
```

```<script type="text/javascript">
var initialValue = document.getElementById('<%= editor.getHiddenContentFieldName()%>_inithtml').value;
var editorTextArea = document.getElementById('<%= editor.getName()%>_rte');
editorTextArea.value = initialValue;
if (initialisedRTEs != null)
{
    initialisedRTEs = initialisedRTEs + 1;
}
</script>
```

**Custom option selection element example with validation**

This example is used to create a selection list of predefined options.

```<%@ taglib uri="/WEB-INF/tld/portlet.tld" prefix="portletAPI" %>
<%@ page import="com.ibm.workplace.wcm.api.authoring.CustomItemBean" %>
```
<portletAPI:init />

<% CustomItemBean customItem =
    (CustomItemBean) request.getAttribute("CustomItemBean");
customItem.setSubmitFunctionName("mysubmit");
String fvalue = (String)customItem.getFieldValue();
fvalue = fvalue.replaceAll("\\", ",");
fvalue = fvalue.replaceAll(""", ",");
%>

<SELECT id='<%=customItem.getFieldName()%>_mycustom' >
<OPTION>
<OPTION>
<% if (((String)fvalue).compareTo("Option1") == 0) {%>
SELECTED
<% } %>
>Option1</OPTION>
<OPTION>
<% if (((String)fvalue).compareTo("Option2") == 0) {%>
SELECTED
<% } %>
>Option2</OPTION>
<OPTION>
<% if (((String)fvalue).compareTo("Option3") == 0) {%>
SELECTED
<% } %>
>Option3</OPTION>
<OPTION>
<% if (((String)fvalue).compareTo("Option4") == 0) {%>
SELECTED
<% } %>
>Option4</OPTION>
</SELECT>

<script language='Javascript'>
    function mysubmit()
    {
        var selIndex=document.getElementById('<%=customItem.getFieldName()%>_mycustom').selectedIndex;

        if (selIndex <= 0)
        {
            document.getElementById("<%=customItem.getFieldName()%>").value = "";
        }
        else
        {
            document.getElementById("<%=customItem.getFieldName()%>").value = 
                document.getElementById("<%=customItem.getFieldName()%>_mycustom'").options[selIndex].text;
        }
    }
</script>

**Date element example**

This example is used to create a selection list of predefined dates.

**Note:** Only dates can be selected, not times.

---

178  Web Content Manager Version 8.0
Number element example

This example is used to create a selection list of predefined numbers.

```html
<%@ taglib uri="/WEB-INF/tld/portlet.tld" prefix="portletAPI" %>
<%@ page import="com.ibm.workplace.wcm.api.authoring.CustomItemBean" %>

<portletAPI:init />

<%
    CustomItemBean customItem = (CustomItemBean) request.getAttribute("CustomItemBean");
    customItem.setSubmitFunctionName("mynumbersubmit");
    String fvalue = (String)customItem.getFieldValue();
    fvalue = fvalue.replaceAll("\"", "\"\"\"\").replaceAll("\"", '&39;');
%

<SELECT id='<%=customItem.getFieldName()%>_mycustomnumber' >
    <OPTION>
        <% if (((String)fvalue).compareTo("6") == 0) { %>
            SELECTED
        <% } %>
        >6</OPTION>
    <OPTION>
        <% if (((String)fvalue).compareTo("7") == 0) { %>
            SELECTED
        <% } %>
        >7</OPTION>
    <OPTION>
        <% if (((String)fvalue).compareTo("8") == 0) { %>
            SELECTED
        <% } %>
        >8</OPTION>
    <OPTION>
        <% if (((String)fvalue).compareTo("9") == 0) { %>
            SELECTED
        <% } %>
        >9</OPTION>
</SELECT>

<script language='Javascript'>
function mydatesubmit()
{
    var selIndex=document.getElementById('<%=customItem.getFieldName()%>_mycustomdate').selectedIndex;
    document.getElementById('<%=customItem.getFieldName()%>_mycustomdate').value = document.getElementById('<%=customItem.getFieldName()%>_mycustomdate').options[selIndex].text;
}
</script>

Chapter 5. Setting up a site  179
User selection element example

This example is used to create a field to enter a user name.

Building a site framework

A site framework is comprised of a hierarchical set of intelligent pages and site areas that are used to define the navigational structure of your website. You store content items within the site framework by saving them under different intelligent pages or site areas.
Planning a site framework

Intelligent pages and site areas are used to define a hierarchical site framework. Content items are saved within the site framework to give your content structure and context.

Planning a site framework

A site framework consists of a single top-level intelligent page or site area beneath which are stored other intelligent pages, site areas and content items. Although a site framework contains only a single top-level site area, you can create multiple site frameworks for use in a single website.

Site Areas:

- You use site areas to build the site framework within which you group content items. The site areas that comprise the site framework can be classified into ancestors, descendants and siblings. The vertical hierarchy of a site framework is split into ancestors and descendants. Depending on where you are within your site framework, site areas can act as ancestors and descendants. Site Areas that share the same ancestor are known as siblings.
The relationships between authoring templates and presentation templates are set in site areas.
You must select an authoring template when creating site areas. The authoring template determines which elements are available by default on the site area form.
Site area specific web content is stored in site areas in the form of elements.

**Content items:**
- You use content items to store web page specific content in the form of elements.
- You save content items within site areas.
- You can save content items within multiple site areas and multiple site frameworks.
- You must select an authoring template when creating content items. The authoring template which elements are available by default on the content item form.
- The site area a content item is located under, and the content item's authoring template, determines which presentation template to use to display elements stored within a content item.
- You use categories and keywords to profile content items that the IBM Web Content Manager application uses to generate further pieces of content such as menus.

**When to use site areas and content items**

Site areas and content items are similar item-types. They all store web content in the form of elements, but they are used in different ways:
- Site areas are used to define different sections within a site framework. They can also be used as the home content for each section of your website.
- Content items represent web page specific content and can be used in multiple site areas and multiple site frameworks.

The structure of a website will determine whether a single site framework, or multiple site frameworks are used.

In the following examples, a company sells three different product brands.
Example 1: A single website

In this example, the simplest method to use is a website comprising a single site framework with multiple site areas:

- Top site area
  - Site area for Brand A
    - Content item A1
    - Content item A2
    - Generic content item
  - Site area for Brand B
    - Content item B1
    - Content item B2
    - Generic content item
  - Site area for Brand C
    - Content item C1
    - Content item C2
    - Generic content item

Example 2: Intranet, extranet and website

In this example, a separate site framework is created for an intranet, an extranet and a website. Some content items are used in more than one site framework.
Table 35. Example 2: Intranet, extranet and website

<table>
<thead>
<tr>
<th>Intranet</th>
<th>Extranet</th>
<th>website</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Intranet</td>
<td>- Extranet</td>
<td>- Website</td>
</tr>
<tr>
<td>- Intranet site area for Brand A</td>
<td>- Extranet Site area for Brand A</td>
<td>- Web site area for Brand A</td>
</tr>
<tr>
<td>- Content item A1</td>
<td>- Content item A1</td>
<td>- Content item A1</td>
</tr>
<tr>
<td>- Content item A2</td>
<td>- Content item A2</td>
<td>- Content item A2</td>
</tr>
<tr>
<td>- Content item A3 (intranet only)</td>
<td>- Content item A4 (extranet only)</td>
<td>- Content item A5 (web only)</td>
</tr>
<tr>
<td>- Generic intranet content item</td>
<td>- Generic extranet content item</td>
<td>- Generic web content item</td>
</tr>
<tr>
<td>- Generic content item</td>
<td>- Generic content item</td>
<td>- Generic content item</td>
</tr>
<tr>
<td>- Intranet site area for Brand B</td>
<td>- Extranet site area for Brand B</td>
<td>- Web site area for Brand B</td>
</tr>
<tr>
<td>- Content item B1</td>
<td>- Content item B1</td>
<td>- Content item B1</td>
</tr>
<tr>
<td>- Content item B2</td>
<td>- Content item B2</td>
<td>- Content item B2</td>
</tr>
<tr>
<td>- Content item B3 (intranet and extranet only)</td>
<td>- Content item B3 (intranet and extranet only)</td>
<td>- Content item B4 (web only)</td>
</tr>
<tr>
<td>- Generic intranet content item</td>
<td>- Generic extranet content item</td>
<td>- Generic web content item</td>
</tr>
<tr>
<td>- Generic content item</td>
<td>- Generic content item</td>
<td>- Generic content item</td>
</tr>
<tr>
<td>- Intranet site area for Brand C</td>
<td>- Extranet site area for Brand C</td>
<td>- Web site area for Brand C</td>
</tr>
<tr>
<td>- Content item C1</td>
<td>- Content item C1</td>
<td>- Content item C1</td>
</tr>
<tr>
<td>- Content item C2</td>
<td>- Content item C2</td>
<td>- Content item C2</td>
</tr>
<tr>
<td>- Content item C3 (intranet only)</td>
<td>- Content item C4 (extranet and web only)</td>
<td>- Content item C4 (extranet and web only)</td>
</tr>
<tr>
<td>- Generic intranet content item</td>
<td>- Generic extranet content item</td>
<td>- Generic web content item</td>
</tr>
<tr>
<td>- Generic content item</td>
<td>- Generic content item</td>
<td>- Generic content item</td>
</tr>
</tbody>
</table>

Related tasks:
- Creating site areas

To create a site framework, you need to create site areas.

**Presentation templates**

Presentation templates determine the structure of each web page in your site and which elements and components are displayed on each page.

Presentation templates allow you to change the look of a page without having to update what is being displayed on a page. For example, a presentation template that displays a menu on one side of a page can be changed to place it on the other side without having to rebuild the menu. All web pages based on the same presentation template are changed.

There is little difference between building a presentation template and using HTML to build a web page. It may even be helpful to build a "mock-up" of the page you are designing in HTML before creating a new presentation template. Simply replace the different sections of your web page with references to elements using web content tags.
You need to create a separate presentation template for each page type in your site.

Related tasks:
- Creating a presentation template

You use a presentation template to define the layout of elements displayed on a web page, and to define the default properties of a web page such as the background and default font of a web page.

**Page layout**
You use HTML to define the layout of a presentation template in the same way you use HTML to define the layout of a web page.

This is an example of a possible layout of a presentation template. Although it is recommended that HTML elements (such as tables) be used to specify the exact layout of a presentation template, you do not have to use them. You can lay out your page in any way you want.

Once the layout of a page is defined, all you need to do is reference different components into the different sections of your HTML table. (You can reference more than one component within a single table cell.)

You can also enter text and HTML tags directly into a presentation template. This is useful if you have an element that needs to appear on all pages using a common presentation template. However, if that element is used on other presentation templates, it would be more efficient to save it as a component.

**Example**

This is an example of the HTML you could enter in a presentation template to set the layout of a presentation template.
Text and IBM Web Content Manager tags are then added to the different table cells to create the finished web page.

**Enabling Connect tags**

Connect tags are advanced Web Content Manager tags that can be used to retrieve data from external sources and apply custom caching. **Process connect tags** must be selected in a presentation template form for connect tags to be processed.
Related tasks:

- **Inserting an image in an element**
  You can insert images into elements containing an HTML or rich text fields

- **Inserting a link in an element**
  You can insert links into elements containing an HTML or rich text field using the Insert Link button.

- **Inserting element tags**
  You can insert elements tags for all the elements of a selected authoring template into the design of a presentation element using the Insert Element Tags button.

- **Creating web content tags**
  You use IBM Web Content Manager tags to reference elements within presentation templates and element designs.

Related reference:

- **Defining authoring tools**
  You can format an authoring tool element's look and feel in different ways, including displaying authoring tools as text-based links, or image based links.

Page style

You use HTML to define the default properties of a presentation template in the same way you use HTML to define the default properties of a web page.

Any valid HTML property can be set including:

- Margin sizes
- Text colors
- Background colors or images

Example

This is an example of the HTML you could enter in a presentation template to set default properties for a presentation template.

```html
<html>
<head></head>
<body bgcolor="#CC0000" text="#000000" link="#6666FF" vlink="#9999FF" alink="#FF33CC" leftmargin="5" topmargin="5">
</body>
</html>
```

**Note:** If the same page properties are used in more than one presentation template, they can be stored in a single text component that is itself referenced within the presentation template:

```html
<body <component name="TextComponentName"/>>
```

This means that by editing a single text component, the page properties of multiple presentation templates can quickly be updated.

Using cascading style sheets

Default style properties cannot be set for components. The default page properties will override any page properties set in a component.

Cascading style sheets can be used to control the style of components. For example, You could make the links in a menu a different color to the links in a navigator by using cascading style sheets to determine the style of different components.
Note: Where possible, it is best practice to use one cascading style sheet for an entire site. A link to the style sheet should be used, rather than embedding the style sheet.

Template maps

Template maps are used to determine which presentation templates are used to display each site area or content item.

Template map strategies

The presentation template used by an item is determined by the relationship between the item's authoring template and a presentation template defined in the authoring template, or a template map defined in a site area in the path of the current item. Template maps assigned in site areas will override those set in authoring templates.

This can result in the following relationships:

- A content item can be displayed using two different presentation templates if linked to different site areas.
- Two items using different authoring templates can be displayed using the same presentation template if both authoring templates are mapped to the same presentation template.

Defined in authoring templates

If you select a default presentation template in an authoring template, it will be used as the default presentation template for all items based on that authoring template. This will ensure that all items based on that authoring template will be rendered with the same presentation template, but it does not guarantee design consistency between other items located in the same site area. If a different template map is specified in any site area in the item path, then the template map defined in the lowest part of the item path will be used instead.

Defined in site areas

If you define a template map in a site area, this will ensure that all items based on the selected authoring template will use the same presentation template in that site area. If a different template map is specified in any child site areas of the parent site area, then the template map defined in site area in the lowest part of the item path will be used.

Template map examples

In these examples the following template maps are used:

- **Authoring Template 1** uses **Presentation Template 1** as its default presentation template
- **Authoring Template 2** uses **Presentation Template 2** as its default presentation template
- **Authoring Template 3** also uses **Presentation Template 2** as its default presentation template
- **Authoring Template 4** has no default presentation template
- **Site Area 1** has no template map
- **Site Area 2** contains a map between **Authoring Template 1** and **Presentation Template 2**
- **Site Area 1** and **Site Area 2** are located under **Site Area A**.
Site Area A contains a map between Authoring Template 4 and Presentation Template 3

The presentation template used by each item will be determined by the authoring template the item used, and the location of the item in the site framework.

Table 36. Template Map Results

<table>
<thead>
<tr>
<th>Content and location</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Content 1 using Authoring Template 1 located in Site Area 1</td>
<td>As Site Area 1 contains no template maps, Content 1 is displayed using Presentation Template 1 which is the default presentation template of Authoring Template 1.</td>
</tr>
<tr>
<td>Content 1 using Authoring Template 1 located in Site Area 2</td>
<td>As Site Area 2 contains a map between Authoring Template 1 and Presentation Template 2, Content 1 is instead displayed using Presentation Template 2.</td>
</tr>
<tr>
<td>Content 2 using Authoring Template 2 located in Site Area 1</td>
<td>As Site Area 1 contains no template maps, Content 2 is displayed using Presentation Template 2 which is the default presentation template of Authoring Template 2.</td>
</tr>
<tr>
<td>Content 3 using Authoring Template 3 located in Site Area 1</td>
<td>As Site Area 1 contains no template maps, Content 3 is also displayed using Presentation Template 2 which is the default presentation template of Authoring Template 3.</td>
</tr>
<tr>
<td>Content 4 using Authoring Template 4 located in Site Area 1</td>
<td>As Site Area 1 contains no template maps, Content 4 is displayed using Presentation Template 3 which is mapped to Authoring Template 4 in Site Area A.</td>
</tr>
</tbody>
</table>

Element references

When referencing elements in a presentation it is important to note the following:

- The elements a presentation template uses must be defined in the authoring template the content is based on.
- If the element being referenced does not exist in the current item, nothing is displayed in that section of the presentation template.

Although the template author can identify a number of elements that can be displayed on the item form, whether the elements are displayed depends on the presentation template used with the authoring template to render the content form. A presentation template might not include every element defined in an authoring template, but in order for an element or element type to be available to a presentation template, the element must be included in the authoring template used to create the content.

Content items

You use content items to store web page specific content in the form of elements.

Each content item you create is based on an authoring template. The authoring template can determine what element types are included in a content item, whether a workflow will be used or not and what workflow to use, and where the content item can be saved.
Each content item you create is the equivalent to a web page in a traditional website. Unlike traditional websites, a single content item can be linked to different areas within your website, or linked to a different website altogether. The changes you make to a single content item will be visible in every place you link the content item to.

The look and feel of a content item when displayed in a website will depend on what authoring template was used to create the content item, and what presentation template is used to display the content. The presentation template used will depend on the current context of the content item, and which template-map applies to the current context of the content item.

Related tasks:

Creating content items

Content items are based on authoring templates. The fields displayed in a content item form can be hidden from different users, so not all of the following steps might be required. Some fields and elements might already contain default data.

Components

You use components to store elements that are used in more than one area of your website. For example, a company logo or a copyright notice.

Static components

Static components are used to store static content such as text, files or images.

- component reference component
- date and time component
- file resource component
- HTML component
- image component
- JSP component
- link component
- number component
- rich text component
- short text component
- style sheet element
- text component
- user selection component

Dynamic components

Dynamic components are used dynamically generate content based on the parameters set in the component properties.

- menu component
- navigator component
- Personalization component
- taxonomy component
- user name component
Tool components

Tool components are used to create tools that can be added to web pages for users to perform tasks such as search, inline editing, and paging through pages of links.

- authoring tools component
- page navigation component
- search component

Profiling strategies

You use the profiling features of IBM Web Content Manager to group content items into different types of content.

Profiling methods

You use the following profiling methods to group content items.

Profiling content items

Taxonomies and categories

A **Category** refers to the subject matter of your content item. For example, your content item may be of the category *New Products* or *Latest News*. You use taxonomies to group categories. Users select from a predefined list of categories when profiling a Content item.

In most cases, you should only use bottom-level categories to profile content. This will give you more control over what will display in menus.

Keywords

You also use **Keywords** to profile content. Unlike categories, which are chosen from a predefined list, you can enter any keywords you like when creating content items.

Additional profiling options

Profiling users

You can add profile information to users. This can be used as search parameters in menu elements.

Profiling rendering portlets

You can enter profile information when configuring a rendering portlet. This can be used as search parameters in menu elements.

Profiling versus access controls

Profiling can be used to personalize a website for different users. This is different from using item access controls to limit what items a user can access. In a profile based personalized site, although a user may not be able to access all the pages via personalized menus, they may still be able to access other pages by using navigators, or by searching for content. Using access controls limits a user to only view items that they have been granted access to.
Related tasks:

**Editing user profiles**
You can add profile information to users. User profiles can be used as search parameters in menus.

**Profiling a rendering portlet**
Use the Portlet Profile section to select categories and site areas that can be used as menu search options.

**Planning a taxonomy**
Before creating a taxonomy, you should analyze how the taxonomy will be used in your site to determine the best structure for your taxonomy.

You group categories within taxonomies.

You cannot use taxonomies in menu searches. If you would like to have a menu return results based on content profiled with any category in a taxonomy, you should create a single top-level category and base the menu on the top-level category.

**“Metabank” example taxonomy**

- Financial
  - Banking Solutions
  - Interest Rates
    - Personal
    - Business
    - Corporate
- News

In this example:
- Taxonomy = MetaBank taxonomy
- “Financial” is the ancestor of “Interest Rates”, “Personal”, “Business”, “Corporate” and “Banking Solutions”.
- “Personal”, “Business” and “Corporate” are the descendants of “Interest Rates” and “Financial”.

**Using categories to profile content**

When building a hierarchy of taxonomies and categories it is important to consider how a menu will use your categories in a search. This is because menus search both upwards and downwards within groups of categories.

Table 37. Examples

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>If you base a menu on a top-level category, all content profiled with</td>
<td>In the previous example, a menu based on Financial will display content</td>
</tr>
<tr>
<td>categories belonging to that top-level category and their descendants</td>
<td>profiled with any of the following:</td>
</tr>
<tr>
<td>will appear in the menu.</td>
<td>• Financial</td>
</tr>
<tr>
<td></td>
<td>– Banking Solutions</td>
</tr>
<tr>
<td></td>
<td>– Interest Rates</td>
</tr>
<tr>
<td></td>
<td>- Personal</td>
</tr>
<tr>
<td></td>
<td>- Business</td>
</tr>
<tr>
<td></td>
<td>- Corporate</td>
</tr>
</tbody>
</table>
Table 37. Examples (continued)

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Example</th>
</tr>
</thead>
</table>
| If you base a menu on a mid-level category, all content profiled with the mid-level category or its descendants or its ancestors appear in the menu. | A Menu based on Interest Rates, it will display Content profiled with any of the following:  
  - Financial  
    - Interest Rates  
    - Personal.  
    - Business.  
    - Corporate. |
| If you base a menu on a bottom-level category, all content profiled with the bottom-level category or its ancestors will be returned by the menu. | A Menu based on Business will display content profiled with any of the following:  
  - Financial  
    - Interest Rates  
    - Business. |

**Elements**

Create a website by building templates, and specifying other key components of the system. Once these items are in place, you can begin adding elements to your templates and use them to create content items.

**Creating links and navigation**

You use these elements to define or generate links between different pages in a website, or add navigational elements to a website.

**Link element**

A link element stores a link to a web content item, or to external content such as a web page.

Most web pages contain links, either to other web pages, or to files. In a web content site, most links are generated using navigator and menu elements. A link element stores a link that is not part of a site's navigation. For example, you can store a link to your "home" content item in a link element. You then add a reference to the link element in every presentation template used by your site to enable users to return to the "home" content item. If you want to change your "home" content, you only need to change the selected item in the link element.

**Creating a link element**

To create a link element, you can either add a link element to an authoring template, site area or content item, or create a link component.

**Related tasks:**

- **Using a link element**
  A link element stores a link to a web content item, or to external content such as a web page.

- **Using a menu element**
  To create a menu element you must specify the criteria to search content items with, and then create a layout for the metadata or content to be displayed in the menu element.
Related concepts:

“Menu element”

A menu element displays metadata and content from content items that match the search criteria of the menu element. The search criteria of a menu element can include matching site areas, authoring templates, categories and keywords.

Menu element:

A menu element displays metadata and content from content items that match the search criteria of the menu element. The search criteria of a menu element can include matching site areas, authoring templates, categories and keywords.

Creating a menu element

You can only use a menu element by creating a menu component. You cannot add a menu element to authoring templates, site areas or content items.

Menu search options

Menu element search options are defined in the Menu Component Query section of the menu element form. These search options define which content items from your site will be displayed in the menu element. Search options can include a combination of search parameters including searches based on authoring templates, categories and site areas.

Menus can search for the following in a website:
- Content with matching authoring templates
- Content with matching site areas
- Content with matching categories
- Content with matching keywords

Between different criteria, menu searches are "and" searches, but within each search criteria, menu searches are "or" searches. For example, a menu element that searches for two different categories and an authoring template will display content items profiled with at least one of each profile type. Content that matches only one profile type are not displayed.

Menus will not display search results if you select a search criteria but do not enter any search parameters. For example, if the menu is configured to display results based on categories, but no categories are specified in the menu form, then no matches are displayed.

Menu sorting options

You can sort menu search results according to following criteria:
- Content document name
- Content document description
- Published date
- Expired date
- General date one
- General date two
- Last modified date

You can select a maximum of three sort options.
Menu paging options

IBM Web Content Manager provides flexible paging options to enable you to display search results are generated by the menu element.

- You can specify the number of results displayed in a menu page. For example, a menu defined to show five results per page would display only five records from the set of search results.
- You can indicate where in the results set you want to begin showing results by specifying which menu page to use as a starting point. As an example, if you are displaying five results per menu page and you want to show records 6–10, you would start showing search results with the second menu page instead of the first.
- To provide easier navigation of the search results in a menu, you can include a page navigation element in the header or the footer of the menu element. The page navigation element enables stepping forward or backward through multiple menu pages without the need for creating multiple menu elements to display the different pages.
- A large number of search results can cause a delay when the menu element is initially rendered. To prevent this delay, you can limit the maximum number of pages of results that are included in the menu. To further improve the efficiency of the menu, you can also specify how many pages of results should be read beyond the current page, so that paging performance is not affected by rebuilding the menu.

While a page navigation element is a convenient way of displaying and navigating a menu’s search results, you can use the menu’s paging options to display search results in other ways. For example, if you wanted to show the results in a 3-column table, you could create three menu elements with the same search criteria and then tailor the paging options of each menu to display different result sets:

<table>
<thead>
<tr>
<th>Menu</th>
<th>Results per page</th>
<th>Start page</th>
<th>Records displayed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Menu element 1:</td>
<td>5</td>
<td>1</td>
<td>1 to 5</td>
</tr>
<tr>
<td>Menu element 2:</td>
<td>5</td>
<td>2</td>
<td>6 to 10</td>
</tr>
<tr>
<td>Menu element 3:</td>
<td>5</td>
<td>3</td>
<td>11 to 15</td>
</tr>
</tbody>
</table>

The three menus could then be referenced within three different cells of a table row in a presentation template.

Related tasks:

"Using a menu element" on page 193

To create a menu element you must specify the criteria to search content items with, and then create a layout for the metadata or content to be displayed in the menu element.

Menu element design examples:

You format the look and feel of menu elements using HTML and placeholder tags.

Simple menu design

This example shows the basic structure of the element design used by a menu to format the search data. You enter the following tags into the Design for each menu search result section of the menu element form.
Table 39. Design for each menu search result

<table>
<thead>
<tr>
<th>Design</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="tag=%22href%22" alt="placeholder" /></td>
<td>Instead of a URL, use an href placeholder.</td>
</tr>
<tr>
<td><img src="tag=%22title%22" alt="placeholder" /></td>
<td>Instead of text, insert a title placeholder.</td>
</tr>
</tbody>
</table>

This is repeated for every link returned by the search query defined in the menu element. You can also use a TitleLink tag:

Table 40. Design for each menu search result

<table>
<thead>
<tr>
<th>Design</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="tag=%22titlelink%22" alt="placeholder" /></td>
<td>This produces the same result as the previous example.</td>
</tr>
<tr>
<td><img src="tag=%22href%22" alt="placeholder" /></td>
<td>Instead of a URL, use an href placeholder. This is where the URL of the menu item is inserted in the rendered menu.</td>
</tr>
<tr>
<td>![element](type=&quot;sitearea&quot; context=&quot;autofill&quot; key=&quot;Image&quot;)</td>
<td>Instead of a name placeholder (as in the previous example), insert an element tag. The source type can either be content or site area. The context is autofill. In this example, the field being referenced is &quot;Image&quot;. The site areas or content being returned must also contain an image element named &quot;Image&quot;. The images you store in the site area or content can be different, but they must all have the same label.</td>
</tr>
</tbody>
</table>

Adding a page navigation element to a menu design

To add navigation controls to a menu you add a reference to a page navigation element in either the footer or header.

Header

    <div>
Menu results

Table 42. Design for each menu search result. This creates a new table row for every item listed in the menu.

<table>
<thead>
<tr>
<th>Design</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;span&gt;</code></td>
<td>Instead of a URL, insert a URL placeholder here. This is where the URL of the menu item is inserted in the rendered menu.</td>
</tr>
<tr>
<td><code>&lt;a href=&quot;[placeholder tag=&quot;href&quot;]&quot;&gt;</code></td>
<td></td>
</tr>
<tr>
<td><code>[placeholder tag=&quot;name&quot;] &lt;/a&gt;&lt;br&gt;</code></td>
<td>Instead of text, insert a name placeholder.</td>
</tr>
<tr>
<td><code>&lt;/span&gt;</code></td>
<td></td>
</tr>
</tbody>
</table>

Footer

Table 43. Footer

<table>
<thead>
<tr>
<th>Design</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;span&gt;</code></td>
<td>Add a reference to a previously created page navigation element to add navigation features to a menu design. In this example, the page navigation element is referenced from a page navigation component called &quot;pagenav&quot;.</td>
</tr>
<tr>
<td><code>[component name=&quot;pagenav&quot;]</code></td>
<td></td>
</tr>
<tr>
<td><code>&lt;/span&gt;</code></td>
<td></td>
</tr>
</tbody>
</table>

Using a navigator element

Create a navigator element to display a list of links based on a section of the site framework of a site.

Navigator elements:

A navigator elements displays metadata and content from a predefined section of a site framework, usually in the form of links.

Navigators are not menus. Menus are a list of hyperlinks that take you to specific pages. Navigators can also be hyperlinks that can take you to specific pages but navigators are organized differently. Navigators present the logical arrangement of a website whereas menus are a list of related web pages in your website.

The navigator element is configured by selecting a start area and determining a child depth, a parent level, and a sibling value relative to the start area. Possible start areas are site areas, or content items.

There are also options to determine if the start area is to be displayed, if content items are to be displayed, and if the hierarchy from the start area to the current site area should be expanded.

A set of element designs is used to format the information for each branch of a navigator.

Navigators display links to different site areas in a website. As such, each site area in a website should have a default content item. Otherwise, some links in a navigator will not work.
Creating a navigator element

You can only use a navigator element by creating a navigator component. You cannot add a navigator element to authoring templates, site areas or content items.

Related tasks:

Defining navigator element design options
You use the element design options of a navigator element to determine how to display the results a navigator.

Navigator formatting examples:

You use HTML to format the layout of a navigator.

- Navigator elements can have more than one element design.
- Each level in the navigator can have its own element design.
- If you want all the levels in your navigator to look the same, then you only have to build one element design.
- If there are three levels in a site area but only two element designs in your navigator, then the last two levels in your navigator use the final element design.

The following tables contain some examples of the ways you can format the look of a navigator.

Simple two-level navigator

This example shows the basic structure of the element design used by a navigator.

Table 44. Simple two-level navigator

<table>
<thead>
<tr>
<th>Design</th>
<th>Details</th>
<th>Design</th>
</tr>
</thead>
<tbody>
<tr>
<td>Navigator result design 1</td>
<td>A <code>&lt;b&gt;</code> tag is added to display the text in the first level of the navigator in bold.</td>
<td><code>&lt;b&gt; [placeholder tag=&quot;namelink&quot; ] &lt;/b&gt;&lt;br&gt;</code></td>
</tr>
<tr>
<td>Navigator result design 2</td>
<td>The second design is repeated for every link returned by the parameters defined in the navigator element after the first level.</td>
<td><code>[placeholder tag=&quot;namelink&quot; ] &lt;br&gt;</code></td>
</tr>
</tbody>
</table>

Navigator used in a rendering portlet

In this example, a URLCmpt tag is used to create a link instead of a placeholder. This enables you to specify the name of the portal page use when viewing the links generated by the navigator:
Table 45. Navigator used in a rendering portlet

<table>
<thead>
<tr>
<th>Design field</th>
<th>Design code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Header</td>
<td>&lt;Table&gt;</td>
</tr>
<tr>
<td>Navigator result design 1</td>
<td>&lt;tr&gt;&lt;td&gt; &lt;b&gt;&lt;a HREF= &quot;&lt;URLCmpnt context=&quot;autofill&quot; type=&quot;content&quot; mode=&quot;portal&quot; portalTarget=&quot;URLMap&quot; target=&quot;_blank&quot;&gt; [Property context=&quot;autofill&quot; field=&quot;title&quot;] &lt;/a&gt;&lt;/b&gt; &lt;/td&gt;&lt;/tr&gt;</td>
</tr>
<tr>
<td>Navigator result design 2</td>
<td>&lt;tr&gt;&lt;td&gt; &lt;a HREF= &quot;[URLCmpnt context=&quot;autofill&quot; type=&quot;content&quot; mode=&quot;portal&quot; portalTarget=&quot;URLMap&quot;]&quot; target=&quot;_blank&quot;&gt; [Property context=&quot;autofill&quot; field=&quot;title&quot;] &lt;/a&gt;&lt;/td&gt;&lt;/tr&gt;</td>
</tr>
<tr>
<td>Footer</td>
<td>&lt;/Table&gt;</td>
</tr>
</tbody>
</table>

An unordered list

In this example, unordered list tags are used to format navigator results. This design requires the following navigator parameters to be selected:

- Show header, footer and separator fields for each result design

Table 46. Simple two-level navigator

<table>
<thead>
<tr>
<th>Design</th>
<th>Details</th>
<th>Design</th>
</tr>
</thead>
<tbody>
<tr>
<td>Header 1</td>
<td>The header is used to open new &lt;ul&gt; and &lt;li&gt; tags at each level in the navigator.</td>
<td>&lt;ul&gt;&lt;li&gt;</td>
</tr>
<tr>
<td>Navigator result design 1</td>
<td>The placeholder tag is used to render the name and link of each item in the navigator.</td>
<td>[placeholder tag=&quot;namelink&quot;]</td>
</tr>
<tr>
<td>Separator 1</td>
<td>The separator field closes and open the &lt;li&gt; tag for each item displayed in the navigator.</td>
<td>&lt;/li&gt;&lt;li&gt;</td>
</tr>
<tr>
<td>Footer 1</td>
<td>The footer is used to close &lt;/ul&gt; and &lt;/li&gt; tags at each level in the navigator.</td>
<td>&lt;/li&gt;&lt;/ul&gt;</td>
</tr>
</tbody>
</table>
Table 46. Simple two-level navigator (continued)

<table>
<thead>
<tr>
<th>Design</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Presentation Markup 2</td>
<td>In this example, you do not need to add any markup to the Presentation Markup 2. The markup entered in Presentation Markup 1 is automatically used for the other levels of the navigator.</td>
</tr>
</tbody>
</table>

Navigator type examples:

You can use navigator elements to display different sections of a site framework in different ways.

**Breadcrumb navigators**

If a website is large and complex, a user can easily lose orientation. A breadcrumb allows the user to see the position of the current web page within the web site and the logical path back to the highest level of the site framework. A breadcrumb does not provide the actual path that the user has traversed in the website; the Back button in the browser provides this. A breadcrumb is the orientation device that shows a user where the displayed web page fits within the site framework.

You use the following configuration settings to create a breadcrumb navigator:

Table 47. Breadcrumb navigator parameter settings

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start Type</td>
<td>Current site area</td>
</tr>
<tr>
<td>Include Start</td>
<td>Yes</td>
</tr>
<tr>
<td>Ancestor Level</td>
<td>All</td>
</tr>
<tr>
<td>Descendant Level</td>
<td>None</td>
</tr>
<tr>
<td>Preceding siblings Level</td>
<td>None</td>
</tr>
<tr>
<td>Next Siblings Level</td>
<td>None</td>
</tr>
<tr>
<td>Show Top</td>
<td>No</td>
</tr>
<tr>
<td>Show Content</td>
<td>Yes</td>
</tr>
<tr>
<td>Expand navigator to display current site area</td>
<td>No</td>
</tr>
<tr>
<td>Expand current navigator branch one level</td>
<td>No</td>
</tr>
</tbody>
</table>
### Table 47. Breadcrumb navigator parameter settings (continued)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>Highlight current site area or content using final navigator result design</td>
<td>No</td>
</tr>
</tbody>
</table>

### Site map navigators

A site map provides, at a glance, the framework of your site. A site map is a navigator component that displays that part of the site framework that you define.

To create a site map, the navigator is configured as follows:

### Table 48. Site map navigator parameter settings

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start Type</td>
<td>Current site</td>
</tr>
<tr>
<td>Include Start</td>
<td>Yes</td>
</tr>
<tr>
<td>Ancestor Level</td>
<td>None</td>
</tr>
<tr>
<td>Descendant Level</td>
<td>2 Levels</td>
</tr>
<tr>
<td>Preceding siblings Level</td>
<td>None</td>
</tr>
<tr>
<td>Next Siblings Level</td>
<td>None</td>
</tr>
<tr>
<td>Show Top</td>
<td>No</td>
</tr>
<tr>
<td>Show Content</td>
<td>No</td>
</tr>
<tr>
<td>Expand navigator to display current site area</td>
<td>No</td>
</tr>
<tr>
<td>Expand current navigator branch one level</td>
<td>No</td>
</tr>
<tr>
<td>Highlight current site area or content using final navigator result design</td>
<td>No</td>
</tr>
</tbody>
</table>

### Using a search element

A search element is used to display the results of a search query. A search element cannot be used in isolation, but must be used together with an HTML element that is used to define the search query form.

To create a search form you must:
1. create a search query using an HTML element
2. create a search results view using a search component

You reference both the HTML element and search component in a single presentation template. The search component is only rendered after a search query is executed by a user.
Related concepts:
“Enabling search for web content” on page 107

You use Portal Search to search for text displayed in web sites created by IBM Web Content Manager.

Search query examples
These are examples of search queries you can create using an HTML element.

Simple search query
This is an example of a simple search query form:

Table 49. Simple search query

<table>
<thead>
<tr>
<th>Code example</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;form action='&lt;PathCmpnt type=&quot;servlet&quot; /&gt;' /library/sitearea/content' method=&quot;post&quot;&gt;</td>
<td>This is the form header where you specify the location of the content item containing the search element used to display the search result. This is typically the same content item that this HTML element is stored in.</td>
</tr>
<tr>
<td>&lt;table&gt;</td>
<td></td>
</tr>
<tr>
<td>&lt;tr&gt;&lt;td&gt;</td>
<td></td>
</tr>
<tr>
<td>&lt;input type=&quot;text&quot; name=&quot;search_query&quot;/&gt;</td>
<td>This is the body of the search form. Like any standard HTML form, it contains an input field and a submit button.</td>
</tr>
<tr>
<td>&lt;/td&gt;&lt;/tr&gt;</td>
<td></td>
</tr>
<tr>
<td>&lt;tr&gt;&lt;td align=&quot;right&quot;&gt;</td>
<td></td>
</tr>
<tr>
<td>&lt;input type=&quot;submit&quot; value=&quot;Search&quot;/&gt;</td>
<td>In this example, a table has been used to format the search query form.</td>
</tr>
<tr>
<td>&lt;/td&gt;&lt;/tr&gt;</td>
<td></td>
</tr>
<tr>
<td>&lt;/table&gt;</td>
<td></td>
</tr>
<tr>
<td>&lt;/form&gt;</td>
<td>This closes the form.</td>
</tr>
</tbody>
</table>

Searching metadata
In this example, two more fields have been added allowing users to search both content title and author name:
Table 50. Searching metadata

<table>
<thead>
<tr>
<th>Code example</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;form action='&lt;PathCmpnt type=&quot;servlet&quot; /&gt; /library/sitearea/content' method=&quot;post&quot;&gt;</code></td>
<td>This is the form header where you specify the location of the content item containing the search element used to display the search result. This is typically the same content item that this HTML element is stored in.</td>
</tr>
<tr>
<td><code>&lt;table&gt;</code></td>
<td>This is the body of the search form. Like any standard HTML form, it contains input fields and a submit button.</td>
</tr>
<tr>
<td><code>&lt;tr&gt;</code></td>
<td>This closes the form.</td>
</tr>
<tr>
<td><code>&lt;td&gt;Content Title&lt;/td&gt;</code></td>
<td><code>&lt;/table&gt;</code></td>
</tr>
<tr>
<td><code>&lt;td&gt;&lt;input type=&quot;text&quot; name=&quot;search_title&quot;/&gt;&lt;/td&gt;</code></td>
<td><code>&lt;form action='&lt;PathCmpnt type=&quot;servlet&quot; /&gt; /library/sitearea/content' method=&quot;post&quot;&gt;</code></td>
</tr>
<tr>
<td><code>&lt;tr&gt;</code></td>
<td><code>&lt;table&gt;</code></td>
</tr>
<tr>
<td><code>&lt;td&gt;Author's Name&lt;/td&gt;</code></td>
<td><code>&lt;tr&gt;&lt;td align=&quot;right&quot;&gt;</code></td>
</tr>
<tr>
<td><code>&lt;td&gt;&lt;input type=&quot;text&quot; name=&quot;search_authors&quot;/&gt;&lt;/td&gt;</code></td>
<td><code>&lt;input type=&quot;submit&quot; value=&quot;Search&quot;/&gt;</code></td>
</tr>
<tr>
<td><code>&lt;tr&gt;</code></td>
<td><code>&lt;/table&gt;</code></td>
</tr>
<tr>
<td><code>&lt;td&gt;Content Body&lt;/td&gt;</code></td>
<td><code>&lt;/form&gt;</code></td>
</tr>
<tr>
<td><code>&lt;td&gt;</code></td>
<td><code>&lt;table&gt;</code></td>
</tr>
<tr>
<td><code>&lt;input type=&quot;text&quot; name=&quot;search_query&quot;/&gt;</code></td>
<td><code>&lt;tr&gt;</code></td>
</tr>
<tr>
<td><code>&lt;/td&gt;&lt;/tr&gt;</code></td>
<td><code>&lt;td align=&quot;right&quot;&gt;</code></td>
</tr>
<tr>
<td><code>&lt;tr&gt;&lt;td align=&quot;right&quot;&gt;</code></td>
<td><code>&lt;input type=&quot;submit&quot; value=&quot;Search&quot;/&gt;</code></td>
</tr>
<tr>
<td><code>&lt;/td&gt;&lt;/tr&gt;</code></td>
<td><code>&lt;/table&gt;</code></td>
</tr>
<tr>
<td><code>&lt;/table&gt;</code></td>
<td><code>&lt;/form&gt;</code></td>
</tr>
</tbody>
</table>

Including hidden data

In this example, a hidden field has been added to restrict the search to content that use the authoring template called "Press Release":

Table 51. Including hidden data

<table>
<thead>
<tr>
<th>Code examples</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;form action='&lt;PathCmpnt type=&quot;servlet&quot; /&gt; /library/sitearea/content' method=&quot;post&quot;&gt;</code></td>
<td>This is the form header where you specify the location of the content item containing the search element used to display the search result. This is typically the same content item that this HTML element is stored in.</td>
</tr>
</tbody>
</table>
Table 51. Including hidden data (continued)

<table>
<thead>
<tr>
<th>Code examples</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;input type=&quot;hidden&quot; name=&quot;search_authoringtemplate&quot; value=&quot;Press Release&quot;/&gt;</code></td>
<td>Here a hidden input field has been added that searches for content that use the authoring template called &quot;Press Release&quot;.</td>
</tr>
<tr>
<td><code>&lt;table&gt;</code></td>
<td>This is the body of the search form. Like any standard HTML form, it contains input fields and a submit button.</td>
</tr>
<tr>
<td><code>&lt;tr&gt;</code></td>
<td>This closes the form.</td>
</tr>
<tr>
<td><code>&lt;td&gt;Content Title&lt;/td&gt;</code></td>
<td></td>
</tr>
<tr>
<td><code>&lt;td&gt;&lt;input type=&quot;text&quot; name=&quot;search_title&quot;/&gt;&lt;/td&gt;</code></td>
<td></td>
</tr>
<tr>
<td><code>&lt;tr&gt;</code></td>
<td></td>
</tr>
<tr>
<td><code>&lt;td&gt;Author's Name&lt;/td&gt;</code></td>
<td></td>
</tr>
<tr>
<td><code>&lt;td&gt;&lt;input type=&quot;text&quot; name=&quot;search_authors&quot;/&gt;&lt;/td&gt;</code></td>
<td></td>
</tr>
<tr>
<td><code>&lt;tr&gt;</code></td>
<td></td>
</tr>
<tr>
<td><code>&lt;td&gt;Content Body&lt;/td&gt;</code></td>
<td></td>
</tr>
<tr>
<td><code>&lt;td&gt;&lt;input type=&quot;text&quot; name=&quot;search_query&quot;/&gt;&lt;/td&gt;</code></td>
<td></td>
</tr>
<tr>
<td><code>&lt;tr&gt;&lt;td align=&quot;right&quot;&gt;</code></td>
<td></td>
</tr>
<tr>
<td><code>&lt;input type=&quot;submit&quot; value=&quot;Search&quot;/&gt;</code></td>
<td></td>
</tr>
<tr>
<td><code>&lt;/table&gt;</code></td>
<td></td>
</tr>
<tr>
<td><code>&lt;/form&gt;</code></td>
<td></td>
</tr>
</tbody>
</table>

Search result examples

These are examples of how to design your search results.

Search element design example for use in a website

In this example, a table is used to lay out the search results.

Table 52. Search element design example for use in a website

<table>
<thead>
<tr>
<th>Design field</th>
<th>Details</th>
<th>Code example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Header</td>
<td></td>
<td><code>&lt;table&gt;</code></td>
</tr>
<tr>
<td>Result</td>
<td>The attributes to display in each search result are defined here.</td>
<td><code>&lt;tr&gt;&lt;td&gt;</code> <code>&lt;attributeResource attributeName=&quot;namelink&quot;/&gt;&lt;br&gt;</code> <code>&lt;attributeResource attributeName=&quot;summary&quot;/&gt;</code> <code>&lt;/td&gt;</code> <code>&lt;/tr&gt;</code></td>
</tr>
<tr>
<td>Separator</td>
<td>A separator can be used to delineate each search result.</td>
<td><code>&lt;tr&gt;&lt;td bgcolor=&quot;#FFFAA&quot; colspan=&quot;2&quot;&gt;&lt;/td&gt;</code> <code>&lt;/tr&gt;</code></td>
</tr>
<tr>
<td>Footer</td>
<td>A page navigation element stored in a component is referenced here to add page navigation to the search results.</td>
<td><code>&lt;tr&gt;&lt;td&gt;</code> <code>&lt;component name=&quot;pagenavigationcomponent&quot;/&gt;</code> <code>&lt;/td&gt;</code> <code>&lt;/tr&gt;</code> <code>&lt;/table&gt;</code></td>
</tr>
</tbody>
</table>

No results

There are no results for your query. Please refine your search and try again.
Search element design example for use in a rendering portlet

In this example, a table is used to lay out the search results.

Table 53. Search element design example for use in a rendering portlet

<table>
<thead>
<tr>
<th>Design field</th>
<th>Details</th>
<th>Code example</th>
</tr>
</thead>
</table>
| Header       | The attributes to display in each search result are defined here. When displaying search results in a rendering portlet you must specify the page that the linked content is displayed in when opened. A URL map to the portal page that contains the rendering portlet is required. | `<tr><td>
  <a href="/[PORTAL_CONTEXT_ROOT]/[PORTAL_PAGE_URL_MAPPING]?WCM_GLOBAL_CONTEXT=
  <AttributeResource attributeName="url" />
  <AttributeResource attributeName="title" />
  <AttributeResource attributeName="summary"/>
</td></tr>` |
| Result       | A separator can be used to delineate each search result. | `<tr bgcolor="#FFFAA" colspan="2"/>
<tr>` |
| Footer       | A page navigation element stored in a component is referenced here to add page navigation to the search results. | `<tr><td>
  [component name="pagenavigationcomponent"]
</td></tr>
</table>` |
| No results   | There are no results for your query. Please refine your search and try again. | |

Storing text and HTML

You use different types of elements to store text or HTML depending on the type of text or HTML being created.

**Tip:** Text that is common to all content using a single presentation template should be entered directly in the presentation template rather than separate elements.

**Text, rich text and HTML elements**

You use the short text, text, rich text and HTML elements to store blocks of text, but each has slightly different properties.

**Creating an element**

Short text, text, rich text and HTML elements can be added to site areas, content items and authoring templates or they can be created as individual components.
Short text element

A short text element is used to store small amounts of fixed-length text where the length is 250 bytes or less. Unlike the other text elements, short text elements can also be used as a search parameter in a Personalization rule.

Text element

You use a text element to store larger amounts of text than can be stored in a short text element. No special processing occurs for this element.

HTML element

An HTML element is used to store fragments of HTML that can be reused in presentation templates and other elements designs. You can enter HTML directly into the element or upload HTML from a previously created HTML file.

Rich text element

A rich text element is similar to the HTML element except that it includes a rich text editor that can be used to format text stored within a rich text element. The main purpose of the rich text element is provide base-level content creators with an easy-to-use text editor. Advanced users who are required to produce more advanced code, including web content tags, or who need to store fragments of HTML should use an HTML element instead.

You should use rich text elements sparingly in authoring templates, site areas and content items as adding multiple rich text elements to these items can reduce authoring performance.

The rich text editors used by Web Content Manager are supplied by other vendors. For information on using the rich text editor, see the user documentation supplied by the specific rich text editor vendor.

Using web content tags in rich text and HTML elements

Table 54. Using web content tags in rich text and HTML elements

<table>
<thead>
<tr>
<th>Element type</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Short Text and Text elements</td>
<td>Web content tags cannot be used in short text and text elements.</td>
</tr>
</tbody>
</table>
| HTML elements | Any combination of web content tags can be used in HTML elements with the following exceptions: 1. You cannot use single quotes around attribute values.  
  - [Component name='example']  
  - [Component name='example' start='a href='' end=' '>link</a>']  
  - [Component name='example' start='<img src='' end=' '/>']  
2. You cannot use double quotes inside attribute values.  
  - [Component name='example' start='a href='' end=' '>link</a>']  
  - [Component name='example' start='<img src='' end=' '/>'] |
Table 54. Using web content tags in rich text and HTML elements (continued)

<table>
<thead>
<tr>
<th>Element type</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rich text elements</td>
<td>Basic Web Content Manager tags can be used in rich text elements. For example, the following tags can be used in Rich Text elements:</td>
</tr>
<tr>
<td></td>
<td>• [component name=&quot;test&quot;]</td>
</tr>
<tr>
<td></td>
<td>• [element type=&quot;content&quot; context=&quot;current&quot; key=&quot;body&quot;]</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> Highlighting of rich text elements is not enabled by default. To enable this support, ensure that your server administrator adds the following property to the WCM WCMConfigService service:</td>
</tr>
<tr>
<td></td>
<td>• Property name: wcm.enableWCMTagHighlighting</td>
</tr>
<tr>
<td></td>
<td>• Value: true</td>
</tr>
<tr>
<td></td>
<td>The following tag formats are invalid:</td>
</tr>
<tr>
<td></td>
<td>1. The use of single quotes around attribute values.</td>
</tr>
<tr>
<td></td>
<td>• [Component name='example']</td>
</tr>
<tr>
<td></td>
<td>• [Component name='example' start='&lt;a href=&quot;&quot; end='&quot;'/&gt;']</td>
</tr>
<tr>
<td></td>
<td>• [Component name='example' start='&lt;img src=&quot;&quot; end='&quot;'/&gt;']</td>
</tr>
<tr>
<td></td>
<td>2. The use of double quotes inside attribute values.</td>
</tr>
<tr>
<td></td>
<td>• [Component name=&quot;example&quot; start=&quot;&lt;a href=&quot;&quot; end=&quot;&quot;&gt;link&lt;/a&gt;&quot;</td>
</tr>
<tr>
<td></td>
<td>• [Component name=&quot;example&quot; start=&quot;&lt;img src=&quot;&quot; end=&quot;&quot;/&gt;&quot;]</td>
</tr>
<tr>
<td></td>
<td>3. Embedding tags inside other HTML tags.</td>
</tr>
<tr>
<td></td>
<td>• &lt;a href='[Component name=&quot;example&quot;]'&gt;link&lt;/a&gt;</td>
</tr>
<tr>
<td></td>
<td>• &lt;img src='[Component name=&quot;example&quot;]'/&gt;</td>
</tr>
</tbody>
</table>

Related tasks:

Setting service configuration properties

IBM WebSphere Portal comprises a framework of services to accommodate the different scenarios that portals need to address. Services are available for both WebSphere Portal and IBM Web Content Manager. You can configure some of these services.

**Number element**

You can store a numerical value in a number element.

**Creating a number element**

To create a number element, you can either add a number element to an authoring template, site area or content item, or create a number component.

**Storing files and images**

You use these elements to store files or images.

**File resource element**

You can store a file in a file resource element. You can then reference the file resource so that users can download the file using a link or, for supported file types, convert the file directly to HTML and render it on the page.

By storing a file in a file resource element:

• you can reference the file anywhere in your site regardless of the original file location or URL.
Creating a file resource element

To create a file resource element, you can either add a file resource element to an authoring template, site area or content item, or create a file resource component.

Using a resource content item versus using a file resource component:

If you configure an authoring template to be a resource template the content items created using that authoring template are known as resource content items. Like a file resource component, files stored within a resource content item can be directly rendered in a web page. The difference between using a resource content item instead of a file resource component is that a resource content item can be listed in navigational components such as menus and navigators.

Referencing a file resource

A file resource can be referenced within presentation templates and other element designs using either a component or element tag.

Creating a link to a file resource

To create a link to a file resource in a presentation template or element design to enable users to download the file reference, use the following tag structures.

To create a link to a file resource component you use a component tag:

```html
<a href=[component name="FileResourceName"]>Link Text</a>
```

To create a link to a file resource element you use an element tag. For example, to link to a file resource element in the current content item:

```html
<a href=[element type="content" content="current" key="FileResourceName"]>Link Text</a>
```

Rendering a file resource on a page

If your file resource is a file type that can be converted to HTML you can instead convert the file to HTML and render the converted HTML directly in your web content using the format="HTML" parameter in a component or element tag. For example:

```html
[component name="FileResourceName" format="HTML"]
[element type="content" content="current" key="FileResourceName" format="HTML"]
```

Examples of supported file types include:

- word-processing documents (*.doc, *.odt)
- spreadsheets (*.xls)
- HTML files (*.htm, *.html)
- Text files (*.txt)

Other file types may also work but you need to test them first.

Note: If you configure an authoring template to be a resource template, then the content items you create are resource content items. When a link to a resource content item is rendered, the file stored in the selected file resource element is rendered on the web page. In this case, you would not use an element tag. Instead, the file resource is rendered either using a placeholder tag in a navigator or menu design, or by writing a link directly to the resource content itself.
Image element
You store an image in an image element.

When you store an image in an image element:
- You can reference the image anywhere in your site regardless of the original file location or URL.
- The image is automatically transferred to other servers during syndication.
- You can reference the image element within other items as required.

Creating an image element

To create an image element, you can either add an image element to an authoring template, site area or content item, or create an image component.

Note: Images with names that include non-ascii characters cannot be pre-rendered. If you are pre-rendering a site, you must rename the image before adding it to the image element.

Valid mime types

You can configure which mime types are valid for an image element by editing the `imageresourcempt.allowedmimetypes` setting. See “Web content authoring options” on page 62 for further information.

JSP elements

You use a JSP element to store a path to a JSP. When rendered within a presentation template or element design, a request to a JSP is generated and processed.

Creating a JSP element

To create a JSP element, you can either add a JSP element to an authoring template, site area or content item, or create a JSP component.

Syndication:

Syndication will not move the JSP page referred to in a JSP element. Only the item containing the JSP element is moved. You will need to store the JSP page in the same folders of both the subscribing and syndicating servers.

Note:

The JSP referenced within a JSP component must not include a reference, directly or indirectly, to the same JSP component. This includes references within web content tags or the API. If it does, a loop is created and errors will occur.

Related tasks:

Using a JSP element
You use a JSP element to store a path to a JSP. When rendered within a presentation template or element design, a request to a JSP is generated and processed.

Using style sheets in items
Style sheets can be used to format the look and feel of IBM Web Content Manager pages in the same way as normal web pages.
Creating and referencing style sheet elements

Style sheet elements can only be stored in style sheet components.

To link a style sheet component to a specific authoring template, you must select a style sheet component as the default style sheet in an authoring template.

To link a style sheet component to a specific site area or content item, you must add a component reference element to a site area or content item and select a style sheet component.

Referencing a style sheet element in a presentation template

Style sheet elements are referenced in the "header" section of a presentation template using either a style element tag or component tag.

Table 55. Referencing a style sheet element in a presentation template

<table>
<thead>
<tr>
<th>Details</th>
<th>Code example</th>
</tr>
</thead>
<tbody>
<tr>
<td>To use the style sheet specified in the authoring template of the current content item, you must use a &lt;styleElement&gt; tag.</td>
<td><code>&lt;HTML&gt;</code>&lt;HEAD&gt;&lt;styleElement source=&quot;template&quot;/&gt;&lt;/HEAD&gt;&lt;BODY&gt;&lt;/BODY&gt;&lt;/HTML&gt;`</td>
</tr>
<tr>
<td>To use the style sheet selected in a component reference element stored in either the current site area or content item, you must use a &lt;styleElement&gt; tag.</td>
<td><code>&lt;HTML&gt;</code>&lt;HEAD&gt;&lt;styleElement source=&quot;path&quot; name=&quot;component reference name&quot;/&gt;&lt;/HEAD&gt;&lt;BODY&gt;&lt;/BODY&gt;&lt;/HTML&gt;`</td>
</tr>
<tr>
<td>To use a specific style sheet, you must use a &lt;component&gt; tag.</td>
<td><code>&lt;HTML&gt;</code>&lt;HEAD&gt;&lt;component name=&quot;style sheet component name&quot;/&gt;&lt;/HEAD&gt;&lt;BODY&gt;&lt;/BODY&gt;&lt;/HTML&gt;`</td>
</tr>
</tbody>
</table>

When rendered in web content, references to style sheet components are rendered as external style sheet links:

```html
<HTML>
<HEAD>
<link href="stylesheet" media="media-type" rel="stylesheet-type" type="text"/>
</HEAD>
<BODY></BODY>
</HTML>
```

Using styles in HTML tags

Styles are used in HTML stored in presentation templates and element designs in the same way as normal HTML. The style must exist in the style sheet that has been referenced in the presentation template used to render the HTML.

For example, to add a class called "wcm" to a heading tag:
Using styles in web content tags

Style sheets can be used to format the style of content retrieved using Web Content Manager tags. The style must exist in the style sheet referenced in the presentation template used to render the Web Content Manager tag.

For example, to format the links in a menu using a style sheet class called "wcm", the following placeholder tag would be used:

```html
<a href=<placeholder tag="href" />" class="wcm"><placeholder tag="name" /></a>
```

Selection elements

You use these elements to make selections from existing elements and data.

Component reference element

You use a component reference element to store a reference to a component.

You can use component reference elements in two ways:

- A component-reference element can be added to an authoring template. This then allows content creators to select an existing component from the component library rather than build the component themselves.
- By referencing a component-reference element in a presentation template or another element, the component being displayed can be changed without having to change the presentation template or element design.

For example, If you referenced a component-reference element called "Featured Product" in more than one presentation template the element referenced within the component-reference element can be changed without having to edit the presentation templates.

Creating a component reference element

To create a component reference element, you can either add a component reference element to an authoring template, site area or content item, or create a component reference component.

Date and time element

You use a date and time element to store a date or time to be displayed on a web page.

Creating a date and time element

To create a date and time element, you can either add a date and time element to an authoring template, site area or content item, or create a date and time component.

Option selection element

An option selection element is used to present a list of predefined options that your content creators can select from when creating a content item or site area. An option selection element can only be added to an authoring template.

When using an option selection element, you can either enter a custom set of options, or select a set of categories. Which option you use will depend on how you want to use the option selection element.
Simplifying item profiling

If you select a set of categories in a option selection element, you can choose to have the categories that a user selects when creating an item added to the item's profile. This means you can use multiple option selection elements on an item form to simplify the process of profiling an item. For example, if you used separate taxonomies for "product type", "team" and "campaign", you could use three separate option selection elements in an authoring template. This would make it easier for your content creators to select categories from each taxonomy on the content form.

Displaying different text on a rendered page

A simple use-case for using option selection elements is when you want to enable content creators to select from a short list of text options that will then be rendered on a page. For example, if you wanted to indicate different document types such as a "procedure", "policy" or "news" on the rendered page.

Working with custom code

Option selection elements can also be used to allow content creators to select different options that can then be used as parameters in some custom code such as a custom workflow or a JSP component.

User selection element

A user selection element stores a selection of users or groups.

Creating a user selection element

To create a user selection element, you can either add a user selection element to an authoring template, site area or content item, or create a user selection component.

Using a user selection element

Click Select Users to select users and groups.
- Select either Users or Groups.
- Type text to search for in the Search field and then click Search. Leave the Search field blank to display all users or groups. Select the required users or groups and then click OK.
- To remove a user or group, select the users or groups you would like to remove and then click Remove.

Note: Virtual users cannot be selected in a user selection element.

Personalized content

You use these elements to create or reference personalized content.

Personalization element

A personalization element stores a reference to a personalization rule or content spot generated by Portal Personalization. To use a personalization element you must create a personalization component.

A personalization element can:
display personalization content within a presentation template or element design.

display a link to personalization content within a presentation template or element design.

display attributes of personalization content within a presentation template or element design.

Creating a personalization element

You can only use a personalization element by creating a personalization component. You cannot add a personalization element to authoring templates, site areas or content items.

Note: A maximum of 100 items can be displayed in a single Personalization element.

Access controls

When creating a personalization element, a user will only be able to select those personalization rules and content spots that they have access to in Portal Personalization.

The personalization rule or content spot selected in the personalization element will only be rendered if the user viewing the web content has access to the personalization rule or content spot in Portal Personalization.

Recreating Personalization Rules and content spots

If you delete a Personalization Rule or content Spot that has been referenced in a Personalization element and then create a new Personalization Rule or content spot with the same name, it will no longer be displayed in the Personalization element. You will need to edit the Personalization element and reselect the Personalization Rule or content spot.

Caching Personalization components

Web content caching can sometimes be used with Personalization components but will depend on the conditions set in the personalization rule, or the resources used to determine the rule results. Cache testing will be required to determine if the content returned by your personalization component can be cached using web content caching.

Related tasks:
Using a Personalization element
A Personalization element stores a reference to a rule or content spot.

Personalization element examples

The layout and design of a personalization element is created in a similar way to a menu element, with a header design, footer design, and a design to be repeated for each result.

Creating a Personalized menu

1. Create a content spot or personalization rule in Portal Personalization based on some Web Content Manager content.

2. Create a personalization element in Web Content Manager.
• Click the **Search** button and select the content spot or personalization rule you created before. Click **OK**.

• Create an element design to display the results of the content spot or personalization rule. This is similar to designing a Menu element or Navigator. For example, enter the following in the **Design for each menu search result** section:

   [placeholder tag="namelink"]
   <br>

3. Save the personalization element.

4. Reference the personalization element in a presentation template.

### Displaying personalized content

To display a single piece of personalized Web Content Manager content for different users:

1. Create an authoring template that includes an element. For example, a text element called "Body".
2. Create a set of content items based on this authoring template.
3. Create a content spot or personalization rule in Portal Personalization based on the authoring template and content. The content spot or personalization rule should only return a single piece of Web Content Manager Content for each user.
4. Create a personalization element in Web Content Manager.
   • Click the **Search** button and select the content spot or personalization rule you created before. Click **OK**.
   • Create an element design to display the results of the content spot or personalization rule. For example, enter the following in the **Design for each menu search result** section:

   [element type="Content" context="autofill" key="Body"]

   This displays the content of the text element called "Body" from the content item returned by the content spot or personalization rule.

5. Save the personalization element.

6. Reference the personalization element in a presentation template.

### Displaying personalized web content components

A set of personalized web content components can be displayed using a personalization element:

1. Create a content spot or personalization rule in Portal Personalization that searches for web content components.
2. Create a personalization element in Web Content Manager.
   • Click the **Search** button and select the content spot or personalization rule you created before. Click **OK**.
   • Create an element design to display the results of the content spot or personalization rule. For example, enter the following in the **element design** section:

   **Header:**
   <div>

   **Design for each menu search result:**
   You must use a Component tag with a context of "autofill".
3. Save the personalization element.
4. Reference the personalization element in a presentation template.

**Displaying attributes of Personalized content**

The attributes of personalized content can also be displayed using a personalization element:

1. Create a content spot or personalization rule in Portal Personalization.
2. Create a personalization element in Web Content Manager.
   - Click the **Search** button and select the content spot or personalization rule you created before. Click **OK**.
   - Create an element design to display the results of the content spot or personalization rule. For example, Enter the following in the **element design** section:
     
     **Header:**
     ```html
     <div>
     Design for each menu search result:
     ```
     You must use a "AttributeResource" tag for each attribute you want to display. For example:
     ```html
     <span>
     [AttributeResource attributeName="ibmcm:title"]
     [AttributeResource attributeName="ibmcm:effectiveDate"]
     </span>
     ```
     **Footer:**
     ```html
     </div>
     ```

3. Save the personalization element.
4. Reference the personalization element in a presentation template.

**Notes:**

**Displaying keywords and categories:**

To retrieve a list of categories or keywords, Use the Property tag.

**Displaying authors and owners:**

To retrieve a list of authors or owners, Use the Property tag.

**Displaying the Site Path:**

To display the site path to a personalized Web Content Manager element, use a placeholder tag.

**Taxonomy element**

A taxonomy element defines the layout of a category selection form that enables users to select categories to display in a personalized menu.
You configure the element by selecting either a taxonomy or a category to be the start area of the category selection tree. You then select a child depth relative to the start area. Select "Include Start" if you would like the start area to display in the category selection tree. This option has no effect if the start area is a taxonomy.

There are two element design options available: one is rendered when the logged in user has selected the category that is to be displayed, and the other is rendered if the user has not selected the category. These element designs are rich text elements, and are used in a similar fashion to the navigator and menu elements.

**Note:** To use this feature you must configure a property extension database to store user-specific data. See [Managing user data](#) for further information.

### Creating a taxonomy element

You can only use a taxonomy element by creating a taxonomy component. You cannot add a taxonomy element to authoring templates, site areas or content items.

**Related tasks:**

- [Using a taxonomy element](#)

You use a taxonomy element to display a list of categories from a taxonomy.

### Creating category selection trees

You use category selection trees to allow users to personalize menus.

**Note:**

- Ensure the `connect.businesslogic.module.ajpecatselect.class` property is defined in the WCM WCMConfigService service, using the WebSphere Integrated Solutions Console, with a value of `com.aptrix.pluto.CategoryProfileUpdaterModule`.
- You cannot use category selection trees in a local rendering portlet. Instead, you must render the tree directly. For example:
  
  ```
  http://host:port/wps/wcm/myconnect/library/sitearea/content
  ```

### Taxonomy element form

The main function of the taxonomy element is to display a category selection tree used to allow a user to select categories for menu personalization.

- You configure the element by selecting either a taxonomy or a category as a start area.
- Select a child depth from the start area and a parent level relative to the start area.
- Select "Include Start" to display the start area. This option has no effect if the start area is a taxonomy.
- There are two element design options available
  - One is rendered when the logged in user has selected the category that is to be displayed.
  - The other is rendered if the user has not selected the category.
  
  These element designs are rich text elements, and are used in a similar fashion to the navigator and menu elements.

The taxonomy element form example creates a check box input form:

- The category identity number is assigned to the "value" attribute in the input fields.
• Check box input fields are created, assigning the "selectedCategories" value to the "name" attribute.
• Hidden input fields are created, assigning the "visibleCategories" value to the "name" attribute.

Element designs

The following code examples are used to develop a basic category selection tree:

### Table 56. Header

<table>
<thead>
<tr>
<th>Code</th>
<th>Details</th>
</tr>
</thead>
</table>
| [PathCmpnt end="/Library]/[SiteArea]"/Content?MOD=AJPECatSelect" method="post" start="
<FORM action='" type="servlet"] | This calls the Category Profile Updater Module. |
| <input type="hidden" value="/Library]/[SiteArea]"/Content" name="redirectURL"] | This points to the page to go to after the request is processed. The element does not render correctly if the path, "[Site area]/[Content]" is not valid. |
| <input type="hidden" name="updateSourceProfile" value="true"] | This line determines how a user's category profile is updated. The "value" parameter is optional. True permanently updates the user category profile. False updates only the user's session profile. **Updating the user's session profile:** The user's selected categories are calculated by combining the categories that are in the user's session profile and the categories that are in the permanent user category profile. Therefore, if a category is in the user category profile and is removed from the user's session profile only, it will still be shown as selected. |

### Table 57. Unselected element design

<table>
<thead>
<tr>
<th>Code</th>
<th>Details</th>
</tr>
</thead>
</table>
| <input type="checkbox" name="selectedCategories" value="[Placeholder tag="idnum"]"] <IndentCmpnt offset="0" repeat=".."] [Placeholder tag="name"] | <input type="hidden" name="visibleCategories" value="[Placeholder tag="idnum"]"] [<br> 
| | This is used to display unselected items in the rendered category selection tree. |
Table 58. Selected element design

<table>
<thead>
<tr>
<th>Code</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;input type=&quot;checkbox&quot; checked name=&quot;selectCategory&quot; value=&quot;[Placeholder tag=&quot;idnum&quot;]&quot;/&gt;</code></td>
<td>Title as categories are displayed selected items in the rendered category selection tree.</td>
</tr>
<tr>
<td><code>&lt;IndentCmpnt offset=&quot;0&quot; repeat=&quot;..&quot;/&gt;</code></td>
<td></td>
</tr>
<tr>
<td><code>&lt;input type=&quot;hidden&quot; name=&quot;visibleCategories&quot; value=&quot;[Placeholder tag=&quot;idnum&quot;]&quot;/&gt;</code></td>
<td></td>
</tr>
</tbody>
</table>

This is used to display selected items in the rendered category selection tree.

Table 59. Footer

<table>
<thead>
<tr>
<th>Code</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;input type=&quot;submit&quot; value=&quot;Set User Categories&quot;&gt;&lt;/input&gt;</code></td>
<td></td>
</tr>
</tbody>
</table>

The footer contains the submit button.

Indent element:

This example uses the indent element tag. This can be used in the navigator and taxonomy elements. This tag represents an HTML/text string that should be repeated depending on the depth of a tree node being rendered in these elements.

In the taxonomy element example, the indent element is used to render and repeat the "." string dependent on the depth of the node the element design is being applied to. It is possible to offset the repeat value by assigning an integer value to the "offset" attribute of the tag. For example, a current node depth of 5 and an offset value of -2 would render the repeat string three times. If the sum of the offset and the node depth is negative or 0, the repeat string is not rendered.

element designs:

The only difference between the unselected element design and the selected element design is that the check box input field in the selected element design has the "checked" attribute set.

User access:

If using a taxonomy element, users must be given "Edit" access to their own user item to enable them to update their selected categories.

Using a URL to update user categories:

You can use a URL as an alternative to using a category selection tree to update a user's selected categories:

http://host:port/wcm/connect/SiteArea/SelectPage?MOD=AJPECatSelect &redirectURL=/wcm/connect/SiteArea/Content&updateSourceProfile=false &selectedCategories=category1D1,category1D3 &visibleCategories=category1D1,category1D2,category1D3,category1D4

The "selectedCategories" and "visibleCategories" parameters have multiple values which are comma delimited. The categories specified in "selectCategories" should be a subset of "visibleCategories".

This URL could be used on a page in the form of a button to allow users to update their user categories. For example, You could create a button that would add the category "News" to a user's selected categories list.
Related tasks:

**Setting service configuration properties**
IBM WebSphere Portal comprises a framework of services to accommodate the different scenarios that portals need to address. Services are available for both WebSphere Portal and IBM Web Content Manager. You can configure some of these services.

**Using a user name element**
A user name element displays the name of the current user in a presentation template, component design, or element design. You can only use a user name element by creating a user name component. You cannot add a user name element to authoring templates, site areas, or content items.

To create a user name component, go to **Applications > Content > Web Content Management** and then click **New > Component > User Name**.

**Using a page navigation element**
A page navigation element provides navigation controls that are used to navigate through a set of results generated by menus, navigators, personalization and search elements.

**Related concepts:**

“Page navigation element”
A page navigation element provides navigation controls that are used to navigate through a set of results generated by menus, navigators, personalization and search elements.

**Related tasks:**

**Setting service configuration properties**
IBM WebSphere Portal comprises a framework of services to accommodate the different scenarios that portals need to address. Services are available for both WebSphere Portal and IBM Web Content Manager. You can configure some of these services.

**Page navigation element**
A page navigation element provides navigation controls that are used to navigate through a set of results generated by menus, navigators, personalization and search elements.

A page navigation element can generate two kinds of page navigation controls:
- Shuttle controls provide navigation relative to the current page. This includes sequential linking to the previous or next page of results and quick linking to the first and last pages in the set.
- Paging controls provide navigation according to the page number of the result set. A list of page number links is displayed, along with a continuation link for access to the previous or next set of page numbers, if all page numbers are not displayed.

A page navigation element can combine both shuttle and paging controls, as in the following example.
Table 60. Page navigation layout example

<table>
<thead>
<tr>
<th>Layout section:</th>
<th>First</th>
<th>Previous</th>
<th>Page numbering</th>
<th>Jump to page</th>
<th>Next</th>
<th>Last</th>
<th>Page size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Navigation example:</td>
<td>&lt;&lt;</td>
<td>&lt;</td>
<td>...</td>
<td>4,5,6</td>
<td>...</td>
<td>Go to page:</td>
<td>&gt;</td>
</tr>
</tbody>
</table>

When rendered the page navigation element might look like this:

**Using the page navigation element in a remote rendering portlet**

The page navigation element will not work when rendered in a remote rendering portlet that is configured to receive no links from other portlets. You must configure a remote rendering portlet to receive links either from "This portlet" or "Other portlets and this portlet" to use a page navigation element.

**Creating a page navigation element**

You can only use a page navigation element by creating a page navigation component. You cannot add a page navigation element to authoring templates, site areas or content items.

**Related tasks:**

"Using a page navigation element" on page 219

A page navigation element provides navigation controls that are used to navigate through a set of results generated by menus, navigators, personalization and search elements.

**Page navigation design example**

This is an example of a design of a page navigation element.

This example uses two page navigation elements to produce a page navigation system like this:

Table 61. Examples of the two page navigation elements that produce a page navigation system

<table>
<thead>
<tr>
<th>&lt;&lt;</th>
<th>&lt;</th>
<th>...</th>
<th>4,5,6</th>
<th>...</th>
<th>&gt;</th>
<th>&gt;&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Page 5 of 10.</td>
<td>Go to page:</td>
<td>4,5,6</td>
<td>...</td>
<td>&gt;</td>
<td>&gt;&gt;</td>
<td></td>
</tr>
<tr>
<td>Number of items to display: 10</td>
<td>50</td>
<td>All</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**First page navigation element**

1. Create a page navigation component named "firstnavigation".
2. Select both Shuttle (first, previous, next, and last controls) and Paging (page numbering and continuation).
3. Select Limit number of pages and type 3 in the associated field.
4. Enter the following text in these element design fields:
Table 62. First page navigation element design

<table>
<thead>
<tr>
<th>Design Element</th>
<th>Design code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Header</td>
<td>&lt;span&gt;</td>
</tr>
<tr>
<td>Footer</td>
<td>&lt;/span&gt;</td>
</tr>
<tr>
<td>Separator</td>
<td></td>
</tr>
<tr>
<td>First control - active design</td>
<td>&lt;font color=&quot;#000000&quot;&gt;&lt;&lt;&lt;/font&gt;</td>
</tr>
<tr>
<td>First control - inactive design</td>
<td>&lt;font color=&quot;#999999&quot;&gt;&lt;&lt;&lt;/font&gt;</td>
</tr>
<tr>
<td>Previous control - active design</td>
<td>&lt;font color=&quot;#000000&quot;&gt;&lt;&lt;/font&gt;</td>
</tr>
<tr>
<td>Previous control - inactive design</td>
<td>&lt;font color=&quot;#999999&quot;&gt;&lt;&lt;/font&gt;</td>
</tr>
<tr>
<td>Next control - active design</td>
<td>&lt;font color=&quot;#000000&quot;&gt;&gt;&lt;/font&gt;</td>
</tr>
<tr>
<td>Next control - inactive design</td>
<td>&lt;font color=&quot;#999999&quot;&gt;&gt;&lt;/font&gt;</td>
</tr>
<tr>
<td>Last control - active design</td>
<td>&lt;font color=&quot;#000000&quot;&gt;&gt;&gt;&lt;/font&gt;</td>
</tr>
<tr>
<td>Last control - inactive design</td>
<td>&lt;font color=&quot;#999999&quot;&gt;&gt;&gt;&lt;/font&gt;</td>
</tr>
<tr>
<td>Continuation</td>
<td>...</td>
</tr>
</tbody>
</table>

Second page navigation element

1. Create a page navigation component named "secondnavigation".
2. Select both Jump to page (page input box) and Page size (page size selection).
3. Define these setting in the Jump to page (page input box) section:
   - Field label: Go to page;
   - Field size: 3
4. Define these setting in the Page size control section:
   - Field label: Number of items to display:
   - Page sizes:
     10 | 10
     50 | 50
     0  | All
5. Enter the following text in these element design fields:

Table 63. Second page navigation element design

<table>
<thead>
<tr>
<th>Design element</th>
<th>Design code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Header</td>
<td>&lt;span&gt;</td>
</tr>
<tr>
<td>Footer</td>
<td>&lt;/span&gt;</td>
</tr>
<tr>
<td>Separator</td>
<td>&lt;/span&gt;</td>
</tr>
</tbody>
</table>

Chapter 5. Setting up a site 221
Referencing the page navigation components in another element design

You use component tags to reference both page navigation components in another element design, such as a menu:

```html
<div>
  [component name="firstnavigation"]
  <br>
  [component name="secondnavigation"]
</div>
```

**Related tasks:**
- Creating a page information tag

You use the page information tag to display page navigation details in the design of a page navigation element.

**URL generation using PathCmpnt and URLCmpnt tags**

There are some special considerations to keep in mind when using URLCmpnt and PathCmpnt tags to create URLs to other web content items from within your content.

**Using the URLCmpnt tag**

The URL component tag URLCmpnt can be used to create URLs to content items in a presentation template. If the attribute `mode="portal"` is set, the content item is displayed through a content viewer portlet.

**Using the PathCmpnt tag**

The path component is used to create the base part of a URL in web content. Typically, the base part is extended by some string that identifies the content to be displayed. If the path component is used inside the context of a web content viewer, the generated URL has the updated URL format and thus cannot be displayed with the traditional web content viewer.

**Adding URL parameters**

Adding URL parameters to a regular portal URL will not affect the web content viewer. For this purpose you must use the URLCmpnt or PathCmpnt tag. Additional URL query parameters, like those used with the traditional web content viewer, can be appended to the URLs generated by these tags.

**Personalizing federated documents**

Portal Personalization provides the federated documents feature to retrieve metadata about documents stored in external content management systems or document repositories. Examples of these systems include IBM Lotus Quickr, IBM Content Manager, IBM FileNet Content Manager, and Microsoft Sharepoint. You can use a personalization component in IBM Web Content Manager to display metadata from federated documents and to create links to download or open the documents.

The federated documents feature can acquire metadata from remote systems that support the following methods:
- Document Services remote interfaces, as supported by Lotus Quickr, IBM Content Manager, and FileNet Content Manager
Content Management Interoperability Services 1.0 (CMIS 1.0), as supported by IBM Connections, IBM Content Manager, IBM FileNet Content Manager, Microsoft Sharepoint, and others.

Atom feeds

Creating a federated documents selection rule

Create a selection rule that selects metadata of documents contained in a specific folder of a remote content management system or document repository.

Before you create a rule to access federated documents information, ensure that you configure the feature as described in “Setting up support for federated documents” on page 71.

1. Open the Personalization Navigator.
2. Navigate to the folder where you want to create the rule, or create a folder for the new rule.
3. Click New > Rule.
4. Enter a name for the rule.
5. Optional: Enter a description for the rule to identify the data that the rule selects.
6. Click the link for the Select action in the rule editor (for example, Web Content), and select Federated Documents.
7. Specify the folder on the remote system either by entering the URL directly or by browsing the remote system. In addition to a folder on a remote system, you can enter the URL of any Atom feed available on the network. The Atom data exposed by those feeds is mapped to corresponding AttributeResource tags in IBM Web Content Manager.
   • To enter the URL for a folder directly, complete these steps:
     a. Click value *.
     b. Enter the URL of the folder.
     c. Click Submit.
   • To browse a Document Services server or CMIS server, click the > symbol on the Feed URL condition, and click Select Document Folder to launch the wizard to select a folder.

To connect to a remote server, you identify the server and the authentication method used to access the server. You can either select a predefined server or enter the server URL directly. By default, no predefined servers are configured, but an administrator can add servers to the list. To authenticate with a server, there are several available methods:
   – If single sign-on is configured between the remote server and the portal, you can connect with the current user.
   – You can enter a user ID and password for the remote server.
   – You can select a credential vault slot associated with the server. Credential vault slots are set up by an administrator and enable users to log in without credentials.

If you use a user ID and password or credential vault slot, the remote server must accept authentication requests that use HTTP basic authentication.

For details on how administrators can add servers and configure authentication, see Configuring the federated documents feature.

Note: To browse remote servers and select a folder, the page containing the Personalization Editor requires a theme and module profile that support the
wp_federated_documents_picker theme module. For example, you can use the Portal 8.0 theme and the full or deferred profile. If the page does not use this theme module, the wizard to select a folder is not available when you click Select Document Folder. You can still manually enter a feed URL to a remote folder by clicking value *. To enable the folder selection wizard, ensure that the wp_federated_documents_picker theme module is available to the page containing the Personalization Editor.

8. Optional: Click show all items, and specify the maximum number of entries to be retrieved.

9. Click Save.

You can now use this rule in a personalization component to render the selection result of this rule in web content.

Related concepts:

“Reserved authoring portlet” on page 94
When working with the web content viewer or Web content pages, some scenarios involve web content authoring tasks accomplished with authoring tools components. Such authoring tasks are performed through a special instance of the authoring portlet that is reserved specifically for these tasks and is installed on page that is hidden from the page navigation available to typical users.

The module framework
The module framework allows extensions to contribute to different areas of a page to provide flexibility, enhance the user experience, and maximize performance.

Related tasks:

“Configuring the federated documents feature” on page 72
Configure the federated documents feature to specify information about the source servers for the documents that are available to users.

Using a federated documents rule in a personalization component
To access the information from a federated documents rule in your web content system, create a personalization component, associate it with the rule, and specify the design for displaying document information retrieved by the rule.

1. Open the authoring portlet, and click New > Component > Personalization.
2. Enter a name and description for the new component.
3. Click Search in the Personalization Element section, and select the federated documents selection rule.
4. Define the design for the headers, footers, and individual documents selected by the configured selection rule. Use the AttributeResource element to reference individual metadata fields from the selected documents.
5. Save the component.

You can now preview your component and reference it from within presentation templates, other element designs, or a web content viewers.

Sample designs for a federated documents selection rule:

When rendering document metadata information retrieved with a federated documents selection rule, you can tailor the header, footer, and menu search result designs for simple or more elaborate presentations.
Bulleted list design

To render a simple bulleted list of links to the documents, you can use a design like that described here.

**Header**

```html
<ul>
  <li>
    <a target="_blank" href="[AttributeResource attributeName="contentLink"]">
      [AttributeResource attributeName="title"]
    </a>
  </li>
</ul>
```

**Footer**

```html
</ul>
```

Table list design

To render a complete list of information with a table, you can use a design like that described here.

**Header**

```html
<table>
  <tr>
    <th>Title</th>
    <th>Authors</th>
    <th>Load</th>
    <th>Load via Portal</th>
    <th>Load via Portal with Authentication</th>
    <th>Edit</th>
    <th>Content Type</th>
    <th>Size</th>
    <th>Updated</th>
    <th>Published</th>
  </tr>
</table>
```

**Design for each menu search result**

```html
<tr>
  <td><b>[AttributeResource attributeName="title"]</b></td>
  <td>[AttributeResource attributeName="authors" separator=","]</td>
  <td><a target="_blank" href="[AttributeResource attributeName="rawContentLink"]">download</a></td>
  <td><a target="_blank" href="[AttributeResource attributeName="contentLink"]">download</a></td>
  <td><a target="_blank" href="[AttributeResource attributeName="contentLinkAuthenticated"]">download</a></td>
  <td><a target="_blank" href="[AttributeResource attributeName="viewLink"]">open</a></td>
  <td>[AttributeResource attributeName="contentType"]</td>
  <td>[AttributeResource attributeName="size"]</td>
  <td>[AttributeResource attributeName="updated"]</td>
  <td>[AttributeResource attributeName="published"]</td>
</tr>
```

**Footer**

```html
</table>
```

**AttributeResource values for federated documents:**

The AttributeResource tag is used as a placeholder to display attributes from a federated documents selection rule within a personalization element design. It cannot be used in a presentation template or other element types.
When used with a federated documents selection rule, the following values for the attributeName attribute of the AttributeResource tag are supported for each document in the result set:

**authors**
This attribute displays the names of the authors of the document. The actual result depends on the corresponding attribute mapping that needs to exist at the remote content management system. If no such mapping exists, an empty value is displayed.

**contentLink**
This attribute displays the absolute URL that can be used to download the document. Unlike the rawContentLink attribute, the contentLink attribute contains the URL that addresses the Ajax proxy with parameters used to download the document through the proxy. You can disable proxied URLs by editing the wcm.pzn.ecm.enable.proxy.content.links property in the WCM WCMConfigService configuration service and setting the value to false.

**contentLinkAuthenticated**
This attribute displays the absolute URL that can be used to download the document, including authentication information when needed. Similar to the contentLink attribute, the contentLinkAuthenticated attribute contains the URL that addresses the Ajax proxy with parameters used to download the document through the proxy. However, depending on the personalization selection rule, the contentLinkAuthenticated attribute might also include information about a shared credential vault used to authenticate the user. The credential vault authentication information is available only if the selection rule was created using a credential vault.

**contentType**
This attribute displays the MIME type of the document. If this information is not served by the remote content management system, an empty value is displayed.

**contributors**
This attribute displays the names of the contributors of the document. The actual result depends on the corresponding attribute mapping that needs to exist at the remote content management system. If no such mapping exists, an empty value is displayed.

**id**
This attribute displays the unique ID of the document.

**published**
This attribute indicates the point in time of the first availability of the document. The actual result depends on the corresponding attribute mapping that exists at the remote content management system. If no such mapping exists, an empty value is displayed.

**rawContentLink**
This attribute displays the raw absolute URL that can be used to download the document. Although similar to the contentLink attribute, the rawContentLink attribute contains the URL as it appears in the federated documents feed used by the federated document selection rule. This value does not include any additional proxy addressed in the URL.

**size**
This attribute displays the size of the document in bytes. If this information is not served by the remote content management system, an empty value is displayed.
summary
This attribute displays the summary of the document. If this information is not returned by the remote content management system, an empty value is displayed.

Note: Because the value of the summary attribute can contain very large character data, the amount of data returned by this attribute is limited. You can increase or decrease the amount of data returned by setting the `wcm.pzn.ecm.max.field.length` property in the WCM `WCMConfigService` configuration service.

title
This attribute displays the title of the document. The actual result depends on the corresponding attribute mapping that needs to exist at the remote content management system. If no such mapping exists, the file name is displayed.

updated
This attribute indicates the point in time of the last update to the given document. The actual result depends on the corresponding attribute mapping that exists at the remote content management system. If no such mapping exists, an empty value is displayed.

viewLink
This attribute displays the absolute URL that can be used to open the given document in context of the remote content management user interface. If no such URL is returned by the remote content management system, an empty value is displayed.

Note: The `viewLink` attribute is not supported if you are connected to a IBM Content Manager repository or are using a CMIS server.

Here is an example of how to use these attribute values in a design:

```xml
<li>
  <a target="_blank" href="[AttributeResource attributeName="contentLink"]">
    [AttributeResource attributeName="title"]
  </a>
</li>
```

Inserting a link to remote content

You can insert links to remote content into elements containing a rich text field by using the Insert Link to Remote Document button in the rich text editor. Only remote content that is configured for remote server access can be selected by using this button.

Click Insert Link to Remote Document, and use the wizard panels to select a document.

To connect to a remote server, you identify the server and the authentication method used to access the server. You can either select a predefined server or enter the server URL directly. By default, no predefined servers are configured, but an administrator can add servers to the list. To authenticate with a server, there are several available methods:

- If single sign-on is configured between the remote server and the portal, you can connect with the current user.
- You can enter a user ID and password for the remote server.
- You can select a credential vault slot associated with the server. Credential vault slots are set up by an administrator and enable users to log in without credentials.
If you use a user ID and password or credential vault slot, the remote server must accept authentication requests that use HTTP basic authentication.

If you access the remote server with a user ID and password, a temporary credential vault slot is created to store and manage authentication data. When you close the wizard, the temporary credential vault slot is automatically deleted. However, if you do not close the wizard explicitly, the temporary credential vault slot cannot be deleted. For example, this situation can occur if you close the browser window before completing the wizard. Any temporary credential vault slots that are not used for at least three hours are removed by a cleanup task (com.ibm.portal.cmis.TransientSlotCleanupTask) that runs once a day. An administrator can change the schedule of the cleanup task using the XML configuration interface.

**Theme note:** The **Insert Link to Remote Document** function requires the wp_federated_documents_picker theme module. The page containing the authoring portlet must use a theme that integrates this theme module. In addition, to ensure that the module is loaded, the module profile that is used by the page must include the wp_federated_documents_picker theme module. Both the deferred profile (profiles/profile_deferred.json) and the full profile (profiles/profile_full.json) of the Portal 8.0 theme support the wizard to insert links to remote content.

If the wp_federated_documents_picker theme module cannot be loaded by the authoring portlet, the wizard is not available when you click **Insert Link to Remote Document**. To access the wizard, ensure that the theme and module profile applied to the portal page containing the authoring portlet include the wp_federated_documents_picker theme module. If the feature is disabled when using inline editing of web content, complete these steps:

- Apply a theme to the hidden authoring page that contains the wp_federated_documents_picker theme module. For example, you can apply the Portal 8.0 theme.
- Apply a module profile to the hidden authoring page that contains the wp_federated_documents_picker theme module. For example, you can apply the full profile of the Portal 8.0 theme.
- Apply a skin without decorations to the reserved authoring portlet instance on the hidden authoring page. For example, you can apply the "Portal 8.0 - No Skin" skin.
Related concepts:

"Reserved authoring portlet" on page 94
When working with the web content viewer or Web content pages, some scenarios involve web content authoring tasks accomplished with authoring tools components. Such authoring tasks are performed through a special instance of the authoring portlet that is reserved specifically for these tasks and is installed on page that is hidden from the page navigation available to typical users.

The module framework
The module framework allows extensions to contribute to different areas of a page to provide flexibility, enhance the user experience, and maximize performance.

Related tasks:

"Configuring remote server access for links" on page 70
Before you can add links to files and documents stored in remote content management systems into web content elements, you must configure your server with information about the remote system and the settings used to handle communication with the system.

"Defining rich text options" on page 267
You can configure IBM Web Content Manager to use either the standard rich text editor, an advanced rich text editor, or a third-party rich text editor in rich text fields.

"Configuring the federated documents feature" on page 72
Configure the federated documents feature to specify information about the source servers for the documents that are available to users.

Related information:

Customizing pages
The page customizer contains portlets for editing the layout, content, and appearance of pages. It also provides the Wires portlet, which allows users to set up connections between cooperative portlets on a page, and the Locks portlet, which allows users to lock and unlock containers and container content. You can configure the settings for these portlets to show a certain set of functions, restricting basic users from performing more advanced tasks.

Determining which client side APIs the current portal page provides
At any time during portlet request processing, a portlet can query on the server side which client side APIs are available in the theme of the current page. The set of capabilities combines all capabilities from the theme modules used for the current page and the capabilities set as theme meta data.

Using start and end attributes
The start and end attributes are used to wrap the data returned by a Web Content Manager tag within other tags, such as HTML. These attributes are not mandatory.

The main advantage to using start and end attributes are that the code you enter in the start and end attributes are only rendered when the component tag itself is rendered. For example, If a user does not have access to the content displayed using a component tag, then neither the content nor the start and end code are displayed. This also applies when a component does not exist or does not contain any content.

Example:

In this example, start and end attributes are used to display a set of text components in a bullet list. This could be entered in a presentation template.
The text components would look like this when rendered:
- Component 1
- Component 2
- Component 3
- Component 4

If a user did not have access to component 3, the rendered list would look like this:
- Component 1
- Component 2
- Component 4

In contrast, you could also add this code to a presentation template:

If a user did not have access to component 3, the rendered list would then look like this:
- Component 1
- Component 2
- Component 4

Although the text component is not rendered, the bullet is still rendered.

**Setting parameters to format dates**

These parameters are used to set the format of dates.

*Table 64. Date formatting parameters*

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Meaning</th>
<th>Presentation</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>G</td>
<td>era designator</td>
<td>(Text)</td>
<td>AD</td>
</tr>
<tr>
<td>y</td>
<td>year</td>
<td>(Number)</td>
<td>1996</td>
</tr>
<tr>
<td>M</td>
<td>month in year</td>
<td>(Text and Number)</td>
<td>July and 07</td>
</tr>
<tr>
<td>d</td>
<td>day in month</td>
<td>(Number)</td>
<td>10</td>
</tr>
<tr>
<td>h</td>
<td>hour in am/pm</td>
<td>(Number)</td>
<td>12</td>
</tr>
<tr>
<td>H</td>
<td>hour in day (0-23)</td>
<td>(Number)</td>
<td>0</td>
</tr>
<tr>
<td>m</td>
<td>minute in hour</td>
<td>(Number)</td>
<td>30</td>
</tr>
<tr>
<td>s</td>
<td>second in minute</td>
<td>(Number)</td>
<td>55</td>
</tr>
<tr>
<td>S</td>
<td>millisecond</td>
<td>(Number)</td>
<td>978</td>
</tr>
</tbody>
</table>
Table 64. Date formatting parameters (continued)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Meaning</th>
<th>Presentation</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>E</td>
<td>day in week (Text)</td>
<td>(Text)</td>
<td>Tuesday</td>
</tr>
<tr>
<td>D</td>
<td>day in year (Number)</td>
<td>(Number)</td>
<td>189</td>
</tr>
<tr>
<td>F</td>
<td>day of week in month (Number)</td>
<td>2 (2nd Wed in July)</td>
<td></td>
</tr>
<tr>
<td>w</td>
<td>week in year (Number)</td>
<td>(Number)</td>
<td>27</td>
</tr>
<tr>
<td>W</td>
<td>week in month (Number)</td>
<td>(Number)</td>
<td>2</td>
</tr>
<tr>
<td>a</td>
<td>am/pm marker (Text)</td>
<td>(Text)</td>
<td>PM</td>
</tr>
<tr>
<td>k</td>
<td>hour in day (1-24)</td>
<td>(Number)</td>
<td>24</td>
</tr>
<tr>
<td>K</td>
<td>hour in am/pm (0-11)</td>
<td>(Number)</td>
<td>0</td>
</tr>
<tr>
<td>z</td>
<td>time zone (Text)</td>
<td>(Text)</td>
<td>Pacific Standard Time</td>
</tr>
<tr>
<td>'</td>
<td>escape for text (Delimiter)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;</td>
<td>single quote (Literal)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The number of letters determines format:

Text

- Four or more pattern letters, use full form.
- Less than four, use short or abbreviated form if one exists.

Example: Day/Month/Year

- d,M,y = 3,3,3.
- dd,MM,yy = 03,03,03.
- dd,MMM,yy = 03,Mar,03.

Lowercase and uppercase:

- The case of letters used in date and time code is not consistent. For example, "M" for month but "d" for day and "y" for year.
- Uppercase and lowercase letters can mean different things. For example, "s" for second and "S" for millisecond.

Incorrect format:

If a date or time code is entered incorrectly, nothing is returned.

Other characters:

Any characters in the pattern that are not in the ranges of ['a'..'z'] and ['A'..'Z'] is treated as quoted text. For example, characters like ′, ′, ′, ′, ′, ′, ′, ′, ′ and ′@′ appear in the resulting time text even if they are not embraced within single quotes.

Examples Using the US locale:

Table 65. Example

<table>
<thead>
<tr>
<th>Format Pattern</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;yyyy:MM.dd G 'at' hh:mm:ss z&quot;</td>
<td>1996.07.10 AD at 15:08:56 PDT</td>
</tr>
</tbody>
</table>

Chapter 5. Setting up a site  231
Table 65. Example (continued)

<table>
<thead>
<tr>
<th>Format Pattern</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;EEE, MMM d, &quot;yy&quot;</td>
<td>Wed, July 10, '96</td>
</tr>
<tr>
<td>&quot;h:mm a&quot;</td>
<td>12:08 PM</td>
</tr>
<tr>
<td>&quot;hh 'o'clock' a, zzzz&quot;</td>
<td>12 o'clock PM, Pacific Standard Time</td>
</tr>
<tr>
<td>&quot;K:mm a, z&quot;</td>
<td>00:00 PM, PST</td>
</tr>
<tr>
<td>&quot;yyyyMMdd:MMMM.dd GGG hh:mm aaa&quot;</td>
<td>1996.July:10 AD 12:08 PM</td>
</tr>
</tbody>
</table>

Note on formatting numbers

The Java Number Format Pattern Syntax is used to set the format of numbers.

Table 66. Java Number Format Pattern Syntax

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Meaning.</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Represents a digit. Leading and trailing zeros are shown.</td>
</tr>
<tr>
<td>#</td>
<td>Represents a digit. Leading and trailing zeros are not shown.</td>
</tr>
<tr>
<td>.</td>
<td>A placeholder for decimal separator.</td>
</tr>
<tr>
<td>,</td>
<td>A placeholder for grouping separator.</td>
</tr>
<tr>
<td>E</td>
<td>Used to separate a mantissa and exponent in exponential formats.</td>
</tr>
<tr>
<td>;</td>
<td>Used to separate formats.</td>
</tr>
<tr>
<td>-</td>
<td>The default negative prefix.</td>
</tr>
<tr>
<td>%</td>
<td>Multiplies the number by 100 and displays it as a percentage.</td>
</tr>
<tr>
<td>?</td>
<td>Multiplies the number by 1000 and displays it per mille.</td>
</tr>
<tr>
<td>¢</td>
<td>Displays the number as a currency.</td>
</tr>
<tr>
<td>'</td>
<td>Used to quote special characters in a prefix or suffix.</td>
</tr>
</tbody>
</table>

Examples:

For the number 123456.789.

Table 67. Example

<table>
<thead>
<tr>
<th>Format Pattern</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>####,####,####</td>
<td>123,456.789</td>
</tr>
<tr>
<td>####,##</td>
<td>123456.79</td>
</tr>
<tr>
<td>00000000.0000</td>
<td>000123456.7890</td>
</tr>
</tbody>
</table>

Tags

Tags are used in your markup to reference content stored or generated by elements, or to display metadata from different items.

Web content tags

You use IBM® Web Content Manager tags to reference elements within presentation templates and element designs.
To create a web content tag, click **Insert a Tag** from a presentation template or element design field. The Tag Helper dialog opens. You then select and configure one of the following tag types:

**Alternate design tag**
You use an alternate design tag to display a different component based on whether the item being returned by a menu or navigator is on the current path or not.

**Attribute resource tag**
You use the attribute resource tag to define the information returned by a search query.

**Component tag**
The Component tag is used to reference the content of a component within a presentation template or element design.

**Element tag**
The Element tag is used to reference an element within a presentation template or element design.

**Indent tag**
You use an indent tag to format element designs that require results to be indented.

**Page information tag**
You use the page information tag to display page navigation details in the design of a page navigation element.

**Path component tag**
The path component tag is used to represent certain parts of the URL such as the servlet path, the base path, or the context path of the current page. This tag can be added to presentation templates, element designs, and component designs.

**Placeholder tag**
You use a placeholder tag to display metadata within an element or component design.

**Plug-in tag**
Rendering plug-ins are referenced using the plug-in tag.

**Property tag**
A property tag is used to display various fields and metadata from content items and site areas.

**Style element tag**
The style element tag is used to reference a style sheet component selected as the default style sheet in an authoring template, or a style sheet component referenced within a site area or content item using a component reference.

**URL tag**
The URLCmpnt tag is used to generate a URL to a site area or content item.

**Indenting element designs**
You use an indent tag to format element designs that require results to be indented.

This is the format of a IndentCmpnt tag:
To create an Indent Cmpnt tag:
1. Click **Insert a Tag** from a presentation template or element design field. The **Tag Helper** dialog opens.
2. Select **Indent** as the tag type.
3. Select an offset level. The offset is used to determine how many times the repeat string is used for each indent. The offsets used are based on the number of nodes of the hierarchical content being displayed. For example, a current node depth of 5 and an offset value of -2 would render the repeat string three times. If the sum of the offset and the node depth is negative or 0, the repeat string is not rendered.
4. Click **OK** to add the tag to your design.

Once you have added the tag to your design, you can also add the following parameters to the tag:

<table>
<thead>
<tr>
<th>Table 68. indent tag parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Tag parameter</strong></td>
</tr>
<tr>
<td>repeat=&quot; &quot;</td>
</tr>
<tr>
<td>start=&quot; &quot;</td>
</tr>
<tr>
<td>end=&quot; &quot;</td>
</tr>
</tbody>
</table>

Double-byte character sets:

Not all double-byte character sets support extended ASCII. To use tags such as 
"&nbsp;" you need to replace "&" with "&amp;".

For example:

```
[indentcmpnt offset="0" repeat="&nbsp;&nbsp;&nbsp;"]
```

**Writing links to Web content**

Links to content items can be written as URLs.

**Linking to web content from other web content**

The following examples show how to write links to web content that are to be used with the web content viewer or the Web Content Manager servlet.

To create a link from a piece of web content to another piece of web content, use the following URL format:

```
[URLCmpnt mode="current" context="Selected" type="Content" name="LIBRARY/SITE_AREA_PATH/CONTENT"]
```

- **LIBRARY** = the name of the web content library.
- **SITE_AREA_PATH** = the path to the site area where the content resides.
- **CONTENT** = the name of the content item.

**Linking to web content from an external portlet or web site**

To create a link from an external portlet or web site that displays web content, use the following URL format:
Linking to content displayed in a web content viewer from an external portlet or web site

To create a link from an external portlet or web site to content displayed in a web content viewer, use the following URL format:


- PORTAL_HOST = the name of the Web Content Manager host.
- wps/mypoc = the Web Content Manager context root, specifying the piece of content lookup service.
  - For protected access, use wps/mypoc.
  - For unprotected access, use wps/poc.
- vp_mapping = the virtual portal mapping, if appropriate. For example, wps/mypoc/myvp or wps/poc/myvp.
- LIBRARY = the name of the web content library.
- SITE_AREA_PATH = the path to the site area where the content resides.
- CONTENT = the name of the content item.

To address a specific portal page, use one of the following parameters. The parameters cannot be combined.

- page: Specify the unique name or the object ID of the page.
- mapping: Specify the URL mapping for the page.
- current: Indicates that the current page should be used.

To address a specific presentation template to use to render the web content you use the following parameter:

- pagedesign: Specify the name of the library and presentation template to use including the names of all folders

**Dynamic page lookup:** The page parameter is optional. You can use the link broadcasting feature of the web content viewer to dynamically look up pages by simply omitting the page parameter. For example, if you have a content item called News1, stored in the library Web Content under the site area News, you can create a link to that content item with the following URL:

http://host_name/wps/mypoc?urile=wcm%3Apath%3A/Web+Content/News/News1

Alternatively you can also add a specific portal page using a URL mapping by using the following format:

http://PORTAL_HOST/PORTAL_CONTEXT_ROOT/PORTAL_PAGE_URL_MAPPING/?current=true&urile=wcm%3Apath%3ALIBRARY/SITE_AREA_PATH/CONTENT

- PORTAL_HOST = the name of the portal host
- PORTAL_CONTEXT_ROOT = the portal context root. For anonymous sites use /wps/portal, otherwise use /wps/myportal
• PORTAL_PAGE_URL_MAPPING = the compound name of the portal URL mapping to the portal page that contains the Web Content Manager portlet (URL mappings can be set up using the portal administration portlets).
• LIBRARY = the name of the web content library.
• SITE_AREA_PATH = the path to the site area where the content resides.
• CONTENT = the name of the content item.

Note: The web content viewer on the target page must be configured to receive links from Other portlets and this portlet.

Adding cache parameters to a URL

You can add web content Cache parameters on page 237 and Cache expire parameters on page 238 to a URL to custom caching strategies to individual items. For example:


Adding a last modified parameter to a URL

You can add the last modified date of the current content item to the header of the rendered page. For example:

http://HOST/wps/wcm/connect/LIBRARY/SITE_AREA_PATH/CONTENT>?returnLastModified=true

Contextual linking

Contextual linking is used in systems where content from one site is shared across multiple sites. When content is linked into a site, embedded links (link elements and links in HTML) will reference the site the original content item is located in. Contextual linking is used so that when content is linked from another site, the link will be rendered relative to the current site if possible.

Contextual path linking

Contextual path linking will attempt to resolve a link using a relative path technique. Contextual path linking assumes that each site framework that the linked content is stored in has the same site structure.

Contextual path linking can be applied to elements referenced using the element tag. For example:

- [Element type="content" context="current" key="body" link="path"]

It can only be used if context=current or context=autofill.

When contextual path linking is used a compatible link is searched for using the same relative path. If no link is found, the original link is used.

Using custom caching

You can overrule the default caching parameters of a site by using "cache" and "expire" parameters in URLs and IBM Web Content Manager tags.

Note: Custom caching can only be used when a server's default Web Content cache is set to none or advanced caching. If basic caching is used as your default web Content cache, Custom caching cannot be used.
There are two basic methods in which custom caching can be used with your default server caching settings:

**Default Server Caching Enabled**
In this scenario, some form of default server caching has been enabled. Caching parameters within connect tags and URLs can be used to either:
- Disable caching for the data being requested.
- Apply different caching parameters to the data being requested.

This method is used with sites that are mostly static, but which contain a few dynamic elements that require a different caching strategy from the server’s default caching strategy.

**Default Server Caching Disabled**
In this scenario, default server caching has been disabled. Caching parameters within connect tags and URLs can be used to enable caching for the data being requested.

This scenario is used with sites that contain a large number of elements requiring different caching strategies.

**Cache parameters**
You use "cache" parameters in IBM Web Content Manager tags and URLs to specify whether the retrieved data should be cached or not, and if so, how it should be cached. The "cache" parameter is not mandatory.

Custom caching parameters can only be used when a server’s default Web content cache is set to none or advanced caching. If basic caching is used as your default web Content cache, custom caching cannot be used. Custom caching can be used to set cache parameters for basic, advanced and data caches. When custom caching is used in a connect tag, the caching applies to the data being retrieved via the connect tag. When custom caching is used in a URL request, the caching applies to the entire page being requested.

**Table 69. Values for the CACHE parameter**

<table>
<thead>
<tr>
<th>Basic caching</th>
<th>Advanced Caching</th>
<th>Data caching</th>
</tr>
</thead>
<tbody>
<tr>
<td>CACHE=SITE</td>
<td>CONTENTCACHE=SITE</td>
<td>CONNECTORCACHE=SITE</td>
</tr>
<tr>
<td>CACHE=SESSION</td>
<td>CONTENTCACHE=SESSION</td>
<td>CONNECTORCACHE=SESSION</td>
</tr>
<tr>
<td>CACHE=None</td>
<td>CONTENTCACHE=USER</td>
<td>CONNECTORCACHE=None</td>
</tr>
<tr>
<td></td>
<td>CONTENTCACHE=SECURED</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CONTENTCACHE=PERSONALIZED</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CONTENTCACHE=None</td>
<td></td>
</tr>
</tbody>
</table>

**Examples:**

<CONNECT MOD=Web SRV=HTML ACTION=http://www.ibm.com CACHE=SITE >
http://host:port/wps/wcm/connect/library/sitearea/content?cache=site&contentcache=session

**Custom caching strategies**
- When applying custom caching to static content, you would mostly use CACHE=SITE, CACHE=SESSION or CONTENTCACHE=USER.
- When User Groups are used in implementing site security, you can use the SECURED custom caching strategy: CONTENTCACHE=SECURED.
- When Categories and/or Keywords, along with User Groups, are used for customization of your site, you can use the PERSONALIZED custom caching strategy: CONTENTCACHE=PERSONALIZED.
• If your Server's default web Content Cache is set to Advanced, you must use CONTENTCACHE=NONE to disable caching.

• If retrieving external data you must use CONNECTORCACHE=NONE to disable caching.

**CacheKey parameter**

The CacheKey parameter is used when caching content via the basic cache. A CacheKey is used as a key instead of a URL. This is useful if you have multiple URLs for the same page but only want it cached once. This reduces the amount of memory used by the cache.

**Example:**

The following URLs may use the same web page called news.html.

```html
CACHE=SITE CACHEKEY=news >

CACHE=SITE CACHEKEY=news >

CACHE=SITE CACHEKEY=news >
```

In this example, "news" is used as the CacheKey to store the value of the response from these connect tags. This means that news.html is cached only once instead of being cached three separate times.

**Cache expire parameters**

You use the "expires" parameter in IBM Web Content Manager tags and URLs to specify how long to maintain data in the cache before it is expired. Once data expires from a cache, the next request for the data will be retrieved from the original server. The expires parameter is not mandatory.

Custom expiring parameters can only be used when a server's default Web content cache is set to none or advanced caching. If basic caching is used as your default web content cache, custom expiring cannot be used. Even though you cannot use custom expiring with basic caching enabled, you can still use custom expiring (when using the advanced cache) to expire data in the basic cache.

Values for the expires parameter can represent either a relative period or an absolute date and time:

**Basic cache**

- EXPIRES=ABS [date and time]
- EXPIRES=REL [integer][units]

**Advanced Caching**

- CONTENTCACHEEXPIRES=ABS [date and time]
- CONTENTCACHEEXPIRES=REL [integer][units]

**Data caching**

- CONNECTORCACHEEXPIRES=ABS [date and time]
- CONNECTORCACHEEXPIRES=REL [integer][units]

**Examples:**
Custom expiring strategies

- CONNECTORCACHEEXPIRY= must be used when setting custom expiry parameters when retrieving external data using a connect tag or URL request.
- If your default cache is basic, or if you specify CACHE= in a connect tag or URL request, you must use EXPIRES=
- If your default cache is advanced, or if you specify CONTENTCACHE= in a connect tag or URL request, you must use CONTENTCACHEEXPIRES=
- If your default cache is none, and only CACHE=, or CONTENTCACHE= is specified in a connect tag or URL request, the connect.connector.httpconnector.defaultcacheexpires property in the WCM WCMConfigService service is used to expire the data.

Specifying an absolute time

An absolute date specifies the date and time the document expires.

To indicate a time use the following format:
- ABS [date and time]

For example:
- ABS Mon, 29 May 2000 03:04:18 GMT

A request for this document after this exact time will cause the document to be flushed from the cache and a new copy retrieved.

When specifying an absolute expiry date, the date must be prefixed with ABS, and the date specified must be in one of the following formats:
- Mon, 06 Nov 2000 09:00:00 GMT
- Monday, 06-Nov-00 09:00:00 GMT
- Mon Nov 6 09:00:00 2000.
- 6 Nov 2000 9:00 AM.

The first three date formats are those used in the standard HTTP specification, while the last is a simple, short date format for convenience.

When using absolute times and dates to expire data, cached items remain in the cache until they expire. Once expired, the original item is retrieved on the next request and a copy placed in the cache, but as the absolute time or date has already expired, the item will immediately be expired. Essentially, once expired, an item will not be permanently cached again when using absolute times and dates. All absolute time values are in GMT.

Specifying a relative period

Rather than specifying an absolute time, a relative time can be used to specify that the document will expire some time after the document is placed in the cache, for example a number of hours or days. The actual time the document expires is then calculated from the time the document is retrieved and added to the cache.
Rather than specifying a fixed time for the expiry of cached data, the expiry can be specified relative to the time that the data was added to the cache, for example, a number of hours or days.

To indicate a relative time use the following format:

- REL [integer][units]

**Note:** The space after REL is required.

The integer specifies a whole number of time units. Decimal numbers are not supported. The units are specified by using a single case-insensitive character:

- S: Seconds
- H: Hours
- D: Days
- M: Months

### Table 70. Formatting examples

<table>
<thead>
<tr>
<th>In a connect tag</th>
<th>In a URL Request</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>.EXPIRES=&quot;REL 2M&quot;</code></td>
<td><code>.EXPIRES=REL 2M</code></td>
</tr>
<tr>
<td><code>.EXPIRES=&quot;REL 9000s&quot;</code></td>
<td><code>.EXPIRES=REL 9000s</code></td>
</tr>
</tbody>
</table>

The first example indicates an expiry of two months. The second indicates 9000 seconds (2.5 hours).

By design only seconds, hours, days or months may be specified. Minutes are not supported to simplify the interface (M is used for months). Instead, a multiple of seconds can be used (for example, 300 seconds for 5 minutes).

**Related tasks:**
- Setting service configuration properties

IBM WebSphere Portal comprises a framework of services to accommodate the different scenarios that portals need to address. Services are available for both WebSphere Portal and IBM Web Content Manager. You can configure some of these services.

**Caching IBM Web Content Manager elements**

You can apply caching to elements by using "connect" tags to reference elements within presentation templates instead of the component or element tag.

**Important:**

- See "Enabling connect tags" on page 98 for information on enabling connect tags.
- **Process Connect Tags** must be selected in a presentation template form for connect tags to be processed.

**Example: Applying custom caching**

This is an example of the type of tag that can be used to cache individual elements within a presentation template.

```xml
<connect
    SRV="cmpnt" PATH="/Library/SiteArea/Content"
    SOURCE="library" CMPNTNAME="TestNav" CONTENTCACHE="site" EXPIRES="REL 9000s">
</connect>
```
Table 71. Connect tag details

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>SRV=&quot;cmpnt&quot;</td>
<td>The Service for this Module is &quot;cmpnt&quot;.</td>
</tr>
<tr>
<td>PATH=&quot;/</td>
<td>This sets the context for the element.</td>
</tr>
<tr>
<td>libraryname/</td>
<td>The &quot;sitepath&quot; and &quot;name&quot; placeholders can be used instead of &quot;PATH=&quot; when caching menus or navigators:</td>
</tr>
<tr>
<td>SiteArea/Content&quot;</td>
<td>[placeholder tag=&quot;sitepath&quot;]/[placeholder tag=&quot;name&quot;]</td>
</tr>
<tr>
<td>SOURCE=&quot;library&quot;</td>
<td>Source is either &quot;content&quot;, &quot;sitearea&quot; or &quot;library&quot;. In this example it is &quot;library&quot; because the element we are caching comes from a component.</td>
</tr>
<tr>
<td>CMPNTNAME=&quot;TestNav&quot;</td>
<td>This is the name of the element to be cached.</td>
</tr>
<tr>
<td>CONTENTCACHE=&quot;site&quot;</td>
<td>This is either &quot;site&quot; or &quot;session&quot;.</td>
</tr>
<tr>
<td>EXPIRES=&quot;REL 9000s&quot;</td>
<td>The time the component will expire from the cache is set here.</td>
</tr>
</tbody>
</table>

The first time the presentation template is rendered, the element will be added to the cache. The next time the presentation template is rendered, the element will be displayed from the cache instead of being rendered afresh by the Web Content Manager application. Not until it is expired from the cache will the element be rendered again by the Web Content Manager application. For this reason, only elements that do not require to be freshly rendered every time a page is accessed should be cached.

If you are caching a component that is used in more than one presentation template, it is best practice to save the connect tag as a HTML component and then reference that component in each presentation template. If you then need to change the cached component tags, you only need to change it in the HTML component rather than in multiple presentation templates.

If you have a set of cached components that use the same "Expires" setting, then it is best practice to save the "Expires=" parameter as a HTML component and then reference that component in each connect tag used to cache components. If you then need to change the "Expires=" parameter, you only need to change it in the HTML component rather than in multiple connect tags. This also applies to any common cache tags.

**Example: Disabling caching**

You can also use this method to disable caching. In this example the property `CONTENTCACHE=NONE` is used to disable caching of this element.

```xml
<connect
    SRV="cmpnt" PATH="/SiteArea/Content"
    SOURCE="library" CMPNTNAME="TestNav" CONTENTCACHE="none" />
</connect>
```

**Rendering plug-ins provided with Web Content Manager**

IBM Web Content Manager provides several rendering plug-ins that you can use to enhance your web content.

**Remote action plug-in**

Use the RemoteAction plug-in to reference remote actions from your web content. Remote actions enable you to trigger actions that are typically performed with the authoring portlet, such as creating and editing items and generating views.
You can reference remote actions using plug-in tags with the following format:

[plugin:RemoteAction action=" " docid=" "
useCurrentSelection=" " dialog=" " useCurrentContext=" "]

**action**=
This attribute is the remote action to perform.

**docid**=
This attribute is the document ID of the item to run the remote action against.

**useCurrentContext**=
If set to true, then the document ID is obtained from the rendering context instead of the docid attribute.

**dialog**=
If this attribute is set to true, when rendered within a web content viewer, the remote action is rendered as a URL. The URL redirects the user to a hidden portal page that is used by the viewer for inline editing.

This example demonstrates a "new" action to create a content item:


For details on the remote actions that you can access with plug-in tags, see *Using remote actions*.

**Related concepts:**
[“Using remote actions” on page 627](#)
Remote actions are used to trigger actions from the IBM Web Content Manager application.

**Site analytics data plug-in**
Use the AnalyticsData rendering plug-in tag to inject microformats for Active Site Analytics into your web content.

The microformats gather information about the content items that are rendered on your portal pages. You can insert more than one tag into your content.

Add the AnalyticsData plug-in tag to your content or presentation template design. Each tag can contain only one attribute. Use the following syntax:

[Plugin:AnalyticsData property="" | element="" | value=""
  css-class = ""]

To select a value that is used in the microformat for Active Site Analytics, use one of the following attributes:

**property**
Use this attribute to select one of the supported properties of the web content item. This attribute can take the following values:

- **id** The unique identifier of the content.
- **title** The display title of the content.
- **path** The unique path of the content in your library.
- **lastmodified** The date when the content was last modified.
authors
The author or authors of the content. If the content has more than one author, one microformat tag is written for each author.

element
Use this attribute to select an element from the content. The element must be a text or short text element type. For the value of this attribute, specify the name of the element that you want to be selected.

value
Use this attribute to pass a text value to the plug-in. If you set this attribute, you must also set the CSS class value by using the attribute css-class. See the following description of this attribute for details.

Note: In a single reference to a tag, you can use only one of the previous attributes. If you want to use more than one attribute, add another reference to the tag.

css-class
Use this attribute to control which CSS class is used for the microformat tags. This attribute is mandatory when you specify the value attribute. The attribute is optional when you use the property attribute or the element attribute. If you do not specify this attribute with the property attribute or the element attribute, the CSS class is determined from the value of the property attribute or the element attribute. In this case, the CSS class is generated by prefixing asa.wcm.content_item. and the value of the property attribute or the element attribute.

Examples

[Plugin:AnalyticsData property="title"]
This setting selects the display title of the content item. The following microformat is inserted in the page:
<span style="display:none" class="asa.wcm.content_item.title">Display title</span>

[Plugin:AnalyticsData property="authors" css-class="com.acme.content.authors"]
This setting selects the authors of the content item. If you use this setting and a custom CSS class, the following microformats are inserted in the page:
<span style="display:none" class="com.acme.content.authors">Author 1</span>
<span style="display:none" class="com.acme.content.authors">Author 2</span>

[Plugin:AnalyticsData element="element1"]
This setting selects the text value of the element on the content item. The following microformat is inserted in the page:
<span style="display:none" class="asa.wcm.content_item.element1">Value of element with name element1</span>

[Plugin:AnalyticsData element="element1" css-class="asa.wcm.content_item.path"]
This setting selects the text value of the element on the content item. The following microformat is inserted in the page:
<span style="display:none" class="asa.wcm.content_item.element1">Value of element with name element1</span>

[Plugin:AnalyticsData value="Some text" css-class="asa.wcm.content_item.title"]
This setting uses the text specified by the value attribute. The following microformat is inserted in the page:
<span style="display:none" class="asa.wcm.content_item.title">Some text</span>
Site path plug-in

Use the SitePath plug-in to render the site path of the current content.

You can reference the site path with the following format:

[plugin:SitePath type="rel | abs"]

This plug-in takes the following attribute:

type = rel | abs

This attribute defines whether the path is relative to the current library or not. If you omit this attribute, the value defaults to rel and uses the relative path. The abs value indicates an absolute path to the rendered content.

Examples:

• Absolute path:
  [plugin:SitePath type="abs"]

• Relative path:
  [plugin:SitePath]

Tagging and rating plug-ins for web content

Just as you can tag and rate portal resources like pages and portlets, you can also tag and rate content items generated with IBM Web Content Manager and displayed with the web content viewer. Two plug-in components are available to support the tagging and rating of content items in your web content system. You can add the [Plugin:tags] component and [Plugin:ratings] component in a presentation template to quickly integrate tagging and rating widgets into the current content item.

Related tasks:

"Managing tagging and rating for web content" on page 117

When using tagging and rating with web content, the web content viewer provides additional scope options for the filtering of tagging and rating results. Because changes in the web content system can affect the accuracy of the tagging and rating information used by the portal, it is important to keep the scope information up to date by synchronizing the scopes on a regular basis.

Adding a tagging widget to web content:

You can add a tagging widget to a content item by adding a [Plugin:tags] component to your presentation template. By default the plug-in component is rendered using the TaggingWidgetDesign design, which is included in the web content library Web Resources v70, or you can create your own design.
Note: When using tagging and rating with web content, ensure that synchronization of the tagging and rating scopes has been set up for the portal.

To add a tagging widget to a content item, include the tagging plug-in in your presentation template. For example:

```html
<div id="tags">[Plugin:tags]/</div>
```

The tagging plug-in supports the following parameters that are specified in a key=value format:

**design=path**

The `design` parameter indicates which design to use when rendering the tagging widget. Specify the `path` information using the full library path to the HTML component that contains the design template for the tagging widget. For example:

```html
[Plugin:tags design="Web Content/folder/myTaggingDesign"]
```

If the `design` parameter is not specified, a default design is defined in the TaggingWidgetDesign component, which is provided with the Web Resources v70 library that is included in the product. During rendering the system checks the following locations for the default design:
1. The current web content library
2. The web content library Web Resources v70

**actionScope=scope**

The `actionScope` parameter indicates the scope of tags that you want to show in this widget. For a list of possible values see the description of the `tagScope` parameter used with the inline tag widget. For example:

```html
[Plugin:tags actionScope="personal_private"]
```

**Stylesheet class note:** The presentation template that includes the widget must ensure that the `lotusui30` stylesheet class is assigned to the markup that contains the widget. You can specify this stylesheet class in the following ways:

- In the presentation template, enclose the widget within a `<div>` element that references the class. For example, `<div class="lotusui30"> ... </div>`.
- Create a design for the widget that specifies the class, and reference the design from the `Plugin` tag with the `design` parameter.

**Related concepts:**

The inline tag widget

Users can use the inline tag widget to view tags that have been applied to a resource.

**Related tasks:**

“Synchronizing scopes for web content” on page 118

When users are tagging or rating web content, the web content viewer provides the tagging or rating information to the portal, where it is stored. If information in the web content system changes, this can cause the tagging and rating information stored in the portal to be out of sync. This can happen, for example, if content items are moved or category information changes. To ensure the tagging and rating information is current, synchronize the scopes used for web content. You can set up automatic synchronization according to different conditions or perform a manual synchronization as needed.

**Adding a rating widget to web content:**
You can add a rating widget to a content item by adding a [Plugin:ratings] component to your presentation template. By default the plug-in component is rendered using the RatingWidgetDesign design, which is included in the web content library Web Resources v70, or you can create your own design.

**Note:** When using tagging and rating with web content, ensure that synchronization of the tagging and rating scopes has been set up for the portal.

To add a rating widget to a content item, include the rating plug-in in your presentation template.

For example:

```html
<div id="ratings">[Plugin:ratings]</div>
```

The rating plug-in supports the following parameters that are specified in a **key=value** format:

- **design=path**
  The design parameter indicates which design to use when rendering the rating widget. Specify the path information using the full library path to the HTML component that contains the design template for the rating widget. For example:

```html
[Plugin:ratings design="Web Content/folder/myRatingDesign"]
```

If the design parameter is not specified, a default design is defined in the RatingWidgetDesign component, which is provided with Web Resources v70 library that is included in the product. During rendering the system checks the following locations for the default design:

1. The current web content library
2. The web content library Web Resources v70

- **actionScope=scope**
  The actionScope parameter indicates the scope of tags that you want to show in this widget. For a list of possible values see the description of the ratingScope parameter used with the inline tag widget. For example:

```html
[Plugin:tags actionScope="community"]
```

**Stylesheet class note:** The presentation template that includes the widget must ensure that the lotusui30 stylesheet class is assigned to the markup that contains the widget. You can specify this stylesheet class in the following ways:

- In the presentation template, enclose the widget within a `<div>` element that references the class. For example, `<div class="lotusui30"> ... </div>.
- Create a design for the widget that specifies the class, and reference the design from the Plugin tag with the design parameter.
Related concepts:

The inline rating widget

Users can use the inline rating widget to view ratings that have been applied to a resource.

Related tasks:

“Synchronizing scopes for web content” on page 118

When users are tagging or rating web content, the web content viewer provides the tagging or rating information to the portal, where it is stored. If information in the web content system changes, this can cause the tagging and rating information stored in the portal to be out of sync. This can happen, for example, if content items are moved or category information changes. To ensure the tagging and rating information is current, synchronize the scopes used for web content. You can set up automatic synchronization according to different conditions or perform a manual synchronization as needed.

Utility plug-ins

Utility plug-ins are available to give you more control over how markup is inserted into your web content. For example, you can use comparisons to conditionally determine when to insert markup or you can add comments that can be viewed only by content authors.

Comment plug-in

Use the Comment plug-in to insert comments that you want only content authors to view. All text written between the plug-in start and end tags is treated as comment. This text is not processed, but is omitted when the web content is rendered. Example:

[Plugin:Comment] This is a comment [/Plugin:Comment]

Equals plug-in

Use the Equals plug-in to insert markup into your web content only if the values of two text attributes match. The plug-in provides the two attributes text1 and text2. All markup between the start and end tags is rendered as part of your web content only if the values of these two attributes match.

The following sample renders the markup between the plug-in start and end tags only if the rendered content is currently in a workflow stage named Review.

[Plugin:Equals text1="Review" text2="[Property type='content' context='current' field='currentstage']"]

Document in review.

[/Plugin:Equals]

NotEquals plug-in

Use the NotEquals plug-in to insert markup into your web content only if the values of two text attributes do not match. The plug-in provides the two attributes text1 and text2. All markup between the start and end tags is rendered with your web content only if the values of these two attributes do not match.

The following sample renders the markup between the start and end tags only if the type element of the rendered content does not have the value Internal:

[Plugin:NotEquals text1="Internal" text2="[Element type='content' context='current' key='type']"]

Public document

[/Plugin:NotEquals]
**Locale plug-in**

Use the Locale plug-in to add locale information to your web content or to insert markup into your web content depending on the current locale. This plug-in determines the preferred supported locale in the current context.

The Locale plug-in uses the following attributes:

**acceptLanguage**

Enables you to define one or more preferred locales that can be selected by the portal in a specific context. Attribute values must conform to the syntax specification of the Accept-Language header field of the HTTP/1.1 protocol. For details on this header field, see the Accept-Language section of the RFC2616 specification.

If the `acceptLanguage` attribute is not defined, the portal considers all available supported locales as equally acceptable when determining the preferred supported locale for the selected context.

**pattern**

Enables you to define the output format for the locale that is printed by the Locale plug-in.

Attribute values specify a pattern that can include the following placeholders:

- `{language}`: Placeholder for the lowercase, two-letter language code of a locale as defined by the ISO-639 standard.
- `{country}`: Placeholder for the uppercase, two-letter country code of a locale as defined by the ISO-3166 standard.
- `{variant}`: Placeholder for the vendor-specific or browser-specific variant of a locale.

If the `pattern` attribute is not defined, the following pattern is used: `{language}-{country}-{variant}`.

**printLocale**

Enables you to define whether the preferred supported locale that is determined by the Locale plug-in is inserted into your web content. The attribute can have the following values:

- `true`: The preferred supported locale is written to the markup of the page.
- `false`: The preferred supported locale is not written to the markup of the page.

By combining this attribute with the `acceptLanguage` attribute, users can define content to be rendered only in the following circumstances:

- If a specific locale is applicable in the current context
- If one locale of a specific set of locales is applicable in the current context

Although the body of the Plugin tag is printed if an acceptable locale is supported, the preferred supported locale is not displayed when this attribute is set to `false`.

If the `printLocale` attribute is not defined, a value of `true` is used.
If the Locale plug-in specifies body content, the markup between the start and end
tags is rendered only if the preferred supported locale is determined. If no
supported locale applies in the current context, the Locale plug-in does not insert
markup into content.

Examples:

- When adding the Locale plug-in to your web content without defining
  additional attributes, the preferred supported locale in the current context is
  inserted into your web content. The default format of {language}-{country}-
  {variant} is used (for example, es-ES-WIN):

  [Plugin:Locale]

- The pattern attribute enables you to define the output format of the preferred
  supported locale. The following sample renders the locale using a custom format
  and omits the variant element (for example, es_ES):

  [Plugin:Locale pattern="{language}_{country}"]

- The following sample renders the language code of the preferred supported
  locale in the current context (for example, da). Only languages of Nordic
  countries are accepted as a result. If none of the specified locales is supported in
  the current context, nothing is printed to your web content. For example, if the
  user configures the web browser or the portal user profile to use only the
  English language (en), the Locale plug-in does not return any locale.

  [Plugin:Locale acceptLanguage="da,fo;q=0.8,fi;q=0.6,is;q=0.4,no;q=0.2,sv;q=0.2" pattern="{language}" pattern="{language}"

- You can prevent the preferred supported locale from being rendered with the
  printLocale attribute. This sample inserts the body of the Plugin tag into the
  web content only if the preferred supported locale matches one of the specified
  Nordic languages. The determined locale is not added to the web content.

  [Plugin:Locale acceptLanguage="da,fo;q=0.8,fi;q=0.6,is;q=0.4,no;q=0.2,sv;q=0.2" printLocale="false"
  
  <div>This markup is displayed only if the current context supports a locale
  that represents one of the specified Nordic languages.</div>

  [/Plugin:Locale]

- You can also combine the Locale plug-in with other plug-ins. The following
  example shows how you can use the Matches plug-in with the Locale plug-in.
  The markup in the body of the Matches plug-in is rendered in your content only
  if the preferred supported locale represents a Spanish language.

  Plugin:Matches pattern="es(.*)" text="[Plugin:Locale]"

  <div>This markup appears only if the preferred supported locale in the
  current context represents a Spanish language. Neither the country code
  nor the variant are important as long as the language code of the
  locale is "es".</div>

  [/Plugin:Matches]

Matches plug-in

Use the Matches plug-in to conditionally insert markup into your web content by
evaluating a regular expression against a passed text value. The plug-in provides
two attributes: text and pattern. All markup between the plug-in start and end
tags is rendered only if the value of the text attribute matches the regular
expression in the pattern attribute.

You can also have negative pattern matching. In this case, the markup of the body
of the plug-in tag is rendered only if the text does not match the regular
expression. To use negative pattern matching, set the negative-match attribute to
true.
Note: The regular expression dialect used by the plug-in is defined by the Java language class java.util.regex.Pattern.

Examples:
- The following sample renders the markup between the start and end tags only if the rendered content is currently in a workflow stage whose name contains the word Review.

  ```xml
  [Plugin:Matches text="[Property type='content' context='current'
                  field='currentstage']" pattern=".*Review.*"]
  Document in review
  [/Plugin:Matches]
  ```

  To have the case of the character for the matching ignored, prefix the control sequence (?i) to the regular expression. The following sample also matches workflow stage names such as review:

  ```xml
  [Plugin:Matches text="[Property type='content' context='current'
                  field='currentstage']" pattern="(?i).*review.*"]
  Document in review
  [/Plugin:Matches]
  ```

- The following sample renders the markup between the start and end tags only if the rendered content is currently in a workflow stage whose name does not contain the word Review:

  ```xml
  [Plugin:Matches text="[Property type='content' context='current'
                  field='currentstage']" negative-match="true" pattern=".*Review.*"]
  Document not in review
  [/Plugin:Matches]
  ```

- You can also use capture groups for regular expressions. To access the value of a group, use the MatchedGroup plug-in. You can use this plug-in only between the start and end tags of the Matches plug-in. It provides the attribute group to retrieve a capture group by its index. For example, to access the first word in a list of comma-separated words, use the plug-in as follows:

  ```xml
  [Plugin:Matches pattern="(.*)",",(.*)" text="key1,key2"]
  First key is: [Plugin:MatchedGroup group="1"]
  [/Plugin:Matches]
  ```

**ThemeCapability plug-in**

Use the ThemeCapability plug-in to retrieve the list of available theme capabilities of the page that is currently rendered. These capabilities are described in Basic artifacts and their relation. The plug-in returns the capabilities, in no specific order, as a string of name-value pairs that are separated by commas.

Here is an example of a string that is returned when the [Plugin:ThemeCapability] tag is rendered on a page that uses the full profile theme:

```xml
{analytics_aggregator=8.0, portal.livetext.hcard=8.0, widget_container=2.1, active_site_analytics=8.0, portal.livetext.action=8.0, open_ajax_hub=2.0, dojo=1.7, mashups.enabler=3.0.0.1, content_mapping picker=8.0, cp_tagging_rating=8.0, oneUI=3.0.1, portal.livetext.adr=8.0, mashups.builder=3.0.0.1, federated_documents picker=8.0, portal.livetext.c2a=8.0}
```

You can use the Matches plug-in to check for the availability of specific theme capabilities and then dynamically add markup to the rendered content. The following sample adds a string (... Dojo is available ...) to the markup only if the dojo theme capability is available in the current rendering context:

```xml
[Plugin:Matches text="[Plugin:ThemeCapability]" pattern=".*dojo.*"]
... Dojo is available ...
[/Plugin:Matches]
```
**ToolbarState plug-in**

Use the ToolbarState plug-in to retrieve the state of the site toolbar that is provided with the Portal 8.0 theme. The plug-in renders the following strings to indicate the current state of the toolbar:

- **open** Indicates that the toolbar is expanded as a result of the page being in edit mode.
- **closed** Indicates that the toolbar is collapsed as a result of the page being in view mode.
- **unknown** Indicates that no page mode is selected. This state can occur, for example, directly after log-in.

Example:

```
[Plugin:ToolbarState]
```

You can use the Equals plug-in to check for the current state of the toolbar and then dynamically add markup to the rendered content. The following sample adds a component (web content templates/edit) to the markup only if the site toolbar is currently expanded:

```
[Plugin:Equals text1="open" text2="[Plugin:ToolbarState]"
  [Component name="web content templates/edit"]
[/Plugin:Equals]
```

**Related concepts:**

[Basic artifacts and their relation](#)

The theme modularization framework foresees the following major artifacts and relations to one another.

**Related information:**

[Accept-Language header in RFC2616 specification](#)

---

**Access control**

You can restrict access to selected users and groups to the views within an authoring portlet, the items managed by the authoring portlet, and to elements and pages displayed within a website.

**How access and security levels are set**

There are three levels of access controls for web content

**Library:**

Library level access controls determine access to the library as a whole. If granted, it provides an entry point to the library. A user needs at least contributor access to a library in order to have access to it on the Authoring Portlet.

**Item type per library:**

Item Type access controls define the item type views and tasks a user can access within the authoring portlet for particular library. The permissions set for item types in a library do not automatically give you access to individual items. They only give you access to specific tasks and views within the authoring portlet.
Item level:
Item level access controls define the actions that a user can perform on an individual item. For example, a Manager to the Components type has access to the Purge and Unlock actions but, if that user does not also have Manager access to an individual component then the Purge and Unlock actions will not be enabled when that component is selected.

Users, Groups and Roles
Your content management system will require different types of users. You will need to create a different group for each type of user and then assign those groups different roles within your system.

Web content management roles
You define the access of a user or group for a library to determine who has access to a library, and to define access to the different views within the authoring portlet.

Table 72. Roles

<table>
<thead>
<tr>
<th>Roles</th>
<th>Rendering and authoring portlet access rights</th>
</tr>
</thead>
<tbody>
<tr>
<td>• User</td>
<td>Users and groups assigned to this role can:</td>
</tr>
<tr>
<td></td>
<td>• view items in a website or rendering portlet that they have been assigned at least user access to.</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> The simplest way to assign users to this role is to select any of the default user groups such as &quot;All Authenticated Portal Users&quot; or &quot;Anonymous Portal User&quot;. Users will still require &quot;user&quot; access to an item before it will be rendered in a website or rendering portlet.</td>
</tr>
<tr>
<td>• Contributor</td>
<td>Users and groups assigned to this role can:</td>
</tr>
<tr>
<td></td>
<td>• view items in a rendering portlet or servlet-rendered website that they have been assigned at least user access to.</td>
</tr>
<tr>
<td></td>
<td>• view libraries they have been assigned contributor access to in an authoring portlet.</td>
</tr>
<tr>
<td></td>
<td>• access the &quot;My Items&quot; and &quot;All Items&quot; views in an authoring portlet for libraries that they have been assigned contributor access to.</td>
</tr>
<tr>
<td></td>
<td>• access the item type view within the authoring portlet for item types that they have been assigned at least user access to.</td>
</tr>
</tbody>
</table>
### Table 72. Roles (continued)

<table>
<thead>
<tr>
<th>Roles</th>
<th>Rendering and authoring portlet access rights</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Editor</strong></td>
<td>• view items in a rendering portlet or servlet-rendered website that they have been assigned at least user access to.</td>
</tr>
<tr>
<td></td>
<td>• view libraries the they have been assigned contributor access to in an authoring portlet.</td>
</tr>
<tr>
<td></td>
<td>• access the &quot;My Items&quot; and &quot;All Items&quot; views in an authoring portlet for libraries that they have been assigned at least contributor access to.</td>
</tr>
<tr>
<td></td>
<td>• for library item types that user and groups have been assigned at least editor access to, editors can access the following actions in the authoring portlet:</td>
</tr>
<tr>
<td></td>
<td>– access the item type view</td>
</tr>
<tr>
<td></td>
<td>– create a new item</td>
</tr>
<tr>
<td></td>
<td>– add/remove links</td>
</tr>
<tr>
<td></td>
<td>– apply authoring template</td>
</tr>
<tr>
<td></td>
<td>– copy</td>
</tr>
<tr>
<td></td>
<td>– delete</td>
</tr>
<tr>
<td></td>
<td>– edit</td>
</tr>
<tr>
<td></td>
<td>– link to</td>
</tr>
<tr>
<td></td>
<td>– move</td>
</tr>
<tr>
<td></td>
<td>– restore a version</td>
</tr>
<tr>
<td></td>
<td>– edit version labels</td>
</tr>
<tr>
<td><strong>Manager</strong></td>
<td>Users and groups assigned to these roles can:</td>
</tr>
<tr>
<td></td>
<td>• view items in a rendering portlet or servlet-rendered website that they have been assigned at least user access to.</td>
</tr>
<tr>
<td></td>
<td>• view libraries the they have been assigned contributor access to in an authoring portlet.</td>
</tr>
<tr>
<td></td>
<td>• access the &quot;My Items&quot; and &quot;All Items&quot; views in an authoring portlet for libraries that they have been assigned at least contributor access to.</td>
</tr>
<tr>
<td></td>
<td>• for library item types that they have been assigned manager access to, managers can access the all of the actions available to editors and also the following actions in the authoring portlet:</td>
</tr>
<tr>
<td></td>
<td>– edit access settings</td>
</tr>
<tr>
<td></td>
<td>– next stage</td>
</tr>
<tr>
<td></td>
<td>– purge</td>
</tr>
<tr>
<td></td>
<td>– unlock</td>
</tr>
<tr>
<td></td>
<td>– edit user profile</td>
</tr>
<tr>
<td><strong>Administrator</strong></td>
<td>Users and groups assigned to these roles can:</td>
</tr>
<tr>
<td></td>
<td>• view items in a rendering portlet or servlet-rendered website that they have been assigned at least user access to.</td>
</tr>
<tr>
<td></td>
<td>• view libraries the they have been assigned contributor access to in an authoring portlet.</td>
</tr>
<tr>
<td></td>
<td>• access the &quot;My Items&quot; and &quot;All Items&quot; views in an authoring portlet for libraries that they have been assigned at least contributor access to.</td>
</tr>
<tr>
<td></td>
<td>• all actions in the authoring portlet for library item types that they have been assigned administrator access to.</td>
</tr>
<tr>
<td>Roles</td>
<td>Rendering and authoring portlet access rights</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>--------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Security Administrator</td>
<td>These roles have no access to Web Content Manager items.</td>
</tr>
<tr>
<td>Delegator</td>
<td></td>
</tr>
<tr>
<td>Privileged User</td>
<td></td>
</tr>
</tbody>
</table>

**WebSphere Portal Administrators:**

WebSphere Portal Administrators automatically have Administrator access to all item-types.

**Additive and subtractive methodology:**

You can assign roles to both a whole library, and the item types within a library using either an additive or subtractive methodology.

For example, with an additive methodology, you apply the "All Authenticated Portal Users" to the "Contributor" role to the entire library. This will give "All Authenticated Portal Users" access to the library and any authoring portlets configured to use the library. You then apply Editor, Manager or Administrator roles to specific resource types to grant additional access to specified users or groups.

With a subtractive methodology, you apply the Manager or Administrator role to a user or group to the entire library. You then apply Editor, Contributor or User roles to specific item types and deselect the inheritance check box. This reduces the access to different item types for specified users or groups.

We recommend that propagation from the web content library is enabled because this will simplify administrating library access and because disabling propagation will result in access related errors.

**All Items view:**

A user who is assigned access to an item can always view that item in the All Items view regardless of whether they have access to the related item-type view. For example, if a user does not have access to the presentation template view, but is granted editor access to a presentation template, they can still view, but not edit, the presentation template from the All items view.

**Assigning roles to anonymous or authenticated users**

When accessing a website, users login as either anonymous users, or authenticated portal users.

The following pre-defined groups can be assigned roles in a library.

<table>
<thead>
<tr>
<th>Group</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Anonymous portal user</td>
<td>Select this user to assign a role to anonymous users.</td>
</tr>
</tbody>
</table>
Table 73. pre-defined groups (continued)

<table>
<thead>
<tr>
<th>Group</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>All Authenticated Portal Users</td>
<td>Select this group to assign a role to users that have logged on to your server.</td>
</tr>
<tr>
<td>Users and User Groups</td>
<td>Select this group to assign a role to all users and groups.</td>
</tr>
<tr>
<td>All Portal User Groups</td>
<td>Select this group to assign a role to all groups.</td>
</tr>
</tbody>
</table>

Related tasks:

Defining roles within a library
You can define the access of a user or group for a library to determine who has access to a library, and to define access to the different views within the authoring portlet.

User roles and access
Different users will have a different access to items and functions in your system depending on the role they have been assigned. Roles can be assigned at the library level, and also assigned on individual items.

Assigning access to items
There are two methods used to assign roles to access controls on items:

- Selecting users or groups directly in the access section of an item.
- Allowing assigned roles to be inherited from parent items up to and including the library. Access roles are inherited in the following hierarchies:
  - Library/site area/content item
  - Library/taxonomy/category
  - Library/folder/component
  - Library/folder/authoring template
  - Library/folder/presentation template
  - Library/workflow
  - Library/workflow stage
  - Library/workflow action

You can stop inheritance at any point in an inheritance hierarchy. For example, you could allow inheritance down to a site area, but assign access roles manually for each content item under that site area.

Inheritance from a library is based on the role assigned to the overall library, not on the role assigned to specific item types. For example, you may not have access to the presentation template view on a library, but if you inherit the role of editor to a presentation template, you will be able to view and edit that presentation template from the All Items view.

Inheritance does not apply to draft items.

Note: By default, inheritance is enabled for all roles and items.
Viewing an item's security settings

The following sections are displayed on the security section of each item.

<table>
<thead>
<tr>
<th>Table 74. Security settings</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Section</strong></td>
</tr>
<tr>
<td>User Defined</td>
</tr>
</tbody>
</table>
| Workflow | If an item is participating in a workflow, then the user-defined option does not appear and the workflow settings are displayed. This cannot be edited. Workflow-defined access is set in workflow stages. **Published items and workflow defined item security:**  
  - If you grant a user editor access to an item in a workflow stage that uses a publish action, then those users are able to edit the published item directly. No draft is created. The same is true for administrator defined security when applied to published items.  
  - If you grant a user manager access to an item in a workflow stage that uses a publish action, then those users are able to edit and delete the published item directly. No draft is created. The same is true for administrator defined security when applied to published items.  
  - If you grant a user approve access to an item in a workflow stage that uses a publish action, then those users are able to create drafts of the published item. |
| Administrator Defined | Administrators can edit user access to an item at any time by changing the administrator defined settings. |
| Inheritance | You can also choose to inherit access assigned in the current web content library, or from an item's parent. Inheritance for all user roles are enabled by default. |

How security is set

When a new item is created, the creator is automatically given manager access to the item. Additional user and group security can be added in the user-defined and system defined settings.

If an item is participating in a workflow, the creator is given manager access to the item only in the first workflow stage. As the item progresses through a workflow, the item security is determined by the combined workflow and system defined security.

<table>
<thead>
<tr>
<th>Table 75. Security matrix</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Security level</strong></td>
</tr>
</tbody>
</table>
| User | • User defined  
• Administrator defined  
• Inherited | • Administrator defined  
• Workflow defined | • Administrator defined  
• Workflow defined |
| Contributor | • User defined  
• Administrator defined  
• Inherited | • Administrator defined  
• Workflow defined or inherited | • Administrator defined  
• Workflow defined or inherited |
Deleting items:

When a new item is created, the creator can also delete the item. If an item is participating in a workflow, the creator can only delete the item in the first workflow stage.

**Assigning access to different types of users or groups**

When accessing a website or rendering portlet, users login as either anonymous users, or authenticated portal users.

The following user and groups can be used to grant access to items.

<table>
<thead>
<tr>
<th>User or group</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>anonymous portal user</td>
<td>Select this user to grant access to anonymous users</td>
</tr>
<tr>
<td>[all users]</td>
<td>Select this group to grant access to all users, anonymous and authenticated.</td>
</tr>
<tr>
<td>[all authenticated portal users]</td>
<td>Select this group to grant access to all authenticated users.</td>
</tr>
<tr>
<td>[all portal user groups]</td>
<td>Select this group to grant access to all user groups.</td>
</tr>
<tr>
<td>[creator]</td>
<td>Select this to grant access to the creator of the item.</td>
</tr>
<tr>
<td>[authors]</td>
<td>Select this to grant access to users who have been selected as an &quot;author&quot; of the item.</td>
</tr>
<tr>
<td>[owners]</td>
<td>Select this to grant access to users who have been selected as an &quot;owner&quot; of the item.</td>
</tr>
</tbody>
</table>

**The access required to view a rendered item**

To view an item on a rendered page, you need the following:
1. You need at least user access to the presentation template used to display the current content item.

2. You need at least user access to every item in the path to the current content item:
   - library/site area/content item

3. You need at least user access to every item in the path to any elements or components referenced in the presentation template:
   - library/folder/component
   - library/element
   - library/site area/element
   - library/site area/content item/element
   These paths do not need to be the same as the path to the current content item.

4. There must be a valid template map.

**The "wcm.path.traversal.security" setting:**

Rendered item behavior will vary depending on how you specify the wcm.path.traversal.security property in the WCM WCMConfigService service. If the property is not specified, the default value is false.

If set to false:
- Menus will display content regardless of whether a user has access to all site areas in the content path.
- Navigators will not display site areas a user does not have access to, but can show content under these site areas in specific circumstances such as within breadcrumb navigators.
- URLs are only checked for content access, not site area access.

If set to true:
- Menus and navigators will not display content under secure site areas if the user does not have access to all site areas in the content path.
- Directly accessing content under secure site areas using a URL will fail if the user does not have access to all site areas in the content path.

Rendering performance will be slower if set to true.

**Button access**

You assign item-level access by assigning users and groups different roles for each item. The role you assign determines what actions a user has access to for each item. The following table describes the minimum access required for access to each button in the user interface. If you have enabled inheritance at the library level, the library access level is inherited by item level access by default. For example, giving a user editor access to a library will automatically be applied to new items they create if inheritance is enabled.
<table>
<thead>
<tr>
<th>Actions</th>
<th>Minimum item access</th>
<th>Minimum role access to library resources</th>
<th>Minimum library access</th>
<th>Item status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add or move children</td>
<td>Contributor access or higher.</td>
<td>Editor access or higher to the library resource type.</td>
<td></td>
<td>Contributor access or higher.</td>
</tr>
<tr>
<td>Add or remove child links</td>
<td>Contributor access or higher.</td>
<td>Editor access or higher to the library resource type.</td>
<td></td>
<td>Contributor access or higher.</td>
</tr>
<tr>
<td>Add or remove workflows</td>
<td>Manager access or higher.</td>
<td>When first created, you require manager access to the library resource in any library. Once saved, you require manager access to both the item and library resource in the library the item is stored in.</td>
<td></td>
<td>Contributor access or higher.</td>
</tr>
<tr>
<td>Apply authoring template</td>
<td>Contributor access or higher.</td>
<td>Editor access or higher to the authoring template library resource.</td>
<td></td>
<td>Contributor access or higher.</td>
</tr>
<tr>
<td>Approve</td>
<td>Approver or administrator.</td>
<td>Editor access or higher to the library resource type.</td>
<td></td>
<td>Contributor access or higher.</td>
</tr>
<tr>
<td>Approve Project</td>
<td>Approver.</td>
<td>Not required.</td>
<td></td>
<td>Contributor access or higher.</td>
</tr>
<tr>
<td>Batch-edit access controls</td>
<td>Editor access or higher.</td>
<td>Editor access or higher to the library resource type.</td>
<td></td>
<td>Contributor access or higher.</td>
</tr>
<tr>
<td>Cancel draft</td>
<td>Manager access or higher.</td>
<td>Editor access or higher to the library resource type.</td>
<td></td>
<td>Contributor access or higher.</td>
</tr>
<tr>
<td>Copy</td>
<td>Contributor access or higher.</td>
<td>Editor access or higher to the library resource type.</td>
<td></td>
<td>Contributor access or higher.</td>
</tr>
<tr>
<td>Create draft</td>
<td>Manager access or higher, or approver access.</td>
<td>Editor access or higher to the library resource type.</td>
<td></td>
<td>Only published or expired items.</td>
</tr>
<tr>
<td>Actions</td>
<td>Minimum item access</td>
<td>Minimum role access to library resources</td>
<td>Minimum library access</td>
<td>Item status</td>
</tr>
<tr>
<td>-------------</td>
<td>---------------------</td>
<td>-----------------------------------------</td>
<td>------------------------</td>
<td>--------------------------------</td>
</tr>
<tr>
<td>Delete</td>
<td>Manager access or higher.</td>
<td>Editor access or higher to the library resource type.</td>
<td>Contributor access or higher.</td>
<td></td>
</tr>
<tr>
<td>Edit</td>
<td>Editor access or higher.</td>
<td>Editor access or higher to the library resource type.</td>
<td>Contributor access or higher.</td>
<td></td>
</tr>
<tr>
<td>Link to</td>
<td>Contributor access or higher, or approver access.</td>
<td>Editor access or higher to the library resource type.</td>
<td>Contributor access or higher.</td>
<td></td>
</tr>
<tr>
<td>Manage elements</td>
<td>Editor access or higher.</td>
<td>Editor access or higher to the library resource type.</td>
<td>Contributor access or higher.</td>
<td></td>
</tr>
<tr>
<td>Move</td>
<td>Editor access or higher.</td>
<td>Editor access or higher to the library resource type.</td>
<td>Contributor access or higher.</td>
<td></td>
</tr>
<tr>
<td>Next Stage</td>
<td>Approver access.</td>
<td>Editor access or higher to the library resource type.</td>
<td>Contributor access or higher.</td>
<td></td>
</tr>
<tr>
<td>Preview item and view rendered item</td>
<td>User access or higher, or approver access.</td>
<td>Not required.</td>
<td>Contributor access or higher.</td>
<td></td>
</tr>
<tr>
<td>Process now</td>
<td>Administrator access</td>
<td>Not required.</td>
<td>Contributor access or higher.</td>
<td></td>
</tr>
<tr>
<td>Purge</td>
<td>Manager access or higher.</td>
<td>Not required.</td>
<td>Manager access or higher.</td>
<td></td>
</tr>
<tr>
<td>Read</td>
<td>User access or higher, or approver access.</td>
<td>Not required.</td>
<td>Contributor access or higher.</td>
<td></td>
</tr>
<tr>
<td>Reference</td>
<td>User access or higher, or approver access.</td>
<td>Not required.</td>
<td>Contributor access or higher.</td>
<td></td>
</tr>
<tr>
<td>Reject</td>
<td>Approver or administrator access.</td>
<td>Editor access or higher to the library resource type.</td>
<td>Contributor access or higher.</td>
<td></td>
</tr>
<tr>
<td>Reject Project</td>
<td>Approver.</td>
<td>Not required.</td>
<td>Contributor access or higher.</td>
<td></td>
</tr>
<tr>
<td>Restart workflow</td>
<td>Manager access or higher, or approver access.</td>
<td>Editor access or higher to the library resource type.</td>
<td>Contributor access or higher.</td>
<td>Only published or expired items.</td>
</tr>
</tbody>
</table>
Table 77. Item access controls (continued)

<table>
<thead>
<tr>
<th>Actions</th>
<th>Minimum item access</th>
<th>Minimum role access to library resources</th>
<th>Minimum library access</th>
<th>Item status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Restore</td>
<td>Editor access or higher.</td>
<td>Editor access or higher to the library resource type.</td>
<td>Contributor access or higher.</td>
<td></td>
</tr>
<tr>
<td>Save version</td>
<td>Editor access or higher.</td>
<td>Editor access or higher to the library resource type.</td>
<td>Contributor access or higher.</td>
<td></td>
</tr>
<tr>
<td>Show hidden fields</td>
<td>Administrator access</td>
<td>Not required.</td>
<td>Contributor access or higher.</td>
<td></td>
</tr>
<tr>
<td>Submit for review (Workflows)</td>
<td>Approver access</td>
<td>Editor access or higher to the library resource type.</td>
<td>Contributor access or higher.</td>
<td></td>
</tr>
<tr>
<td>Submit for review (Projects)</td>
<td>Editor access or higher.</td>
<td>Editor access or higher to the library resource type.</td>
<td>Contributor access or higher. Only when a project is in an active state.</td>
<td></td>
</tr>
<tr>
<td>System security</td>
<td>Administrator access</td>
<td>Not required.</td>
<td>Contributor access or higher.</td>
<td></td>
</tr>
<tr>
<td>Unlock</td>
<td>Manager access or higher.</td>
<td>Not required.</td>
<td>Manager access or higher.</td>
<td></td>
</tr>
<tr>
<td>View references</td>
<td>User access or higher, or approver access.</td>
<td>Not required.</td>
<td>Contributor access or higher.</td>
<td></td>
</tr>
<tr>
<td>View versions</td>
<td>User access or higher, or approver access.</td>
<td>Not required.</td>
<td>Contributor access or higher.</td>
<td></td>
</tr>
<tr>
<td>Withdraw approval</td>
<td>Approver.</td>
<td>Not required.</td>
<td>Contributor access or higher. Only when a project is in the review state. Only when Joint Approval is selected.</td>
<td></td>
</tr>
<tr>
<td>Withdraw from review</td>
<td>Approver.</td>
<td>Not required.</td>
<td>Contributor access or higher. Only when a project is in the review state.</td>
<td></td>
</tr>
</tbody>
</table>

Creating new items:

The ability to create new items is set at the library level, not item level. You must have at least contributor access to a library and editor access to an item-type to create a new item. If you have access to create any item type, you can also create folders and projects.

Button access on content items:
You can choose to hide selected buttons on content item forms when creating an authoring template. This means a user may not have access to all buttons on a content item form regardless of their role. Administrators can choose to display hidden buttons if required.

**Profiling versus security:**

Using profiling to personalize a site is different from using security to limit what items a user can access. In a profile based personalized site, although a user may not be able to access all the pages using personalized menus, they may still be able to access other pages by using navigators, or by searching for content. In a secured site, a user can only view items that they have been granted access to.

**Related tasks:**

- Setting service configuration properties

IBM WebSphere Portal comprises a framework of services to accommodate the different scenarios that portals need to address. Services are available for both WebSphere Portal and IBM Web Content Manager. You can configure some of these services.

---

### Authoring interface

The primary role of a web content authoring system is to allow your content creators to author content in the form of content items. There are various features you can use to customize the Web Content Manager user interface to simplify the content authoring process for your content creators.

### Custom portal pages for authoring

You do not have to use the default Web Content Manager page to create content. You can create a new portal page to act as the home page of your authoring system.

You can create separate sub-pages under an authoring home page. For example:

- add an authoring portlet to one sub-page
- add web content viewer portlets to other sub-pages to allow you quickly preview different parts of your website

You can also create pages specifically for different types of users. For example, you can create a separate page for your site designers and content creators. The authoring portlets you add to each page can be configured specifically for each user type.

### Creating new pages

When creating new pages that contain an authoring portlet, add the following metadata parameter to the advanced settings in the page properties:

- Parameter: `resourceaggregation.profile`
- Value: `profiles/profile_full.json`

### Example authoring home page

In this example you have two users groups; site designers and content creators. The website is split between a design library and a content library.
To create a shared authoring environment for both sets of users you would create a parent home page with separate sub-pages for each group, plus a third sub-page which will be used to preview the site:

Table 78. Example authoring sub-pages

<table>
<thead>
<tr>
<th>Site designers page</th>
<th>Content creators page</th>
<th>Header</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Includes an authoring portlet configured to use both the design and content libraries</td>
<td>• Includes an authoring portlet configured to use only the content library</td>
<td>• Includes a web content viewer portlet used to preview the website</td>
</tr>
<tr>
<td>• Only site designers can access this page</td>
<td>• Only content creators can access this page</td>
<td>• Both site designers and content creators can access this page</td>
</tr>
</tbody>
</table>

Related concepts:
“Managed pages” on page 152
Managed pages streamline site management in your portal by simplifying how you create pages and add content. Because page information and content are stored in web content libraries, you can more easily coordinate and publish changes with syndication.

Authoring system access strategies

The roles you assign each library on your authoring system will determine what views and features in an authoring portlet are accessible to your users.

You should only grant each user or group access to roles and item types to match the kind of work they will perform. For example:
• Only assign website designers editor access to authoring templates and presentation templates as they are required to create new authoring templates.
• Assign both website designers and web content authors editor access to components if they are both required to create components.
• Content approvers are only assigned contributor access to content as they are not required to create new content items, but need approver access to content items during a workflow.

Authoring portlet customization

You can edit the shared settings of each authoring portlet on your authoring system to customize each authoring portlet for the people who use them.

For example:
• Select only the libraries that will be used by your users. For example, if you are configuring an authoring portlet to be used only by content creators, only select libraries used to store content items.
• Edit the preview options to best suit your users and the type of website they are creating. You can choose to preview pages in a standard website, a local web content viewer on the same server as the authoring portlet or a remote web content viewer portlet on a different server.
• Customize the look and feel of the authoring portlet by defining various user interface settings. You can use these settings to change some of the default settings of an authoring portlet, or to select a custom launch page to use in place of the default user interface.
• Select an appropriate rich text editor for your users.
Related concepts:

"Authoring portlet settings"

An authoring portlet is used to create and manage web content. You can edit the settings of an authoring portlet from within the Preferences section of the authoring portlet.

**Authoring portlet settings**

An authoring portlet is used to create and manage web content. You can edit the settings of an authoring portlet from within the Preferences section of the authoring portlet.

Use the **Configure** mode to specify settings for all users of all instances of the authoring portlet, regardless of the page on which the portlet instance appears.

Use the **Shared Settings** mode to specify settings for the current instance of an authoring portlet.

**Selecting web content libraries:**

You select which libraries are available to users when using this authoring portlet in the Library Selection section.

1. Select **Show selected libraries** to select the libraries you want to make visible in the authoring portlet.
   a. To add a library, select a library from the list of available libraries, then click **Add**.
   b. To remove a library, select a library from the list of selected libraries, and then click **Remove**.
   c. Use the arrow buttons to change the order of the selected libraries. This determines the order the libraries appear in the authoring portlet.

2. Select **Show new libraries in the library explorer** if you want any newly created libraries to automatically be shown in the library explorer.

3. Alternately, to make all libraries visible in the authoring portlet, select **Show all libraries**. You can then select individual libraries to hide in the authoring portlet.
   a. To hide a library, select a library from the list of available libraries, and then click **Add**.
   b. To remove a library from the list, select a library from the list of selected libraries, and then click **Remove**.

**Configuring or editing shared settings of an authoring portlet:**

Libraries selected using the "configure" view are available on all instances of the authoring portlet, regardless of the page on which the portlet appears. Libraries selected with the "edit shared settings" view are only available for the current instance of an authoring portlet.

The libraries available in the "insert links" and "insert images" dialogues are based on the libraries selected in the "configure" view. If you select a library in the "edit shared settings" view that is not selected in the "configure" view, you are not able to select items from this library when using the "insert links" and "insert images" dialogues.

You can select libraries specifically for the "insert links" and "insert images" dialogues by doing the following:
1. Go to Administration > WebSphere Portal > Portal User Interface > Manage Pages >.

2. Search for the page with the unique name of com.ibm.wps.hiddenpage.wcm.Authoring_Portlet.

3. Edit the page layout.

4. Edit the shared settings of the web Content Authoring portlet.

5. Select the required libraries and click OK.

6. Click Done.

**Syndicated and imported libraries:**

If you syndicate or import a library, it is not automatically added to the list of configured libraries for an authoring portlet on the target server. You need to add the syndicated or imported library to each authoring portlet on each server.

**The Portal Site and Web Content libraries:**

When the authoring portlet is first accessed, both the Portal Site library and Web Content library are displayed by default. When you first edit the list of selected items, the Portal Site will no longer be displayed unless specifically selected. The Web Content library will continue to be displayed until it is removed from the list of selected libraries.

**Defining preview options:**

The preview options determine how content can be previewed. Preview options are defined within the Previewing Options section.

1. Select **Allow authors to preview content in a Web page** to allow users to preview pages using the Web Content Manager servlet.

2. To allow users to preview content in portal pages, you must select specific portal pages from the list located under **Allow authors to preview content in the local portal pages selected**. The portal pages displayed here are the pages available on the same instance of the portal where your Web Content Manager application is installed. The selected pages must contain a web content viewer to display the content.

3. To allow users to preview content in a portlet located on a different portal server, you must enter the URL to the remote portal page in the **Allow authors to preview content using the following URLs** field. The portal pages entered here must contain a web content viewer to display the content.

**Note:** When using a web content viewer to preview content, ensure that the web content viewer is configured to receive links from Other portlets and this portlet. Otherwise the preview will not work.

**Defining user interface options:**

You use the **User Interface Options** section to define the user interface options of an authoring portlet.

1. Select a visibility option for the navigation bar. This is the section of the authoring portlet that displays navigational links to the item views, group by views, and personal views.

   **Show** If this option is selected, the navigation bar is visible when the home page, launch page, or item forms are displayed.
Hide  If this option is selected, the navigation bar is hidden when the home page, launch page, or item forms are displayed. This is the recommended option for users who only need to perform basic web content management tasks.

Hide when home page or launch page is open  If this option is selected, the navigation bar is hidden when the home page or launch page is displayed, but is visible when the item forms are displayed. This means that the initial interface is made simpler, while still having the navigation bar available in other views. This is the recommended option for users who need to perform more complex web content management tasks.

Note: The navigation bar is always displayed when using the library explorer.

2. Select a default view:

Basic Home Page  If selected, the basic home page is displayed when you first access the authoring portlet. The basic home page is designed for content authors who only need to create content items and other simple item types, and who would not usually need access to the more advanced library explorer. The basic home page will allow the creation of content using up to six different authoring templates. The authoring templates displayed are the favorites of the current user, and the most recently used by the current user.

You can define a set of authoring templates to display on the home page when a user does not have any favorites or recently used items. To do this, add the wcm.authoringui.homePageTemplates parameter to the authoring options in the WCM WCMConfigService service using the WebSphere Integrated Solutions Console. The paths of the templates should be specified, separated by colons. For example:

Library 1/Template Name:Library 2/folder/Template Name 2:Library 3/Template Name 3

Home Page  If selected, the standard home page is displayed when you first access the authoring portlet. One section of the home page is used to create items and open their favorite locations. Another section of the home page is used to view recent activity. This interface is designed for content authors who need to complete tasks such as creating content, editing their draft content and approving and declining content, but who would not usually need access to the more advanced library explorer.

Launch page  If you have created a custom launch page to use in place of the default user interface, select **Launch Page**. For example, file.jsp.

A custom launch page is a JSP file and must be stored in the WAR file directory for the Authoring portlet. `was_profile_root/installApps/cellname/PA_WCM_Authoring_UI.ear/ilwwcm-authoring.war/jsp/html`, where `cellname` is unique to your installation. Enter the name of the JSP file in the custom launch page field.

Library Explorer  To use the default user interface, select **Library Explorer**.

3. Select **Hide the open item and view lists** to hide these functions from users in the authoring portlet.
4. To improve performance you can limit the number of tasks a user can open at the same time in the authoring portlet by entering a number in the Maximum open tasks per Authoring Portlet instance field.

5. To improve performance you can limit the number of items a user can select in an index at the same time in the authoring portlet by entering a number in the Maximum selected items per action before warning and Maximum selected items per action before denying the action fields.

6. The number of rows that appear in an index is defined in the Maximum rows per table field.
   - If the value is omitted, this setting defaults to 10.
   - If the value is set then this becomes the default number of items shown per page in the library explorer and other views launched from the view navigator.
   - If the value set is 50 or greater then it will be shown in the quick page size selector within each view.
   - A maximum of 250 rows can be set, but setting the value this high can reduce the performance of your system.
   - This setting does not apply to pop-up dialog and selectors. They always default to 10 items per page.

7. To enable People Awareness, select Enable people awareness. People Awareness allows you to select user names that appear in views and forms within the Authoring Portlet, and send those users an email or Sametime message.

8. Select the default display mode of the library explorer.
   - List view mode:
     This mode only displays lists of items as you browse a library.
   - Tree view mode:
     This mode displays both lists of items plus a navigational tree as you browse a library.

9. Select the default behavior of the "Save" button. This setting only applies to item types that don't require a template, and site areas created using the default site area template.

Defining rich text options:

You can configure IBM Web Content Manager to use either the standard rich text editor, an advanced rich text editor, or a third-party rich text editor in rich text fields.

There are three options available when configuring the default rich text editor for your authoring portlet:

Default
   Select this option to use the default JavaScript editor. This does not require a working Java runtime environment on the client computer.

EditLive! Java Editor
   Select this option to use the EditLive! Java Editor. This requires a working Java runtime environment on the client computer.

Custom
   Selecting Custom allows you to use a third-party rich text editor as your default editor. Before using a compatible third-party rich text editor, you should read the installation and configuration instructions of the
third-party rich text editor. These should include instructions for enabling the third-party rich text editor to be used in a Web Content Manager solution.

When configuring a third-party rich text editor, you need to copy a JSP file supplied by the third-party rich text editor. This file is used to launch the third-party rich text editor. You enter the name of this JSP file in the Rich Text Options section of the authoring portlet configuration.

If the third-party rich text editor is not available the standard rich text editor is used.

Storing JSP Files:

JSP files can be located:

- within the $was_profile_root$installedApps/node-name/wcm.ear/ilwwcm.war directory of your server.
- within the $was_profile_root$installedApps/node-name/wcm.ear/PA_WCMLocalRendering.ear/ilwwcm-localrende.war directory of your server.
- within any other web application running on portal. When referencing JSP files in another web application, use the following path: contextPath;jspPath
  For example: /wps/customapplication;/jsp/editor.jsp

Custom authoring interfaces

You can use the Web Content Manager API and remote action parameters to create customized authoring interfaces specifically for your content creators.

You may not want to use an authoring portlet as the user interface for all your users. In some cases it may be better to create a custom authoring interface using the Web Content Manager API and remote action parameters. For example, you could create a very simple content authoring interface for a specific content authoring team.

Custom launch pages

You can configure an authoring portlet to use a launch page of your own design instead of the default user interface. A custom launch page can either be a JSP or HTML file. You use remote actions to call different views and functions from with the authoring portlet's user interface. You can also use the web content API to add other functions to your launch page. Once you have created a custom launch page, you then configure your authoring portlet to use the custom launch page instead of the default authoring portlet user interface.

Remote actions

Remote actions are used in the query string of a URL to trigger actions from the Web Content Manager application. You can use remote actions to add standard Web Content Manager functions to a custom user interface.
Related concepts:

“Managed pages” on page 152
Managed pages streamline site management in your portal by simplifying how you create pages and add content. Because page information and content are stored in web content libraries, you can more easily coordinate and publish changes with syndication.

“Creating a custom launch page” on page 639
You can configure an authoring portlet to use a launch page of your own design instead of the default user interface.

“Using remote actions” on page 627
Remote actions are used to trigger actions from the IBM Web Content Manager application.

“The IBM Web Content Manager API” on page 617
You can use the Web Content Manager API to extend functions of Web Content Manager.

Related tasks:

Managing pages
Manage Pages allows you to create, edit, activate, order, and delete pages as well as external Web pages and labels. Available tasks depend on which item is selected. Each page can contain multiple pages. All pages on which you have the User or greater role are displayed in a navigation menu. You must expand pages to access nested pages. The options that you see are dependent upon your access level.

Creating pages using the Manage Pages portlet
A page displays content, such as portlets and other pages, in a single area. By creating pages, you can organize your information and add new navigational elements to the site.

Web content inline editing strategies
An inline editing system is used to deliver editable web sites such as an intranet or a wiki. It combines the features of both an authoring system and a delivery system.

Using an authoring tools element
The authoring tool element is used to add authoring portlet functions to web pages. When creating an authoring tool element, you need to define the layout of the authoring tool and any required actions, and select parameters for each action layout as required.

Related concepts:

“Authoring tools element”
The authoring tool element is used to add authoring portlet functions to web pages.

Authoring tools element:
The authoring tool element is used to add authoring portlet functions to web pages.

You can add the following authoring portlet functions to a web page:
• Create a new content item.
• Perform inline editing of a content item displayed in a web page.
• Delete the content item displayed in a web page.
• Approve or reject the current content being previewed. These buttons are only visible to approvers when previewing a draft item, or by opening the URL sent by an email action during a workflow.

Authoring tools can be referenced within presentation templates, menu element designs and navigator element designs. When added to menus and navigators, the edit, delete and approve functions are applied to each item displayed in a menu or navigator.

Creating an authoring tool element

You can only use an authoring tool element by creating an authoring tool component. You cannot add an authoring tool element to authoring templates, site areas or content items.

Using an authoring tool

When previewing content, users with access to an authoring tool are able to execute various authoring portlet functions.

Note: Care should be taken when previewing content items that use a presentation template that includes an authoring tool. Some functions are active and ready to use while other functions may not work as normal.

The authoring tool will also be visible when viewing a published site.

Using an authoring tool on multiple servers

When using an authoring tool on more than one server, you use two-way syndication to keep each server being used to author content synchronized. This may lead to the occasional “save” conflict where an item updated on one server will be overwritten with changes to the same item on another server when syndication occurs.

User access to an authoring tool

The authoring tools available to users on a web page are determined by:
• Whether a user has access to the authoring tool component.
• Which tools have been enabled in the authoring tool element.
• The user’s level of access to the content item displayed in a web page.
• Whether a user has access to the authoring portlet. It is recommended that users are assigned at least contributor access to each web content library in a site to ensure they have access to the authoring portlet.
• You only grant “editor” access or higher to users who need to edit the authoring tool.
• You only grant “user” access to users who also have access to the authoring server and who will be using the authoring tool.
• In most cases, users who only have access to the published site would not be granted access to an authoring tool as the tool is designed to be used as an authoring tool on an authoring server, not a published site.
Related tasks:

"Using an authoring tools element” on page 269

The authoring tool element is used to add authoring portlet functions to web pages. When creating an authoring tool element, you need to define the layout of the authoring tool and any required actions, and select parameters for each action layout as required.

Working with authoring tools components in the web content viewer:

When rendering authoring tool components in the Web content viewer, you must account for some changes in the way placeholder tags are specified and in the way users navigate to pages containing authoring tool components. Authoring tasks accessed through a web content viewer, such as inline editing, require the use of an instance of the IBM Web Content Manager authoring portlet that is reserved specifically for such tasks.

Controlling the behavior of authoring tools components:

Authoring tools components rendered in a web content viewer enable you to create, read, edit, delete, approve, or reject content items directly in the web content viewer, instead of requiring you to navigate to the IBM Web Content Manager authoring portlet to perform the same action. The web content viewer either launches a pop-up window that opens from the current page or redirects the user to another portal page that contains the authoring portlet.

You can specify which behavior to use in the authoring tools element design. Typically placeholder tags are used to display authoring tools elements. The value of the format attribute of the placeholder tag determines what kind of URL is created to perform an authoring task:

format="tag"

The placeholder is rendered as a URL that opens a pop-up window containing the authoring portlet.

format="url"

The placeholder is rendered as a URL that redirects the user to another portal page that is used by the web content viewer for inline editing.

Note: Authoring tasks performed in the web content viewer are accomplished through a special instance of the authoring portlet that is reserved specifically for these tasks and is installed on a page that is hidden from the page navigation available to typical users. You can customize the authoring experience for these tasks by configuring the reserved authoring portlet and the page used to display it.

Using authoring tools components when launching a pop-up window

When using a pop-up window to perform an authoring task, the pop-up window opens on the portal page and can be moved within the boundaries of the browser window while still showing the portal page. After you complete the task triggered by the authoring tools element, the pop-up window closes automatically, and the portal page refreshes, updating the view in the web content viewer. You can cancel the authoring task by clicking the close icon in the pop-up window's title bar. When cancelling the task, no web content information is saved, unless you explicitly save changes before manually closing the window.
The default value of the format attribute for a placeholder tag is tag, so to use pop-up windows for inline editing, it is not necessary to specify a value for the format attribute. Either of the following design examples creates a URL that opens a pop-up window for authoring tasks:

```html
<Placeholder tag="namelink"/>
<Placeholder tag="namelink" format="tag"/>
```

```html
<a href="<Placeholder tag="href"/>">
  <Placeholder tag="name"/>
</a>
```

```html
<a href="<Placeholder tag="href" format="tag"/>">
  <Placeholder tag="name"/>
</a>
```

**Note:** It is not possible to launch the pop-up window in a separate browser window by adding `target=_blank` to the HTML anchor tag in the design.

**Using authoring tools components when navigating to another page**

Instead of performing tasks from authoring tools elements in a pop-up window on the current page, you can perform authoring tasks by navigating to a hidden portal page that contains a web content viewer containing the reserved authoring portlet. Clicking a link for an authoring tools element automatically redirects you to the other page, but after you complete the authoring task, you must manually navigate back to the original page. If the page with the reserved authoring portlet was opened in a new browser window or tab, you must close the window or tab and manually refresh the original page to see any changes.

To redirect users to another page for authoring tasks, specify a value of `url` for the format attribute in the placeholder tag in the authoring tools element design. Either of the following design examples creates a URL that redirects users to another portal page for authoring tasks:

```html
<Placeholder tag="namelink" format="url"/>
```

```html
<a href="<Placeholder tag="href" format="url"/>">
  <Placeholder tag="name"/>
</a>
```

```html
<a href="<Placeholder tag="href" format="url"/>" target="_blank">
  <Placeholder tag="name"/>
</a>
```

**Note:** You can open the portal page in a separate browser window by adding `target=_blank` to the HTML anchor tag in the design.
Related tasks:

"Configuring the reserved authoring portlet" on page 95

The reserved authoring portlet is essential to the proper operation of web content pages and the web content viewer, so it is important that the configuration of the reserved authoring portlet reflect similar settings for performing authoring tasks as the configuration of other instances of the IBM Web Content Manager authoring portlet.

Setting service configuration properties

IBM WebSphere Portal comprises a framework of services to accommodate the different scenarios that portals need to address. Services are available for both WebSphere Portal and IBM Web Content Manager. You can configure some of these services.

Related reference:

Defining authoring tools

You can format an authoring tool element's look and feel in different ways, including displaying authoring tools as text-based links, or image based links.

Referencing an authoring tool:

An authoring tool component can be referenced within presentation templates, menu element designs, and navigator element designs. When added to menus and navigators, the edit and delete functions are applied to each item displayed in a menu or navigator.

Referencing an authoring tool component in a presentation template

An authoring tool element is referenced in a web content component tag:

```
[Component name="authoringtoolname"]
```

Referencing an authoring tool element in a menu or navigator design

When referencing an authoring tool component in menu or navigator designs you use a component tag with a parameter of compute="always". For example:

```
[Component name="authoringtoolname" compute="always"]
```

Content

IBM WebSphere Portal provides different ways for you to add content to the site. You can add standard markup, such as HTML, portlets, and content from other sources.

IBM Web Content Integrator

The Web Content Integrator is a solution for integrating externally managed Web content with WebSphere Portal. Through the use of standard content syndication feed technologies based on RSS 2.0, the Web Content Integrator provides a loosely-coupled mechanism for transferring published content and metadata to the portal after they have been approved in the source system. Once the content and metadata have been transferred to the portal, it is possible to use the built-in content management features of WebSphere Portal to secure, personalize, and display the content to users.

To use the Web Content Integrator you have to:

1. create a feed on your source system using the Web Content Integrator feed format specification.
2. configure WebSphere Portal to consume the feed.

**Feed format specification**

RSS, or Really Simple Syndication, is an XML-based format that is widely used for syndicating content from sources such as Web sites and blogs to readers who have subscribed to the feeds. The input to the Web Content Integrator is a content feed which complies with the RSS 2.0 format. The core feed format is relatively simple, with only a limited number of elements that need to be specified for each item in the feed. However, the RSS 2.0 specification allows the base format to be extended using XML namespaces to support additional functionality. In order to enable a deeper level of control over how items are created in Web Content Manager, an RSS extension has been defined which contains elements that map to many of the attributes of the Web Content Manager object model.

**Related concepts:**
- Web content feed management

To access a feed you have created on your source server, you must create a feed configuration.

**Feed format overview**

RSS 2.0 is a dialect of XML, and all RSS files must comply with the XML 1.0 specification as published by the World Wide Web Consortium (W3C). RSS feed files typically have file extensions of either .rss or .xml. The Web Content Integrator does not impose any file naming conventions on the feed producer.

**RSS 2.0 feed file format**

The most commonly used media type options are "text/xml" and "application/rss+xml". The choice of media type does impact how the Web Content Integrator is able to determine the character encoding of the feed. If the character encoding cannot be determined correctly, the Web Content Integrator will produce errors when parsing the feed. Therefore it is important to choose an appropriate media type for your environment.

Following the XML prolog, an RSS 2.0 file begins with a single <rss> element. This element has one required attribute, "version", which must be set to "2.0". The file must also contain a single <channel> element which contains a number of sub-elements that provide some metadata about the feed as a whole. The <channel> element must contain one or more <item> elements. The <item> elements in turn contain sub-elements which provide information about the content that is being syndicated. For example:

```xml
<?xml version="1.0"?>
<rss version="2.0">
  <channel>
    <title>Sample Feed</title>
    <link>http://www.ibm.com/feeds/sample.rss</link>
    <description>An example RSS Feed</description>
    <lastBuildDate>Tue, 31 Oct 2006 10:31:00 EST</lastBuildDate>
    <item>
      <title>News Item Two</title>
      <description>This is a summary of the second news article</description>
      <pubDate>Tue, 31 Oct 2006 10:31:00 EST</pubDate>
    </item>
    <item>
      <title>News Item One</title>
      <description>This is a summary of the first news article</description>
      <pubDate>Tue, 31 Oct 2006 10:31:00 EST</pubDate>
    </item>
  </channel>
</rss>
```
Note: If non-ascii data is used in a feed, then encoding="UTF-8" must be specified in the feed: <?xml version="1.0" encoding="UTF-8"?>

Channel-level Elements

Each RSS feed file must contain only one channel element. There are a number of allowable sub-elements of the channel which provide some metadata about the channel itself. The following elements are either required or used by the Web Content Integrator.

**title**  
This element is used to provide the name of the feed. This element is required by the RSS 2.0 specification but is not used by the Web Content Integrator.

**link**  
This element contains a URL that points to the Web page containing the feed. This element is required by the RSS 2.0 specification but is not used by the Web Content Integrator.

**description**  
This element contains a brief description of the content of the channel. This element is required by the RSS 2.0 specification but is not used by the Web Content Integrator.

**lastBuildDate**  
This element contains a date and time stamp representing the last time the content of the feed was changed. This date, as well as any others in the feed, must conform to the RFC 822 format. This is an optional element according to the RSS 2.0 specification, however some feed reader applications may depend on it. In certain cases the Web Content Integrator will store the value in the lastBuildDate element and then pass it back to the feed producer on the next request as a way of indicating which version of the feed it has already syndicated.

Item-level Elements

For the purposes of the Web Content Integrator, each item in the feed represents an item type. The following item types can be created or updated via the feed:

- Content items
- Site Areas
- Taxonomies
- Categories
- Component

The following sub-elements are either required or used by the Web Content Integrator:

**title**  
The value of this element will be stored in the Name field of Web content items. For content items this will become part of the URL to the content page. As this will be used in the Name field of Web content items, the title
can contain only alphanumeric characters (a-z, A-Z, 0-9), spaces, and the following characters: $ - . ! ( ) . This is a required sub-element.

**link** This is the URL to the source content. In some cases it will be used as the base URL from which any relative links embedded in the content are resolved.

**description** The value of this element will be stored in the Description field of Web content items. Although the RSS specification allows entity-encoded or escaped HTML to be placed in this element, the Description field in Web content items is not designed to store HTML. For the purposes of the Web Content Integrator this element must only contain plain text.

**pubDate** The value of this element must be an RFC 822 time and date stamp representing the time that the item was added to, or updated in, the feed. The Web Content Integrator will use this date in combination with the <guid> element to determine whether or not it has already processed the item. Each time an item is updated via the feed the value of the <pubDate> in the feed entry will be updated as well to indicate that something has changed. This is a required sub-element.

**guid** The <guid> element must contain an ID to uniquely identify the item. This will often be a unique ID from the source content management system. The Web Content Integrator will maintain a mapping of this ID to the item’s internal Web Content Manager ID. This is necessary in order to be able to correctly update or delete items that already exist in Web Content Manager. This field is case sensitive and can contain any string of characters up to a maximum 256 characters in length. The isPermaLink attribute will be ignored. This is a required element.

**category** Each <category> element will contain a hierarchical meta data tag that describes the content. The value of this element will be translated into taxonomy and category items in Web Content Manager. If the category tree specified in the <category> element does not already exist in Web Content Manager it will be created automatically by the Web Content Integrator when the feed entry is processed. The RSS 2.0 specification defines an optional domain attribute for the category element. Feed producers can use this attribute to store the name of the Web content library where the category tree is to be created. This element only applies to content items. A single <item> may contain multiple category elements. As this will be used in the Name field of Web content taxonomy and category items, the title can contain only alphanumeric characters (a-z, A-Z, 0-9), spaces, and the following characters: $ - . ! ( ) . This is a required sub-element.

**author** According to the RSS 2.0 specification this element contains the author’s e-mail address. The specification only allows a single <author> element per item. Generally this will be the author of the content item in the source content management system. The Web Content Integrator will attempt to resolve the e-mail address into the common name of a portal user and then store the name of that user in the author field the Web Content Manager item. If this element is not present in the feed, or if the e-mail address cannot be resolved, then the name of the system user will stored in the author field of the Web Content Manager item instead.
**RSS Namespace Extension for Web content**

Web Content Manager items contain a set of controls which are used to store information for various purposes. The elements in this namespace roughly map to the fields that are available on those controls. Depending on the type of the item, it may or may not contain certain controls so some of the elements in the namespace are only relevant to specific item types. All of the elements in this namespace are sub-elements of `<item>`, none are used at the `<channel>` level.

**Adding the custom namespace definition to the feed:**

The first requirement is to add the namespace reference to the `<rss>` element at the beginning of the feed.

The URL of the namespace is specified using the following tag:

```xml
<rss version="2.0" xmlns:ibmwcm="http://purl.org/net/ibmfeedsvc/wcm/1.0" >
```

This is a fixed property value that the feed parser uses as the key to identify elements which belong to the custom Web Content Integrator namespace.

For example:

```xml
<?xml version="1.0"?>
<rss version="2.0" xmlns:ibmwcm="http://purl.org/net/ibmfeedsvc/wcm/1.0" >
  <channel>
    <title>Sample Feed</title>
    <link>http://www.ibm.com/feeds/sample.rss</link>
    <description>Some example RSS Feed</description>
    <lastBuildDate>Tue, 31 Oct 2006 10:31:00 EST</lastBuildDate>
    <item>
      <title>News Item Two</title>
      <description>This is a summary of the second news article</description>
      <pubDate> Tue, 31 Oct 2006 10:31:00 EST</pubDate>
      <ibmwcm:itemType>Content</ibmwcm:itemType>
    </item>
  </channel>
</rss>
```

The namespace label "ibmwcm" can be changed so long as the label specified in the namespace declaration matches the label used on the extended elements in the feed.

**Process Control Elements:**

These elements are used to provide the Web Content Integrator with some information about how to handle the data contained within the `<item>`.

**action**

This element indicates the action to be performed on the item represented by the `<item>`.

<table>
<thead>
<tr>
<th>Element parameters:</th>
<th>Details for this element:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Applies to item types</td>
<td>All</td>
</tr>
<tr>
<td>Required for item types</td>
<td>All</td>
</tr>
</tbody>
</table>

*Table 79. action element*
Table 79. action element (continued)

<table>
<thead>
<tr>
<th>Element parameters:</th>
<th>Details for this element:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Allowable Values</td>
<td>&quot;add&quot;, &quot;update&quot;, or &quot;delete&quot;.</td>
</tr>
<tr>
<td>Required Attributes</td>
<td>None</td>
</tr>
<tr>
<td>Optional Attributes</td>
<td>None</td>
</tr>
<tr>
<td>Required sub-elements</td>
<td>None</td>
</tr>
<tr>
<td>Optional sub-elements</td>
<td>None</td>
</tr>
</tbody>
</table>

Example:
<i>ibmwcm:action>update</ibmwcm:action>

itemType

The itemType element indicates the type of item represented by the feed entry. These correspond to the Web Content Manager item types that can be created or updated via the feed. In some cases the value of this element is combined with the value in the <i>ibmwcm:path> element to determine which Web Content Manager item type to create.

Table 80. itemType element

<table>
<thead>
<tr>
<th>Element parameters:</th>
<th>Details for this element:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Applies to item types</td>
<td>All</td>
</tr>
<tr>
<td>Required for item types</td>
<td>All</td>
</tr>
<tr>
<td>Allowable Values</td>
<td>&quot;siteArea&quot; for site areas.</td>
</tr>
<tr>
<td></td>
<td>&quot;content&quot; for content items.</td>
</tr>
<tr>
<td></td>
<td>&quot;component&quot; for components.</td>
</tr>
<tr>
<td></td>
<td>&quot;category&quot; for taxonomies and categories.</td>
</tr>
<tr>
<td>Required Attributes</td>
<td>None</td>
</tr>
<tr>
<td>Optional Attributes</td>
<td>None</td>
</tr>
<tr>
<td>Required sub-elements</td>
<td>None</td>
</tr>
<tr>
<td>Optional sub-elements</td>
<td>None</td>
</tr>
</tbody>
</table>

Example:
Site Area
<i>ibmwcm:itemType>siteArea</ibmwcm:itemType> + any other value for <ibmwcm:path> or the lack of an <ibmwcm:path> element.

Taxonomy
<i>ibmwcm:itemType>category</ibmwcm:itemType> + <ibmwcm:path>/</ibmwcm:path>

Category
<i>ibmwcm:itemType>category</ibmwcm:itemType> + any other value for <ibmwcm:path> or the lack of an <ibmwcm:path> element.

Component
<i>ibmwcm:itemType>component</ibmwcm:itemType>

Content
<i>ibmwcm:itemType>content</ibmwcm:itemType>

Location Control Elements:
These elements are used to provide the Web Content Integrator with some information about an item’s relative location.

library

You must specify a default Web content library name when configuring a task to consume an input feed. The Web Content Integrator will perform all operations within that library including creating new Web Content Manager items, searching for existing versions of items, and searching for any associated design artifacts such as authoring templates and workflows. You can also override this setting so that a single feed can be used to create content in multiple libraries.

There are three ways that Web Content Integrator will check if a value was specified in this element:

- If the library specified in the library element matches an existing Web content library, all operations during the processing of this feed entry will be processed within the context of the library specified in the library element.
- If no library element is present in the feed entry, or if the library element is present but no value was specified then the default library from the task configuration is used.
- If a value is specified in the library element but no matching Web content library is found, then an error message will be logged and the feed entry will not be processed.

<table>
<thead>
<tr>
<th>Table 81. library element</th>
</tr>
</thead>
<tbody>
<tr>
<td>Element parameters:</td>
</tr>
<tr>
<td>Applies to item types</td>
</tr>
<tr>
<td>Required for item types</td>
</tr>
<tr>
<td>Allowable Values</td>
</tr>
<tr>
<td>Required Attributes</td>
</tr>
<tr>
<td>Optional Attributes</td>
</tr>
<tr>
<td>Required sub-elements</td>
</tr>
<tr>
<td>Optional sub-elements</td>
</tr>
</tbody>
</table>

Example:

<ibmwcm:library>LibraryName</ibmwcm:library>

path

The path element is used to indicate the hierarchical path to the Web content item. For site areas and content items, this element contains the parent site area path. For categories, the value is the parent category path.

Site areas and categories can only have a single path element. If more than one path element is specified, only the first will be used. Content items can have multiple parents, so multiple path elements can be used. The first path element is used as the main content item and the following path elements will be treated as content links. An optional “library” parameter can be specified for path elements referring to linked content located on a different library from the main content item.
New items will be added to the end of the current list of items within Web Content Manager.

Table 82. path element

<table>
<thead>
<tr>
<th>Element parameters:</th>
<th>Details for this element:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Applies to item types</td>
<td>siteArea, Category, Content</td>
</tr>
<tr>
<td>Required for item types</td>
<td>Site Areas and Taxonomies.</td>
</tr>
<tr>
<td>Allowable Values</td>
<td>All values should start with a leading forward slash &quot;/&quot; character. For site area and taxonomy items the value should be just a single forward slash &quot;/&quot;.</td>
</tr>
<tr>
<td>Required Attributes</td>
<td>None</td>
</tr>
<tr>
<td>Optional Attributes</td>
<td>&quot;library&quot;</td>
</tr>
<tr>
<td>Contains the name of a library where the site path resides. This attribute is used where linked content reside on a different library from the main content. This attribute is ignored on the first path element listed in the feed entry.</td>
<td></td>
</tr>
<tr>
<td>Required sub-elements</td>
<td>None</td>
</tr>
<tr>
<td>Optional sub-elements</td>
<td>None</td>
</tr>
</tbody>
</table>

Examples:

<ibmwcm:path>/</ibmwcm:path>
<ibmwcm:path>/IBM/Products</ibmwcm:path>
<ibmwcm:path library="en_US">/Intranet/Home/News</ibmwcm:path>

createLinks

This is used to specify which parent items to link to when creating content items.

Table 83. createLinks element

<table>
<thead>
<tr>
<th>Element parameters:</th>
<th>Details for this element:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Applies to item types</td>
<td>Content</td>
</tr>
<tr>
<td>Required for item types</td>
<td>None</td>
</tr>
<tr>
<td>Allowable Values</td>
<td>The createLinks element is a container for readability and has no attributes or expected values.</td>
</tr>
<tr>
<td>Required Attributes</td>
<td>None</td>
</tr>
<tr>
<td>Optional Attributes</td>
<td>None</td>
</tr>
<tr>
<td>Required sub-elements</td>
<td>parentGuid</td>
</tr>
<tr>
<td>The parentGuid element should contain the unique ID of another item in the feed which describes the parent item.</td>
<td></td>
</tr>
<tr>
<td>Optional sub-elements</td>
<td>None</td>
</tr>
</tbody>
</table>

Examples:
The Orphan Container

If no valid parents or paths are specified in the feed then the item will be placed into an orphan container until it is able to be updated with a valid path. The Web Content Integrator will automatically create an orphans container in each Web content library as it is needed. For site areas and content items, the orphan container will be a site area path names "WCI/Orphans". For categories, the orphan container will be a taxonomy and category path with the name "WCI/Orphans". The name of the orphan containers can be controlled through a setting in the WCMConsumerPlugin.properties file.

children

The children element is used to specify the children of the current item. The value of this element should be the GUID of another entry in the feed which describes a child of the current item.

The referenced child item must be of an appropriate type. For site areas, the children must be site areas or content items. For taxonomies and categories, the children must be categories. If the type is not valid, the parent item will still be added or updated, but the child reference will not be created.

Multiple childGuid sub-elements can be contained within the children element. If multiple children are specified, they will be added in the order in which they are listed in the feed. This allows the feed producer to control the order in which site areas and content items are linked to their parent site area which can be useful for setting up navigation displays.

There are two attributes of the children element that control how the children specified in the feed get combined with any children which may already be contained by the parent item. The action attribute controls whether or not the child list in the feed will replace the existing list of children. If the value of that attribute is set to "add" then the children specified in the feed will be combined with the existing list of children. Any other value, including an empty string or the absence of this attribute, will cause the list of children specified in the feed to completely replace the pre-existing list.

The position attribute is only relevant when the action attribute is set to "add". This attribute controls whether the children specified in the feed are added to the start or to the end of the pre-existing child list. If this attribute is not specified, the children will be added to the end of the pre-existing list. If the children element is not present in the feed then the pre-existing child list remains intact.

Site areas and categories can only have one immediate parent, any pre-existing parent relationships will be removed prior to adding them as a child of this item. As content items can have multiple parents, adding a content item as a child of this item does not remove it from any of its other parents.

Table 84. children element

<table>
<thead>
<tr>
<th>Element parameters:</th>
<th>Details for this element:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Applies to item types</td>
<td>siteArea, Category.</td>
</tr>
</tbody>
</table>

Chapter 5. Setting up a site  281
Table 84. children element (continued)

<table>
<thead>
<tr>
<th>Element parameters:</th>
<th>Details for this element:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Required for item types</td>
<td>None</td>
</tr>
<tr>
<td>Allowable Values</td>
<td>The children element is just a container for readability, it has no expected values.</td>
</tr>
<tr>
<td>Required Attributes</td>
<td>None</td>
</tr>
<tr>
<td>Optional Attributes</td>
<td>action</td>
</tr>
<tr>
<td></td>
<td>position</td>
</tr>
<tr>
<td>Required sub-elements</td>
<td>childGuid</td>
</tr>
<tr>
<td>Optional sub-elements</td>
<td>None</td>
</tr>
</tbody>
</table>

Examples:

```xml
<ibmwcm:children action="add" position="start">
  <ibmwcm:childGuid>8234cb51df43</ibmwcm:childGuid>
</ibmwcm:children>
```

defaultContent

The defaultContent element only applies to site areas. It allows the feed producer to specify which content item will be used as the default content for a site area. If this element is missing, has a blank value, or cannot be resolved, then the default content of the site area will be cleared.

Table 85. defaultContent element

<table>
<thead>
<tr>
<th>Element parameters:</th>
<th>Details for this element:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Applies to item types</td>
<td>siteArea.</td>
</tr>
<tr>
<td>Required for item types</td>
<td>None</td>
</tr>
<tr>
<td>Allowable Values</td>
<td>Value should be the GUID of another item in the feed which describes a content item.</td>
</tr>
<tr>
<td>Required Attributes</td>
<td>None</td>
</tr>
<tr>
<td>Optional Attributes</td>
<td>None</td>
</tr>
<tr>
<td>Required sub-elements</td>
<td>None</td>
</tr>
<tr>
<td>Optional sub-elements</td>
<td>None</td>
</tr>
</tbody>
</table>

Examples:

```xml
<ibmwcm:defaultContent>8234cb5</ibmwcm:defaultContent>
```
**templateMap**

The templateMap element only applies to site areas. It allows the feed producer to specify the template maps to be used when rendering content that is contained within the designated site area. Use multiple instances of the templateMap element to create multiple template mappings on a site area.

When performing an update of an existing site area, the following steps will be executed for each `<ibmwcm:templateMap />` element specified in the feed:

1. Get the names of the authoring and presentation templates specified in the templateMap element.
2. Attempt to locate an authoring template that matches the name specified in the feed.
3. If a matching authoring template item cannot be located, log an error and start processing the next templateMap element.
4. Attempt to locate a presentation template that matches the name specified in the feed.
5. If a matching presentation template cannot be located, log an error and start processing the next templateMap element.
6. Check if the site area already contains a mapping for the specified authoring template:
   a. If so, check whether it maps to the same presentation template as specified in the feed;
      1) If so, go to the next templateMap element.
      2) If not, remove the mapping and replace it with one to the specified presentation template.
   b. If not, create a new mapping of the specified authoring template to the specified presentation template.
7. Process the next templateMap element

The WCM APIs don’t provide a mechanism to get a complete list of template mappings that exist for a given site area. As a result, it is not possible to remove a template mapping via the feed. Once a mapping has been set on a site area it can be updated via the feed but can only be removed manually through the WCM authoring portlet.

<table>
<thead>
<tr>
<th>Table B6. templateMap element</th>
</tr>
</thead>
<tbody>
<tr>
<td>Element parameters:</td>
</tr>
<tr>
<td>Applies to item types</td>
</tr>
<tr>
<td>Required for item types</td>
</tr>
<tr>
<td>Allowable Values</td>
</tr>
</tbody>
</table>
Table 86. templateMap element (continued)

<table>
<thead>
<tr>
<th>Element parameters:</th>
<th>Details for this element:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Required Attributes</td>
<td></td>
</tr>
<tr>
<td>authoring</td>
<td>The value of this attribute must be a string that exactly matches the name of an existing authoring template.</td>
</tr>
<tr>
<td>presentation</td>
<td>The value of this attribute must be a string that exactly matches the name of an existing presentation template.</td>
</tr>
<tr>
<td>Optional Attributes</td>
<td>None</td>
</tr>
<tr>
<td>Required sub-elements</td>
<td>None</td>
</tr>
<tr>
<td>Optional sub-elements</td>
<td>None</td>
</tr>
</tbody>
</table>

Examples:
```
<ibmwcm:templateMap authoring="AT News" presentation="PT News" />
<ibmwcm:templateMap authoring="AT Announcement" presentation="PT Announcement" />
```

Identity control elements:

Most identification fields in Web Content Manager items map directly to core RSS elements; `<title>` maps to the name field, `<description>` maps to the description field, and `<author>` maps to the authors field. Other identification fields can be populated using the following identity control elements.

**displayTitle**

The displayTitle element allows the feed producer to specify a separate value for the Display Title field in Web Content Manager. If this element is not present in the feed entry, Web Content Manager will automatically set the value of the Display Title field to match the Name field when the item is saved.

Table 87. displayTitle element

<table>
<thead>
<tr>
<th>Element parameters:</th>
<th>Details for this element:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Applies to item types</td>
<td>All</td>
</tr>
<tr>
<td>Required for item types</td>
<td>None</td>
</tr>
<tr>
<td>Allowable Values</td>
<td>A string to be used as the Display Title in Web Content Manager. Unlike the Name field, this field can contain double-byte and non-ASCII characters.</td>
</tr>
<tr>
<td>Required Attributes</td>
<td>None</td>
</tr>
<tr>
<td>Optional Attributes</td>
<td>None</td>
</tr>
<tr>
<td>Required sub-elements</td>
<td>None</td>
</tr>
<tr>
<td>Optional sub-elements</td>
<td>None</td>
</tr>
</tbody>
</table>

Example:
owner

The owner element provides a mechanism to set the value of the Owners field via the feed. Multiple owner elements can be specified. Each owner element should contain the common name of a single WebSphere Portal user or group. If the Web Content Integrator is unable to resolve a specified name to an actual user or group then that user or group will not be added to the Owners field but all other processing will continue as normal.

Table 88. owner element

<table>
<thead>
<tr>
<th>Element parameters:</th>
<th>Details for this element:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Applies to item types</td>
<td>All</td>
</tr>
<tr>
<td>Required for item types</td>
<td>None</td>
</tr>
<tr>
<td>Allowable Values</td>
<td>[all users]</td>
</tr>
<tr>
<td></td>
<td>[all authenticated portal users]</td>
</tr>
<tr>
<td></td>
<td>The common name of any valid WebSphere Portal user or group</td>
</tr>
<tr>
<td>Required Attributes</td>
<td>None</td>
</tr>
<tr>
<td>Optional Attributes</td>
<td>None</td>
</tr>
<tr>
<td>Required sub-elements</td>
<td>None</td>
</tr>
<tr>
<td>Optional sub-elements</td>
<td>None</td>
</tr>
</tbody>
</table>

Example:

```xml
<ibmwcm:owner>[all authenticated portal users]</ibmwcm:owner>
<ibmwcm:owner>jsmith</ibmwcm:owner>
```

Profile Control Elements:

Taxonomies and category fields are populated using the RSS <category> element. To populate the Keywords field, a keywords element is required.

keywords

The keywords element should contain a comma-delimited list of metadata tags that describe the content. The list will be stored as-is in the Keywords field of content items.

Table 89. keywords element

<table>
<thead>
<tr>
<th>Element parameters:</th>
<th>Details for this element:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Applies to item types</td>
<td>Content</td>
</tr>
<tr>
<td>Required for item types</td>
<td>None</td>
</tr>
<tr>
<td>Allowable Values</td>
<td>A comma-delimited list of keywords in plain text format.</td>
</tr>
<tr>
<td>Required Attributes</td>
<td>None</td>
</tr>
<tr>
<td>Optional Attributes</td>
<td>None</td>
</tr>
<tr>
<td>Required sub-elements</td>
<td>None</td>
</tr>
<tr>
<td>Optional sub-elements</td>
<td>None</td>
</tr>
</tbody>
</table>
Table 89. keywords element (continued)

<table>
<thead>
<tr>
<th>Element parameters:</th>
<th>Details for this element:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optional sub-elements</td>
<td>None</td>
</tr>
</tbody>
</table>

Example:
<ibmwcm:keywords>software,testing</ibmwcm:keywords>

The authoringTemplate element:

The authoringTemplate element is used to specify the authoring template that will be applied to a content item or site area. The value of this element should be the name of an existing authoring template.

authoringTemplate

If the authoring template is located in a different library from the item, you can specify the library name as well. If no library is specified, the Web Content Integrator will search the library that was specified in the <ibmwcm:library> element. If no <ibmwcm:library> element is present, the Web Content Integrator will search the default library specified in the task configuration.

Note: Once an item has been created, it is not possible to change its authoring template. Therefore, when processing an update, if the authoringTemplate specified in the feed entry does not match the name of the authoring template that was originally used to create the item the Web Content Integrator will generate an error message.

Table 90. authoringTemplate element

<table>
<thead>
<tr>
<th>Element parameters:</th>
<th>Details for this element:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Applies to item types</td>
<td>Content, site area</td>
</tr>
<tr>
<td>Required for item types</td>
<td>None</td>
</tr>
</tbody>
</table>
| Allowable Values | The name of an existing authoring template, or the name of an existing library and authoring template separated by a forward slash. For example:   
• News-AT 
• TmplLib/News-AT |
| Required Attributes | None |
| Optional Attributes | None |
| Required sub-elements | None |
| Optional sub-elements | None |

Example:
<ibmwcm:authoringTemplate>News</ibmwcm:authoringTemplate>
<ibmwcm:authoringTemplate>TmplLib/News</ibmwcm:authoringTemplate>

Element control element:

The element control element is used to populate elements stored in components, site areas and content items.
Element overview

The Web Content Integrator uses the following business logic to process each component element in the feed entry:

1. It checks whether the component site area or content item contains a field whose name matches the value in the feed element's name attribute. This is a case-sensitive comparison so the names must match exactly.
2. If a matching field is found, it checks whether the data type matches what was specified in the feed element's sub-element.
3. If both the name and data type match, the element in the site area or content item will be updated to match the data contained the feed element.
4. If an element is found which matches the feed element's name, but not its data type, the Web Content Integrator will remove the old field from the site area or content item and attempt to replace it with a new field that matches the data type specified in the feed.
5. If no matching field can be found on the content, the Web Content Integrator will attempt to create a new element.

Note: When creating elements in Web Content Manager using an authoring portlet, it is possible to specify a number of field validation criteria such as the maximum size of a text field or the allowable range of a date field. These are validated during the save operation. If the data in the field does not meet the validation criteria the entire save operation will fail, meaning that none of the changes in the feed entry will be applied. The Web Content Integrator does not have the ability to check that the data in the feed element is valid prior to attempting the save operation. Therefore, if you elect to implement validations on your authoring templates, it is important for the feed producer to insure that the content is valid during the generation of the feed.

Text element

Table 91. text element

<table>
<thead>
<tr>
<th>Element parameters:</th>
<th>Details for this element:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Applies to item types</td>
<td>Site areas and content items</td>
</tr>
<tr>
<td>Required for item types</td>
<td>None.</td>
</tr>
<tr>
<td>Allowable Values</td>
<td>A text component should contain a plain text value.</td>
</tr>
<tr>
<td>Required Attributes</td>
<td>name</td>
</tr>
<tr>
<td>Optional Attributes</td>
<td>None</td>
</tr>
<tr>
<td>Required sub-elements</td>
<td>type</td>
</tr>
<tr>
<td>Optional sub-elements</td>
<td>None</td>
</tr>
</tbody>
</table>

Example:
**HTML element**

*Table 92. html element*

<table>
<thead>
<tr>
<th>Element parameters:</th>
<th>Details for this element:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Applies to item types</td>
<td>HTML components, site areas and content items</td>
</tr>
<tr>
<td>Required for item types</td>
<td>HTML components.</td>
</tr>
<tr>
<td>Allowable Values</td>
<td>The value of the value sub-element should be HTML which will be stored in the corresponding HTML element. The HTML can either be entity-encoded or contained within a CDATA element. Alternatively, the feed producer can provide a URL to an HTML file whose contents will be retrieved by the Web Content Integrator.</td>
</tr>
<tr>
<td>Required Attributes</td>
<td>name</td>
</tr>
<tr>
<td>Optional Attributes</td>
<td>None</td>
</tr>
<tr>
<td>Required sub-elements</td>
<td>type</td>
</tr>
<tr>
<td></td>
<td>value</td>
</tr>
<tr>
<td></td>
<td>source</td>
</tr>
<tr>
<td>Optional sub-elements</td>
<td>None</td>
</tr>
</tbody>
</table>

**Examples:**

```xml
<ibmwcm:element name="footer">
  <ibmwcm:type>html</ibmwcm:type>
  <ibmwcm:value>
    &lt;strong&gt;Copyright 2006&lt;/strong&gt;
  </ibmwcm:value>
</ibmwcm:element>
```

```xml
<ibmwcm:element name="footer">
  <ibmwcm:type>html</ibmwcm:type>
  <ibmwcm:value>
    <![CDATA[&lt;strong&gt;Copyright 2006&lt;/strong&gt;]]>
  </ibmwcm:value>
</ibmwcm:element>
```
Rich text element

Table 93. Rich text element

<table>
<thead>
<tr>
<th>Element parameters:</th>
<th>Details for this element:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Applies to item types</td>
<td>Site areas and content items</td>
</tr>
<tr>
<td>Required for item types</td>
<td>None</td>
</tr>
<tr>
<td>Allowable Values</td>
<td>The value of the value sub-element should be HTML which will be stored in the corresponding rich text element. The HTML can either be entity-encoded or contained within a CDATA element.</td>
</tr>
<tr>
<td>Required Attributes</td>
<td>name The value of this attribute should match the name of an existing rich text element in a site area or content item.</td>
</tr>
<tr>
<td>Optional Attributes</td>
<td>None</td>
</tr>
<tr>
<td>Required sub-elements</td>
<td>type The value of this sub-element must be &quot;rich text&quot;. These values are not case sensitive.</td>
</tr>
<tr>
<td></td>
<td>value The value of this sub-element should be HTML markup that is either entity-encoded or contained within a CDATA element.</td>
</tr>
<tr>
<td>Optional sub-elements</td>
<td>None</td>
</tr>
</tbody>
</table>

Examples:

```xml
<ibmwcm:element name="body">
  <ibmwcm:type>rich text</ibmwcm:type>
  <ibmwcm:value>
    &lt;p&gt;This is the content&lt;/p&gt;
  </ibmwcm:value>
</ibmwcm:element>
```

```xml
<ibmwcm:element name="body">
  <ibmwcm:type>rich text</ibmwcm:type>
  <ibmwcm:value>
    <![CDATA[<p>This is the content</p>]]>
  </ibmwcm:value>
</ibmwcm:element>
```

File resource element

Table 94. File resource element

<table>
<thead>
<tr>
<th>Element parameters:</th>
<th>Details for this element:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Applies to item types</td>
<td>File resource components, site areas and content items</td>
</tr>
</tbody>
</table>
### Table 94. File resource element (continued)

<table>
<thead>
<tr>
<th>Element parameters:</th>
<th>Details for this element:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Required for item types</td>
<td>File resource components.</td>
</tr>
<tr>
<td><strong>Allowable Values</strong></td>
<td>The value should be a fully qualified URL that points to the binary file which is to be uploaded into file resource element.</td>
</tr>
<tr>
<td><strong>Required Attributes</strong></td>
<td>name The value of this attribute should match the name of an existing file resource element in a site area or content item.</td>
</tr>
<tr>
<td><strong>Optional Attributes</strong></td>
<td>None</td>
</tr>
<tr>
<td><strong>Required sub-elements</strong></td>
<td>type The value of this sub-element must be &quot;file&quot;. These values are not case sensitive. source The value should be a fully qualified URL that points to the binary file which is to be uploaded into a file resource element. <strong>Note:</strong> If the URL contains non-ascii characters, the non-ascii characters must be encoded. For example: <a href="http://server:port/wps/wcm/%E4%B8%AD%E6%96%87/%E7%BB%84%E4%BB%B6.pdf">http://server:port/wps/wcm/%E4%B8%AD%E6%96%87/%E7%BB%84%E4%BB%B6.pdf</a></td>
</tr>
<tr>
<td><strong>Optional sub-elements</strong></td>
<td>fileName The value of this sub-element should be the file name and extension to be applied to the file attachment when added to the file resource element. If this sub-element is present in the feed entry then its value will be used as the file name of the attachment in Web Content Manager regardless of what the name of the file was on the source server. This is useful if the URL to the binary file does not include the file name as is the case with some content management systems. If the fileName sub-element is not present in the feed then the Web Content Integrator will attempt to determine the file name from the URL in the value sub-element by taking all of the characters following the last forward slash in the URL.</td>
</tr>
</tbody>
</table>

**Examples:**

```xml
<ibmwcm:element name="attachment">
  <ibmwcm:type>file</ibmwcm:type>
  <ibmwcm:source>
    http://sourcecms.yourco.com/files/plan.doc
  </ibmwcm:source>
</ibmwcm:element>
```
Image element

Table 95. Image element

<table>
<thead>
<tr>
<th>Element parameters:</th>
<th>Details for this element:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Applies to item types</td>
<td>Image components, site areas and content items</td>
</tr>
<tr>
<td>Required for item types</td>
<td>Image components.</td>
</tr>
<tr>
<td>Allowable Values</td>
<td>The value should be a fully qualified URL that points to the binary file which is to be uploaded into image element.</td>
</tr>
<tr>
<td>Required Attributes</td>
<td>name</td>
</tr>
<tr>
<td>Optional Attributes</td>
<td>None</td>
</tr>
<tr>
<td>Required sub-elements</td>
<td>type</td>
</tr>
<tr>
<td></td>
<td>source</td>
</tr>
</tbody>
</table>
Table 95. Image element (continued)

<table>
<thead>
<tr>
<th>Element parameters:</th>
<th>Details for this element:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optional sub-elements</td>
<td>fileName</td>
</tr>
</tbody>
</table>

The value of this sub-element should be the file name and extension to be applied to the file attachment when added to the image element. If this sub-element is present in the feed entry then its value will be used as the file name of the attachment in Web Content Manager regardless of what the name of the file was on the source server. This is useful if the URL to the binary file does not include the file name as is the case with some content management systems. If the fileName sub-element is not present in the feed then the Web Content Integrator will attempt to determine the file name from the URL in the value sub-element by taking all of the characters following the last forward slash in the URL.

Examples:

```xml
<ibmwcm:element name="image">
  <ibmwcm:type>image</ibmwcm:type>
  <ibmwcm:source>
    http://sourcecms.yourco.com/images/logo.gif
  </ibmwcm:source>
</ibmwcm:element>

<ibmwcm:element name="image">
  <ibmwcm:type>image</ibmwcm:type>
  <ibmwcm:source>
    http://sourcecms.yourco.com/images/logo.gif
  </ibmwcm:source>
  <ibmwcm:fileName>yourco_logo.doc</ibmwcm:fileName>
</ibmwcm:element>
```

Date element

Table 96. date element

<table>
<thead>
<tr>
<th>Element parameters:</th>
<th>Details for this element:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Applies to item types</td>
<td>Date and time components, site areas and content items</td>
</tr>
<tr>
<td>Required for item types</td>
<td>Date and time components.</td>
</tr>
<tr>
<td>Allowable Values</td>
<td>A date or time value to be stored in a date and time element.</td>
</tr>
<tr>
<td>Required Attributes</td>
<td>name The value of this attribute should match the name of an existing date and time element in a site area or content item.</td>
</tr>
<tr>
<td>Optional Attributes</td>
<td>None</td>
</tr>
</tbody>
</table>
Table 96. date element (continued)

<table>
<thead>
<tr>
<th>Element parameters:</th>
<th>Details for this element:</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Required sub-elements</strong></td>
<td></td>
</tr>
<tr>
<td><strong>type</strong></td>
<td>The value of this sub-element must be &quot;date&quot;. This value is not case sensitive.</td>
</tr>
<tr>
<td><strong>value</strong></td>
<td>The value of this sub-element should be a date and time value in the RFC 822 format.</td>
</tr>
</tbody>
</table>

| **Optional sub-elements** | | |
| **format** | This allows the feed producer to specify the format of the date or time. A value of "date" will cause the date and time element to be set to only display the date portion of the data. Likewise, a value of "time" will only display the time portion of the data. Any other value, or the absence of this sub-element, will result in the both portions of the data being displayed. The values for this sub-element are not case sensitive. |

Example:

```xml
<ibmwcm:element name="StartDate">
  <ibmwcm:type>date</ibmwcm:type>
  <ibmwcm:value>
    Thu, 14 Apr 1966 15:15:00 EDT
  </ibmwcm:value>
  <ibmwcm:format>date</ibmwcm:format>
</ibmwcm:element>
```

Link element

Table 97. link element

<table>
<thead>
<tr>
<th>Element parameters:</th>
<th>Details for this element:</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Applies to item types</strong></td>
<td>Link components, site areas and content items</td>
</tr>
<tr>
<td><strong>Required for item types</strong></td>
<td>Link components.</td>
</tr>
<tr>
<td><strong>Allowable Values</strong></td>
<td>This component contains the information that is required to configure a link element. There are a number of optional sub-elements that can be used to set the various parameters of the Link field.</td>
</tr>
<tr>
<td><strong>Required Attributes</strong></td>
<td>name</td>
</tr>
<tr>
<td><strong>Optional Attributes</strong></td>
<td>None</td>
</tr>
<tr>
<td>Element parameters:</td>
<td>Details for this element:</td>
</tr>
<tr>
<td>---------------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td>Required sub-elements</td>
<td>type</td>
</tr>
</tbody>
</table>
|                      | format | - ExternalLink: creates a link to a URL external to your Web Content Manager system.  
|                      |        | - ManagedContent: creates a link to another Web Content Manager item.  
|                      |        | - ExistingLink: creates a link to a pre-existing link component |
|                      | value | The value of this sub-element depends on what was specified in the "format" sub-element:  
|                      |        | - ExternalLink: a URL.  
|                      |        | - ManagedContent: the GUID of an existing Web Content Manager item.  
<p>|                      |        | - ExistingLink: the GUID of an existing link component. |</p>
<table>
<thead>
<tr>
<th>Element parameters:</th>
<th>Details for this element:</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Optional sub-elements</strong></td>
<td></td>
</tr>
</tbody>
</table>
| **linkText** | This sub-element allows the feed producer to specify what will be rendered between the `<a href="">` and `</a>` portions of the anchor tag. Possible values include:  
  - a plain text string  
  - a string of HTML markup that is either entity-encoded or enclosed in a CDATA tag  
  - the GUID of a feed entry that describes an image component  
A required type attribute indicates which of the previous value types to use. It can be set to "text", "html", or "imageGuid".  
If the linkText sub-element is not included in the feed, the link text default to the value set in either "ExternalLink", "ManagedContent" or "ExistingLink". |
| **linkDescription** | This sub-element provides a mechanism to specify a description for the link element. If this sub-element is not present, the description on the link element will default to the value of the description of the site area or content item. |
| **linkTarget** | The sub-element is used to set the target browser window where the link will be displayed when it is clicked. Allowable values are: "none", "_blank", "_top", "_self", "_parent", and "[NEW_WINDOW_NAME]". If this element is not present in the feed the link target will default to "none" meaning that the link will open in the same browser window as the page containing it. |
| **queryString** | If present, the value of this sub-element will be appended to the generated HREF of the link. This value should be encapsulated in a CDATA tag to prevent parsing problems. |
| **additionalAttributes** | The value of this sub-element will be inserted into the `<a>` tag as additional HTML attributes. |
| **allowClear** | This sub-element controls whether the value in the link element can be deleted. It should be set to either "true" or "false". |
Simple Link to external URL for IBM.com

```xml
<ibmwcm:element name="Link">
  <ibmwcm:type>link</ibmwcm:type>
  <ibmwcm:format>ExternalLink</ibmwcm:format>
  <ibmwcm:linkText type="text">IBM</ibmwcm:linkText>
</ibmwcm:element>
```

Expanded link to external URL for IBM.com

```xml
<ibmwcm:element name="Link">
  <ibmwcm:type>link</ibmwcm:type>
  <ibmwcm:format>ExternalLink</ibmwcm:format>
  <ibmwcm:linkText type="text">RSS Feed Format Resources</ibmwcm:linkText>
  <ibmwcm:linkDescription>Search for RSS Feed Format</ibmwcm:linkDescription>
  <ibmwcm:queryString><![CDATA[?q=rss+feed+format]]></ibmwcm:queryString>
  <ibmwcm:linkTarget>_blank</ibmwcm:linkTarget>
  <ibmwcm:additionalAttributes>class="extLink"</ibmwcm:additionalAttributes>
</ibmwcm:element>
```

Simple link to a file resource component

```xml
<ibmwcm:element name="Link">
  <ibmwcm:type>link</ibmwcm:type>
  <ibmwcm:value>63000001</ibmwcm:value>
  <ibmwcm:format>ManagedContent</ibmwcm:format>
  <ibmwcm:linkText type="imageGuid">290df293e20a</ibmwcm:linkText>
  <ibmwcm:allowClear>true</ibmwcm:allowClear>
</ibmwcm:element>
```

Simple Link to another content item

```xml
<ibmwcm:element name="Link">
  <ibmwcm:type>link</ibmwcm:type>
  <ibmwcm:value>80220102</ibmwcm:value>
  <ibmwcm:format>ManagedContent</ibmwcm:format>
  <ibmwcm:linkText type="html"><![CDATA[<b>Marketing Plan</b>]]></ibmwcm:linkText>
</ibmwcm:element>
```

Number element

**Table 98. number element**

<table>
<thead>
<tr>
<th>Element parameters:</th>
<th>Details for this element:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Applies to item types</td>
<td>Number components, site areas and content items</td>
</tr>
<tr>
<td>Required for item types</td>
<td>Number components.</td>
</tr>
<tr>
<td>Allowable Values</td>
<td>A numeric value to be stored in a number element.</td>
</tr>
<tr>
<td>Required Attributes</td>
<td>name The value of this attribute should match the name of an existing number element in a site area or content item.</td>
</tr>
<tr>
<td>Optional Attributes</td>
<td>None</td>
</tr>
<tr>
<td>Required sub-elements</td>
<td>type The value of this sub-element must be &quot;number&quot;. This value is not case sensitive.</td>
</tr>
<tr>
<td></td>
<td>value A numeric value to be stored in a number element.</td>
</tr>
</tbody>
</table>
Table 98. number element (continued)

<table>
<thead>
<tr>
<th>Element parameters:</th>
<th>Details for this element:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optional sub-elements</td>
<td>format</td>
</tr>
</tbody>
</table>

Example:

```xml
<ibmwcm:element name="FileSize">
  <ibmwcm:type>number</ibmwcm:type>
  <ibmwcm:value>34082</ibmwcm:value>
  <ibmwcm:format>integer</ibmwcm:format>
</ibmwcm:element>
```

Option selection element

Table 99. option selection element

<table>
<thead>
<tr>
<th>Element parameters:</th>
<th>Details for this element:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Applies to item types</td>
<td>Site areas and content items</td>
</tr>
<tr>
<td>Required for item types</td>
<td>None</td>
</tr>
<tr>
<td>Allowable Values</td>
<td>This component contains a list of values that will be set as the selected options in an option selection element.</td>
</tr>
<tr>
<td>Required Attributes</td>
<td>name</td>
</tr>
<tr>
<td>Optional Attributes</td>
<td>None</td>
</tr>
<tr>
<td>Required sub-elements</td>
<td>type</td>
</tr>
</tbody>
</table>
Table 99. option selection element (continued)

<table>
<thead>
<tr>
<th>Element parameters:</th>
<th>Details for this element:</th>
</tr>
</thead>
</table>
| Optional sub-elements | **optionType**<br>This is used to define the type of option selection element.<br>• Specify user when referencing from a list of user defined options.<br>• Specify taxonomy when referencing options from an existing taxonomy. You can only specify one option type per option selection element. If no option type is defined, "user" will be used by default.<br>**selectedCategory**<br>This is used with the option type of "taxonomy" and is used to specify the path to a category you want to use in the option selection element. You can use as many selectedCategory elements as you require.<br>If the taxonomy exists in a different library from the option selection element, you can also specify the library name. For example:<br>\[
\text{<ibmwcm:selectedCategory library="shared">}
\text{Days/Monday</ibmwcm:selectedCategory>}
\]
You must specify the name of each category and taxonomy you want to reference. If they do not exist, they will be created when the feed is processed.<br>**selectedOption**<br>This is used with the option type of "user" and is used to specify a list of user defined options. You can use as many selectedOption elements as you require.|

Example 1:<br>Selecting a single category where the category is in the same library as the item. In this example "Days" is the name of a taxonomy and "Monday" is the name of a category.<br>\[
\text{<ibmwcm:element name="DayOfTheWeek">}
\text{<ibmwcm:type>option</ibmwcm:type>}
\text{<ibmwcm:optionType>taxonomy</ibmwcm:optionType>}
\text{<ibmwcm:selectedCategory>Days/Monday</ibmwcm:selectedCategory>}
\text{</ibmwcm:element>}
\]

Example 2:<br>Selecting multiple categories where the categories are in the same library as the item.
Example 3:
Selecting a single category where the category is in a different library to the item.

```xml
<ibmwcm:element name="DayOfTheWeek">
  <ibmwcm:type>option</ibmwcm:type>
  <ibmwcm:optionType>taxonomy</ibmwcm:optionType>
  <ibmwcm:selectedCategory library="shared">Days/Monday</ibmwcm:selectedCategory>
</ibmwcm:element>
```

Example 4:
Selecting a user defined option. In this example "False" is an option defined by a user on the Authoring Template for an item.

```xml
<ibmwcm:element name="Enable">
  <ibmwcm:type>option</ibmwcm:type>
  <ibmwcm:optionType>user</ibmwcm:optionType>
  <ibmwcm:selectedOption>False</ibmwcm:selectedOption>
</ibmwcm:element>
```

Component reference element

<table>
<thead>
<tr>
<th>Element parameters:</th>
<th>Details for this element:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Applies to item types</td>
<td>Site areas and content items</td>
</tr>
<tr>
<td>Required for item types</td>
<td>None</td>
</tr>
<tr>
<td>Allowable Values</td>
<td>This component contains the GUID of a component.</td>
</tr>
<tr>
<td>Required Attributes</td>
<td>name</td>
</tr>
<tr>
<td>Optional Attributes</td>
<td>None</td>
</tr>
<tr>
<td>Required sub-elements</td>
<td>type</td>
</tr>
<tr>
<td>Optional sub-elements</td>
<td>None</td>
</tr>
</tbody>
</table>

Example:

```xml
<ibmwcm:element name="Footer">
  <ibmwcm:type>reference</ibmwcm:type>
  <ibmwcm:value>29bc2daf3209</ibmwcm:value>
</ibmwcm:element>
```
**User selection element**

*Table 101. user selection element*

<table>
<thead>
<tr>
<th>Element parameters:</th>
<th>Details for this element:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Applies to item types</td>
<td>User selection components, site areas and content items</td>
</tr>
<tr>
<td>Required for item types</td>
<td>User selection components.</td>
</tr>
<tr>
<td>Allowable Values</td>
<td>A list of names of users to be selected in a user selection element.</td>
</tr>
<tr>
<td>Required Attributes</td>
<td>name</td>
</tr>
<tr>
<td>Optional Attributes</td>
<td>None</td>
</tr>
<tr>
<td>Required sub-elements</td>
<td>type</td>
</tr>
<tr>
<td></td>
<td>value</td>
</tr>
<tr>
<td>Optional sub-elements</td>
<td>None</td>
</tr>
</tbody>
</table>

Example:

```xml
<ibmwcm:element name="Users">
  <ibmwcm:type>userSelect</ibmwcm:type>
  <ibmwcm:value>wpsadmin,John Smith</ibmwcm:value>
</ibmwcm:element>
```

**Style sheet element**

*Table 102. style sheet element*

<table>
<thead>
<tr>
<th>Element parameters:</th>
<th>Details for this element:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Applies to item types</td>
<td>Style sheet components</td>
</tr>
<tr>
<td>Required for item types</td>
<td>Style sheet components</td>
</tr>
<tr>
<td>Allowable Values</td>
<td>A URL that points to a CSS file which will be uploaded into a style sheet component.</td>
</tr>
<tr>
<td>Required Attributes</td>
<td>None</td>
</tr>
<tr>
<td>Optional Attributes</td>
<td>None</td>
</tr>
<tr>
<td>Required sub-elements</td>
<td>type</td>
</tr>
<tr>
<td></td>
<td>source</td>
</tr>
<tr>
<td>Optional sub-elements</td>
<td>None</td>
</tr>
</tbody>
</table>
Table 102. style sheet element (continued)

<table>
<thead>
<tr>
<th>Element parameters:</th>
<th>Details for this element:</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Optional sub-elements</strong></td>
<td><strong>format</strong> This sub-element is used to specify the type of the style sheet. Valid options are: &quot;preferred&quot;, &quot;alternate&quot;, and &quot;persistent&quot;. If this sub-element is not present in the feed, or if any value other than the previous values is specified, the type will default to &quot;preferred&quot;.</td>
</tr>
<tr>
<td></td>
<td><strong>mediaType</strong> This sub-element specifies the media type of the style sheet. Valid values are: &quot;all&quot;, &quot;aural&quot;, &quot;braille&quot;, &quot;handheld&quot;, &quot;print&quot;, &quot;projection&quot;, &quot;screen&quot;, &quot;tty&quot;, &quot;tv&quot;, and &quot;unspecified&quot;. If the mediaType sub-element is not present in the feed, or if any value other than the previous values is specified, the media type will default to &quot;unspecified&quot;.</td>
</tr>
<tr>
<td></td>
<td><strong>cssTitle</strong> The value of this sub-element should be a string which will be set as the value of the title attribute when the link to the CSS file is rendered.</td>
</tr>
</tbody>
</table>

Example:

```xml
<ibmwcm:element name="cssFile">
  <ibmwcm:type>styleSheet</ibmwcm:type>
  <ibmwcm:format>alternate</ibmwcm:format>
  <ibmwcm:mediaType>print</ibmwcm:mediaType>
</ibmwcm:element>
```

Workflow control elements:

Workflow elements allow you to specify workflow related parameters when creating content items that use a workflow.

Workflow element

This element specifies the workflow name and stage in which a content item should be created. The value in the name attribute must match that of an existing workflow.

If the workflow is located in a different library from the content item, you can specify the library name as well. If no library is specified, the Web Content Integrator will search the library that was specified in the `<ibmwcm:library>` element. If no `<ibmwcm:library>` element is present, the Web Content Integrator will search the default library specified in the task configuration.
Table 103. workflow element

<table>
<thead>
<tr>
<th>Element parameters:</th>
<th>Details for this element:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Applies to item types</td>
<td>Content items</td>
</tr>
<tr>
<td>Required for item types</td>
<td>None</td>
</tr>
<tr>
<td>Allowable Values</td>
<td>Workflow name and related workflow stage.</td>
</tr>
<tr>
<td>Required Attributes</td>
<td>The name of an existing workflow, or the library and name of an existing library and workflow separated by a forward slash. For example:</td>
</tr>
<tr>
<td></td>
<td>• 3-Stage-Workflow</td>
</tr>
<tr>
<td></td>
<td>• Library1/3-Stage-Workflow</td>
</tr>
<tr>
<td>Optional Attributes</td>
<td>None</td>
</tr>
<tr>
<td>Required sub-elements</td>
<td>workflowStage</td>
</tr>
<tr>
<td></td>
<td>The value of this sub-element should be the name of a workflow stage that is included in the named workflow.</td>
</tr>
<tr>
<td>Optional sub-elements</td>
<td>None</td>
</tr>
</tbody>
</table>

Examples:

```xml
<ibmwcm:workflow name="StdWorkflow">
  <ibmwcm:workflowStage>Live</ibmwcm:workflowStage>
</ibmwcm:workflow>

<ibmwcm:workflow name="DesignLib/StdWorkflow">
  <ibmwcm:workflowStage>Live</ibmwcm:workflowStage>
</ibmwcm:workflow>
```

Date elements

This element is used to specify an RFC 822 formatted date which will be used as the date of one of the date fields in the content item.

Table 104. Date elements

<table>
<thead>
<tr>
<th>Element parameters:</th>
<th>Details for this element:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Applies to item types</td>
<td>Content items</td>
</tr>
<tr>
<td>Required for item types</td>
<td>None</td>
</tr>
</tbody>
</table>
**Table 104. Date elements (continued)**

<table>
<thead>
<tr>
<th>Element parameters:</th>
<th>Details for this element:</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Allowable Values</strong></td>
<td></td>
</tr>
<tr>
<td><strong>publishDate</strong></td>
<td>Used to set the published date of an item. If this element is not present in the <code>&lt;item&gt;</code>, the value in the <code>&lt;pubDate&gt;</code> element will be used to set the publish date of the content item. This requires a publish action to have been included in a workflow stage in the specified workflow.</td>
</tr>
<tr>
<td><strong>expirationDate</strong></td>
<td>Used to set the expiry date of an item. This requires an expire action to have been included in a workflow stage in the specified workflow.</td>
</tr>
<tr>
<td><strong>genDateOne</strong></td>
<td>Used to populate the general date fields of an item.</td>
</tr>
<tr>
<td><strong>genDateTwo</strong></td>
<td></td>
</tr>
</tbody>
</table>

**Required Attributes**
- An RFC 822 formatted date.

**Optional Attributes**
- None

**Required sub-elements**
- None

**Optional sub-elements**
- None

Examples:

```xml
<ibmwcm:publishDate>
Fri, 31 Oct 2008 15:32:00 EST
</ibmwcm:publishDate>

<ibmwcm:expirationDate>
Sun, 1 Nov 2009 12:00:00 EST
</ibmwcm:expirationDate>

<ibmwcm:genDateOne>
Wed, 1 Nov 2006 09:30:00 EST
</ibmwcm:genDateOne>

<ibmwcm:genDateTwo>
Thu, 2 Nov 2006 09:30:00 EST
</ibmwcm:genDateTwo>
```

**additionalViewer element**

This element allows the additional viewers field to be populated via the feed. It should contain the common name of a single user or group that is to be added to the field. To specify multiple users and groups, use multiple additionalViewer elements.
Table 105. additionalViewer element

<table>
<thead>
<tr>
<th>Element parameters:</th>
<th>Details for this element:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Applies to item types</td>
<td>Content items</td>
</tr>
<tr>
<td>Required for item types</td>
<td>None</td>
</tr>
</tbody>
</table>
| Allowable Values | • [all users]  
• [all authenticated portal users]  
• The common name of any valid portal user or group. |
| Required Attributes | None |
| Optional Attributes | None |
| Required sub-elements | None |
| Optional sub-elements | None |

Examples:

```xml
<ibmwcm:additionalViewer>[all users]</ibmwcm:additionalViewer>
<ibmwcm:additionalViewer>Sales</ibmwcm:additionalViewer>
<ibmwcm:additionalViewer>jsmith</ibmwcm:additionalViewer>
```

Security control elements:

The security control element is used to set access controls on the item being created.

access element

The access element provides a mechanism to set the access controls on Web Content Manager items via the feed. A feed entry can contain multiple access sub-elements. Each access element should contain the common name of a single, valid, portal user or group. The users and groups specified in the access element are added to the system defined security fields on the Web Content Manager items.

Table 106. access element

<table>
<thead>
<tr>
<th>Element parameters:</th>
<th>Details for this element:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Applies to item types</td>
<td>Content</td>
</tr>
<tr>
<td>Required for item types</td>
<td>None</td>
</tr>
</tbody>
</table>
| Allowable Values | • [all users]  
• [all authenticated portal users]  
• The common name of any valid portal user or group. |
| Required Attributes | None |
Table 106. access element (continued)

<table>
<thead>
<tr>
<th>Element parameters:</th>
<th>Details for this element:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optional Attributes</td>
<td></td>
</tr>
<tr>
<td>type</td>
<td>This attribute allows the feed producer to specify the exact access level that should be granted to the user or group. The allowable values for the type attribute correspond to the access levels available in Web Content Manager: &quot;user&quot;, &quot;contributor&quot;, &quot;editor&quot; and &quot;manager&quot;. If the type attribute is not specified, &quot;user&quot; access will be applied.</td>
</tr>
<tr>
<td>inheritance</td>
<td>You can specify whether inheritance will be enabled or disabled by adding either inheritance=&quot;enabled&quot; or inheritance=&quot;disabled&quot;. If not specified, the default or current inheritance setting for an item will be used.</td>
</tr>
</tbody>
</table>

Required sub-elements

Optional sub-elements

None

Examples:

```xml
<ibmwcm:access type="user">
  [all users]
</ibmwcm:access>
```

```xml
<ibmwcm:access type="editor" inheritance="enabled">
  Sales
</ibmwcm:access>
```

Association control element:

The association element is used to create an association between two or more separate content items. It is used to support some content item relationships that are not native to Web Content Manager but which are found in some other web content management systems. One example would be a technical journal article which needs to include some data from one or more author biography content items.

An association element is tied to a text element. The value of the element in the feed is a comma-delimited list of GUIDs which represent the other external content items that are to be associated with the text element. When the Web Content Integrator processes the feed, it will attempt to resolve the GUIDs with the corresponding Web Content Manager items. A custom JSP component is then used in the presentation template to locate the linked content items at render time and display them in the context of the container content item.

Since this type of linkage is not a natural feature of Web Content Manager it relies on a custom JSP component for its implementation and there are some restrictions related to using an association element:
• As Web Content Manager DocumentIds are used to create the association to the other content items, the format of the DocumentId is not guaranteed to be constant from version to version so there is some risk that the linkages could break following a Web Content Manager upgrade or migration.

• The referential integrity of Web Content Manager does not apply to association element links.

Table 107. association element

<table>
<thead>
<tr>
<th>Element parameters:</th>
<th>Details for this element:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Applies to item types</td>
<td>Content</td>
</tr>
<tr>
<td>Required for item types</td>
<td>None</td>
</tr>
<tr>
<td>Allowable Values</td>
<td>A list of GUIDs that map to other feed entries which are linked to associated content items.</td>
</tr>
<tr>
<td>Required Attributes</td>
<td>name - The value of this attribute should match the name of an existing text element.</td>
</tr>
<tr>
<td>Optional Attributes</td>
<td>None</td>
</tr>
</tbody>
</table>
| Required sub-elements | type - The value of this sub-element must be "association". It is not case sensitive.  
| | value - The value of this sub-element should be a comma-delimited list of GUIDs that point to the feed entries which are linked to associated content items. |
| Optional sub-elements | None                      |

Example:

```xml
<ibmwcm:element name="authors">
  <ibmwcm:type>association</ibmwcm:type>
  <ibmwcm:value>234ed298cf,7023bc23f1e</ibmwcm:value>
</ibmwcm:element>
```

Handling embedded links

From time to time, the content being retrieved by the Web Content Integrator will contain embedded links to images, files and other content within the feed. You can use the link tag in your feed to represent embedded links so that they are converted into links to other Web Content Manager items that are created when the feed is processed.

There are three types of embedded links that can be processed: images, files, and content. In all cases, the link tag must include a GUID that points to another item in the feed which describes the target item.

Content links

In this example, the following feed content contains a link to another Web page:

```xml
<p>
</p>
```
To ensure that the embedded link is converted into a link to a related content item being created when the feed is processed, you add the following code to your feed. It contains a GUID to the HTML file also being processed by the Web Content Integrator:

```xml
<![CDATA[<p>
<a href='<link type="content" guid="80000002" />'>Press Release</a>
This is some text.
</p>]]>
```

When the content of the feed is saved within Web Content Manager, a link to a content item is added to the content.

```xml
<p>
<a href="wps/wcm/myconnect/Portal/Press/Announcement1?contentIDR=29e04295034efb">Press Release</a>
This is some text.
</p>
```

Example feed

```xml
<?xml version="1.0"?>
<rss version="2.0" xmlns:ibmwcm="http://purl.org/net/ibmfeedsvc/wcm/1.0" >
<channel>
<title>Sample Feed</title>
<link>http://www.ibm.com/feeds/sample.rss</link>
<description>An example RSS Feed</description>
<lastBuildDate>Tue, 31 Oct 2006 10:31:00 EST</lastBuildDate>
<item>
<title>News Item Two</title>
<description>This is a summary of the second news article</description>
<pubDate>Tue, 31 Oct 2006 10:31:00 EST</pubDate>
<guid>80000001</guid>
<ibmwcm:action>add</ibmwcm:action>
<ibmwcm:itemType>content</ibmwcm:itemType>
<ibmwcm:authoringTemplate>News</ibmwcm:authoringTemplate>
<ibmwcm:path>/Portal/News</ibmwcm:path>
<ibmwcm:workflow name="Std Workflow">Live</ibmwcm:workflow>
<ibmwcm:component name="BODY">
<ibmwcm:type>rich text</ibmwcm:type>
<ibmwcm:value><![CDATA[<p>
<a href='<link type="content" guid="80000002" />'>some content link</a>
This is some text.
</p>]]></ibmwcm:value>
</ibmwcm:component>
</item>
<item>
<title>Announcement1</title>
<description>This is an announcement press release.</description>
<pubDate>Tue, 31 Oct 2006 10:11:00 EST</pubDate>
<guid>80000002</guid>
<ibmwcm:action>add</ibmwcm:action>
<ibmwcm:itemType>Content</ibmwcm:itemType>
<ibmwcm:authoringTemplate>Press Release</ibmwcm:authoringTemplate>
<ibmwcm:path>/Portal/Press</ibmwcm:path>
<ibmwcm:workflow name="Std Workflow">Live</ibmwcm:workflow>
<ibmwcm:component name="BODY">
<ibmwcm:type>rich text</ibmwcm:type>
<ibmwcm:value>
...</ibmwcm:value>
</ibmwcm:component>
</item>
</channel>
</rss>
```
File links

In this example, the following feed content contains a link to a PDF file:

```xml
<p>
This is some text.
</p>
```

To ensure that the embedded link is converted into a link to a related file resource component being created when the feed is processed, you add the following code to your feed. It contains a GUID to the PDF file also being processed by the Web Content Integrator:

```xml
<![CDATA[<p>
<a href='<link type="file" guid="50000002"/>'>Product Spec</a>
This is some text.
</p>]]>
```

When the content of the feed is saved within Web Content Manager, a component tag is added to the content.

```xml
<p>
<a href="<Component name="ProductSpec" />">Product Spec</a>
This is some text.
</p>
```

Example feed

```xml
<?xml version="1.0"?>
<rss version="2.0" xmlns:wcm="http://purl.org/net/ibmfeedsvc/wcm/1.0" >
  <channel>
    <title>Sample Feed</title>
    <link>http://www.ibm.com/feeds/sample.rss</link>
    <description>An example RSS Feed</description>
    <lastBuildDate>Tue, 31 Oct 2006 10:31:00 EST</lastBuildDate>
    <item>
      <title>News Item Two</title>
      <description>
        This is a summary of the second news article
      </description>
      <pubDate> Tue, 31 Oct 2006 10:31:00 EST</pubDate>
      <guid>80000001</guid>
      <ibmwcm:action>add</ibmwcm:action>
      <ibmwcm:ItemType>Content</ibmwcm:ItemType>
      <ibmwcm:authoringTemplate>News</ibmwcm:authoringTemplate>
      <ibmwcm:path>/Portal/News</ibmwcm:path>
      <ibmwcm:workflow name="Std Workflow">
        <ibmwcm:workflowStage>Live</ibmwcm:workflowStage>
      </ibmwcm:workflow>
      <ibmwcm:component name="BODY">
        <ibmwcm:type>rich text</ibmwcm:type>
        <ibmwcm:value>
          <![CDATA[
            <p>
              <a href='<link type="file" guid="50000002"/>'>some file link</a>
              This is some text.
            </p>
          ]]>]
        </ibmwcm:value>
      </ibmwcm:component>
    </item>
  </channel>
</rss>
```
Image links

In this example, the following feed content contains a link to an image file:

```xml
<p>
<img src="http://cmsserver.ibm.com/images/mylogo.gif" />
This is some text.
</p>
```

To ensure that the embedded image link is converted into a link to a related image component being created when the feed is processed, you add the following code to your feed. It contains a GUID to the image file also being processed by the Web Content Integrator:

```xml
<![CDATA[
<link type="image" guid="50000001" />
This is some text.
</p>]]>
```

When the content of the feed is saved within Web Content Manager, a component tag is added to the content.

```xml
<p>
<Component name="My Logo" />
This is some text.
</p>
```

Example feed

```xml
<?xml version="1.0"?>
<rss version="2.0" xmlns:ibmwcm="http://purl.org/net/ibmfeeds/vc/wcm/1.0">
  <channel>
    <title>Sample Feed</title>
    <link>http://www.ibm.com/feeds/sample.rss</link>
    <description>An example RSS Feed</description>
    <lastBuildDate>Tue, 31 Oct 2006 10:31:00 EST</lastBuildDate>
    <item>
      <title>News Item Two</title>
      <description>This is a summary of the second news article</description>
      <pubDate>Tue, 31 Oct 2006 10:31:00 EST</pubDate>
      <guid>8000000001</guid>
      <ibmwcm:action>add</ibmwcm:action>
      <ibmwcm:itemType>Content</ibmwcm:itemType>
      <ibmwcm:authoringTemplate>News</ibmwcm:authoringTemplate>
```
Automatic image tag processing

The Web Content Integrator has an optional feature which can be used in place of the `<link type="image" />` tag. If enabled, the Web Content Integrator will automatically parse the values of any HTML or rich text elements and search for HTML `<img/>` tags embedded within the content. If any are found, the Web Content Integrator will attempt to retrieve the referenced image file, create an image component, and then re-write the reference so that it points to the new image component.

This feature can be enabled by editing the `disable.img.proc` setting in the WCMConsumerPlugin.properties file and setting it to "false". Automatic image tag processing is disabled by default.

If enabled:

- only the content being imported into HTML elements and components or Rich Text elements and components will be processed.
- all image references matching the pattern "<img src="..." are processed. Image references that are included within Javascript code or CSS styles will not be processed.
- the URLs specified in the "src" attributes of those image tags are converted to fully qualified URLs using other information about the content item in the feed. Relative links must be relative to the URL in the `<link>` element of the feed following the standard rules for relative links.
- the image files are stored as shared image components. The access controls on the image component is set to match that of the content item which references it. The name of the image component will be based on the server-relative path to...
the image. For example, an image located at http://<host_name>/resources/images/sm_logo.gif will be named resources.images.sm_logo.gif.

Processing images
The Web Content Integrator has an image processing feature which allows images referenced within HTML and rich text elements to be created as image components while processing a feed.

Image processing configuration

The following parameters can be set in the WCM WCMConfigService service using the WebSphere Integrated Solutions Console:

Enable and disable image processing
By default image processing is enabled. To disable image processing set the property: disable.img.processing=true

Enable and disable embedded rich text images
Images can be embedded directly into the rich text elements instead of first creating an image component. This functionality makes images within rich text elements consumed by the Web Content Integrator be processed identically to images added to rich text elements in the authoring UI. By default embedded rich text images are disabled. To enable them, set the properties:
- disable.img.processing=false
- richtext.embedded.images.enabled=true

Using absolute image source URLs

Images within HTML and rich text elements can be specified using absolute HTTP URLs within the image source attribute. The Web Content Integrator will process absolute image source URLs as specified in the feed.

Example feed item:

```xml
<item>
  <title>RichText Component With Image 1</title>
  <pubDate>Thu, 30 Mar 2011 16:00:00 EDT</pubDate>
  <guid>Image_Example_1</guid>
  <ibmwcm:action>add</ibmwcm:action>
  <ibmwcm:itemType>Component</ibmwcm:itemType>
  <ibmwcm:element>
    <ibmwcm:type>rich text</ibmwcm:type>
    <ibmwcm:value>
      <![CDATA[
        <p>Image 2:</p><img src="http://wci-feed-server/img/g/guitar.jpg"/>
      ]]>"
    </ibmwcm:value>
  </ibmwcm:element>
</item>
```

Using relative image source URLs

Images within HTML and rich text elements can be specified using relative HTTP URLs within the image source attribute and a base URL from the link element of the item. The Web Content Integrator will process relative image source URLs as a concatenation of the item's link element and image source attributes in the HTML.

Example feed item:
Using both absolute and relative image source URLs

Images within HTML and rich text elements can be specified using a combination of relative and absolute HTTP URLs within the image source attributes. The Web Content Integrator will process relative image source URLs as a concatenation of the item's link element and image source attributes in the HTML. The Web Content Integrator will process absolute image source URLs as specified in the feed.

Example feed item:

```xml
<item>
  <title>RichText Component With Image 3</title>
  <pubDate>Thu, 30 Mar 2011 16:00:00 EDT</pubDate>
  <guid>Image_Example_3</guid>
  <link>http://wci-feed-server</link>
  <ibmwcm:action>add</ibmwcm:action>
  <ibmwcm:itemType>Component</ibmwcm:itemType>
  <ibmwcm:element>
    <ibmwcm:type>rich text</ibmwcm:type>
    <ibmwcm:value><![CDATA[
      <p>Image 2:</p><img src="/img/f/fish.jpg"/>
      <p>Image 3:</p><img src="/img/g/grapes.jpg"
    ]]>]]>
    </ibmwcm:value>
  </ibmwcm:element>
</item>
```

Expected results – rich text component

Example rich text component feed item:

```xml
<item>
  <title>RichText Component With Image 4</title>
  <pubDate>Thu, 30 Mar 2011 16:00:00 EDT</pubDate>
  <guid>Image_Example_4</guid>
  <link>http://wci-feed-server</link>
  <ibmwcm:action>add</ibmwcm:action>
  <ibmwcm:itemType>Component</ibmwcm:itemType>
  <ibmwcm:element>
    <ibmwcm:type>rich text</ibmwcm:type>
    <ibmwcm:value><![CDATA[
    ]]>]]>
    </ibmwcm:value>
  </ibmwcm:element>
</item>
```
Rich text component with image processing disabled:

- A rich text component named: “RichText Component With Image 4” is created in the feed library.
- The rich text HTML is not modified.
- The image source is unmanaged, because Web Content Manager cannot detect a broken image source.

Rich text component with image processing enabled:

- A rich text component named: “RichText Component With Image 4” is created in the feed library.
- An Image Component named “img.l.leaf.jpg” is created in the feed library.
- The image source is replaced with a URL pointing to the newly created image.
- The image source is managed by Web Content Manager and will be updated as the image component is updated.

Rich text component with embedded images enabled:

- A rich text component named: “RichText Component With Image 4” is created in the feed library.
- The rich text HTML is modified replacing the image source with a URL for the image embedded within the rich text Component.
- The image source is managed by Web Content Manager and will be updated as the rich text changes.

Expected results – HTML component

Example feed item:

```xml
<item>
  <title>HTML Component With Image 1</title>
  <pubDate>Thu, 30 Mar 2011 16:00:00 EDT</pubDate>
  <guid>Image_Example_5</guid>
  <ibmwcm:action>add</ibmwcm:action>
  <ibmwcm:itemType>Component</ibmwcm:itemType>
  <ibmwcm:element>
    <ibmwcm:type>html</ibmwcm:type>
    <ibmwcm:value>
      <![CDATA[
      ]]> 
    </ibmwcm:value>
  </ibmwcm:element>
</item>
```

HTML element with image processing disabled

- HTML Component named “HTML Component With Image 1” is created in the feed library.
Example feeds
View some example feeds which illustrate how to represent the various item types within a feed.

Note: If non-ascii data is used in a feed, then encoding="UTF-8" must be specified in the feed: <?xml version="1.0" encoding="UTF-8"?>

Site area feed
<?xml version="1.0"?>
<rss version="2.0" xmlns:ibmwcm="http://purl.org/net/ibmfeedsvc/wcm/1.0" >
  <channel> 
    <title>Sample Feed</title> 
    <link>http://www.ibm.com/feeds/sample.rss</link> 
    <description>An example RSS Feed</description> 
    <lastBuildDate>Tue, 31 Oct 2006 10:31:00 EST</lastBuildDate> 
    <item> 
      <title>News</title> 
      <description>This is a Site Area.</description> 
      <guid>20000001</guid> 
      <pubDate> Tue, 31 Oct 2006 10:31:00 EST </pubDate> 
      <author>jsample@ibm.com</author> 
      <ibmwcm:action>add</ibmwcm:action> 
      <ibmwcm:itemType>siteArea</ibmwcm:itemType> 
      <ibmwcm:path>/SiteAreaName</ibmwcm:path> 
      <ibmwcm:owner>jsample</ibmwcm:owner> 
      <ibmwcm:defaultContent>80000002</ibmwcm:defaultContent> 
      <ibmwcm:children> 
        <ibmwcm:childGuid>20000011</ibmwcm:childGuid> 
        <ibmwcm:childGuid>80000002</ibmwcm:childGuid> 
      </ibmwcm:children> 
      <ibmwcm:access type="user">[all users]</ibmwcm:access> 
      <ibmwcm:templateMap authoring="AT News" presentation="PT News" /> 
    </item> 
  </channel> 
</rss>
Taxonomy feed
<?xml version="1.0"?>
<rss version="2.0" xmlns:ibmwcm="http://purl.org/net/ibmfeedsvc/wcm/1.0">
  <channel>
    <title>Sample Feed</title>
    <link>http://www.ibm.com/feeds/sample.rss</link>
    <description>An example RSS Feed</description>
    <lastBuildDate>Tue, 31 Oct 2006 10:31:00 EST</lastBuildDate>
    <item>
      <title>Audience</title>
      <link>http://www.ibm.com/index.htm</link>
      <description>This is the top-level Taxonomy.</description>
      <guid>30000001</guid>
      <pubDate>Tue, 31 Oct 2006 10:31:00 EST</pubDate>
      <author>jsample@ibm.com</author>
      <ibmwcm:action>add</ibmwcm:action>
      <ibmwcm:itemType>category</ibmwcm:itemType>
      <ibmwcm:path>/</ibmwcm:path>
      <ibmwcm:owner>jsample</ibmwcm:owner>
      <ibmwcm:access type="user">[all users]</ibmwcm:access>
    </item>
  </channel>
</rss>

Category feed
<?xml version="1.0"?>
<rss version="2.0" xmlns:ibmwcm="http://purl.org/net/ibmfeedsvc/wcm/1.0">
  <channel>
    <title>Sample Feed</title>
    <link>http://www.ibm.com/feeds/sample.rss</link>
    <description>An example RSS Feed</description>
    <lastBuildDate>Tue, 31 Oct 2006 10:31:00 EST</lastBuildDate>
    <item>
      <title>Employees</title>
      <link>http://www.ibm.com/index.htm</link>
      <description>This is a second-level Category.</description>
      <guid>40000001</guid>
      <pubDate>Tue, 31 Oct 2006 10:31:00 EST</pubDate>
      <author>jsample@ibm.com</author>
      <ibmwcm:action>add</ibmwcm:action>
      <ibmwcm:itemType>category</ibmwcm:itemType>
      <ibmwcm:path>/Audience</ibmwcm:path>
      <ibmwcm:createLinks>
        <ibmwcm:parentGuid>30000001</ibmwcm:parentGuid>
      </ibmwcm:createLinks>
      <ibmwcm:owner>jsample</ibmwcm:owner>
      <ibmwcm:children>
        <ibmwcm:childGuid>40000001</ibmwcm:childGuid>
      </ibmwcm:children>
      <ibmwcm:access type="user">[all users]</ibmwcm:access>
    </item>
  </channel>
</rss>

Component feed
<?xml version="1.0"?>
<rss version="2.0" xmlns:ibmwcm="http://purl.org/net/ibmfeedsvc/wcm/1.0">
  <channel>
    <title>Sample Feed</title>
    <link>http://www.ibm.com/feeds/sample.rss</link>
    <description>An example RSS Feed</description>
    <lastBuildDate>Tue, 31 Oct 2006 10:31:00 EST</lastBuildDate>
  </channel>
</rss>
The long anticipated Release 2.0 of our flagship product became generally available earlier today.

RSS Namespace Extension for the Feed Service
The RSS namespace extension is used to exchange control information between the feed producer and consumer applications.

To implement this namespace, specify the <rss> element as follows:

```xml
<rss version="2.0" xmlns:ibmwcm="http://purl.org/net/ibmfeedsvc/wcm/1.0"
     xmlns:ibmfs="http://purl.org/net/ibmfeedsvc/feedsvc/1.0"/>
```

The input feeds only need to include this namespace in certain scenarios where control information is going to be passed in the feed itself rather than in the HTTP headers.

The Handshake Protocol:
In many Web Content Integrator implementations the input content feeds are produced by an application which generates them dynamically in response to requests from the feed consumer application. In those cases it is useful for the consumer to be able to indicate to the producer application which versions of a feed the Consumer has already seen so that the producer can make some intelligent decisions about what to include in the next feed.

The handshake protocol consists of the exchange of two feed attributes: a Last Modified Date, and an arbitrary Entity Tag. The values of both of these attributes are managed exclusively by the producer. The consumer simply receives the values from the producer, stores them, and passes them back unchanged on the next request. These attributes can be passed in one of two ways:

1. As HTTP header fields:
   - Requests from the consumer contain:
     
     ```
     If-Modified-Since: {last-modified_value}
     If-None-Match: {etag_value}
     ```
   - Responses from the producer contain:
     
     ```
     ETag: {etag_value}
     Last-Modified: {last-modified_value}
     ```

2. As elements in the feed and query string parameters:
   - Requests from the consumer are in the form:
     
     ```
     http://cmsserver.example.org/ProducerApp?etag={etag_value}&lastMod={last-modified_value}
     ```
Feeds sent back from the producer contain the following channel-level elements:

\[
\text{<lastBuildDate>{last-modified_value}</lastBuildDate>}
\text{<ibmfs:etag>{etag_value}</ibmfs:etag>}
\]

**How the handshake works**

1. The consumer makes its first request to the producer. This request does not contain any special header fields.

2. The producer receives the request, and since there were no special headers, it responds with a feed that contains items which represent all transactions that have occurred up to that point. Before sending the response it sets the Last-Modified header to the current time and the ETag header to a value that enables the producer identify this specific instance of the feed.

   Last-Modified: Thu, 7 Sep 2006 12:00:00 GMT
   ETag: ABC0011

3. The consumer receives the response, processes the feed, and stores the Last-Modified and ETag values only if the feed was processed successfully.

4. The next time the consumer is triggered, it again makes a request to the producer for the transaction feed. This time it sets the If-Modified-Since header to the Last-Modified date it received on the last request and the If-None-Match field to the value of the ETag it received on the last request.

   If-Modified-Since: Thu, 7 Sep 2006 12:00:00 GMT
   If-None-Match: ABC0011

5. The producer receives the request, and checks the values of the header fields. It then uses its own internal logic to generate a new feed which only contains changes that occurred since the last feed it sent to this consumer. It sends back a feed with the following header values:

   Last-Modified: Thu, 7 Sep 2006 13:00:00 GMT
   ETag: ABC0032

6. The consumer receives the response, processes the feed, and updates its stored values for the Last-Modified date and ETag.

Ideally, if no changes have occurred since the last time the consumer requested the feed, the producer should respond with an HTTP 304 (Not Modified) response code. This will signal the consumer that there are no changes and thus that it does not need to attempt to parse the feed and do the subsequent processing. However, if the producer is unable to implement this feature it won't affect any of the other processing.

The consumer will always send a request containing the Etag label of the last transaction feed it successfully received. In this case the producer application will be responsible for re-sending any entries that might have been missed due to a communication failure between the servers.

**Results Feeds:**

When feed processing is initiated via a call to the feed service servlet, the Web Content Integrator will respond with an output feed. The first entry in this feed will contain status information for the feed as a whole. Each of the subsequent entries in the output feed will correspond to an item that was in the input feed. These latter entries will contain status information about the results of processing each item. Feed producers could use this information to attempt to automatically recover from certain types of errors.

Each entry in the output feed will have the following general format:
Channel-level Elements:

The following element should be a direct child of the feed's <channel /> element since it applies to the feed as a whole rather than to an individual item in the feed.

**etag**

Some producer applications are not able to manipulate the HTTP headers to set the handshake data. In order to support those applications, there is an alternative method which entails passing the same information directly in the feed using the RSS <lastBuildDate> element as well as a custom namespace element, <ibmfs:etag>.

*Table 108. etag element*

<table>
<thead>
<tr>
<th>Element parameters:</th>
<th>Details for this element:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Allowable Values</td>
<td>A string value that represents some meaningful label for the specific instance of the feed.</td>
</tr>
<tr>
<td>Required Attributes</td>
<td>None</td>
</tr>
<tr>
<td>Optional Attributes</td>
<td>None</td>
</tr>
<tr>
<td>Required sub-elements</td>
<td>None</td>
</tr>
<tr>
<td>Optional sub-elements</td>
<td>None</td>
</tr>
</tbody>
</table>

Example:
<ibmfs:etag>ABC0012</ibmfs:etag>

Item-level Elements:

The following elements are applied at the <item /> level since they contain information that is specific to each individual feed entry. These elements are only used in the output feed. They have no meaning in the context of an input feed.
resultCode

Table 109. resultCode element

<table>
<thead>
<tr>
<th>Element parameters:</th>
<th>Details for this element:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Allowable Values</td>
<td>This element will have one of the following values depending on what happened when the input feed entry was processed:</td>
</tr>
<tr>
<td></td>
<td>OK The entry was processed completely with no warnings or errors.</td>
</tr>
<tr>
<td></td>
<td>WARN The entry was processed completely but one or more warnings were logged.</td>
</tr>
<tr>
<td></td>
<td>ERROR One or more error messages were logged. The entry may not have been saved in Web Content Manager.</td>
</tr>
<tr>
<td></td>
<td>FAIL The content item could not be saved or updated.</td>
</tr>
</tbody>
</table>

Required Attributes: None
Optional Attributes: None
Required sub-elements: None
Optional sub-elements: None

Example:
<ibmfs:resultCode>OK</ibmfs:resultCode>
<ibmfs:resultCode>WARN</ibmfs:resultCode>

resultMsg

Table 110. resultMsg element

<table>
<thead>
<tr>
<th>Element parameters:</th>
<th>Details for this element:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Allowable Values</td>
<td>Each instance of this element will contain the details of a warning or error message.</td>
</tr>
<tr>
<td>Required Attributes</td>
<td>level The value of this attribute indicates the severity of the message. Allowable values are: &quot;WARN&quot; and &quot;ERROR&quot;.</td>
</tr>
<tr>
<td></td>
<td>code This attribute should hold the error code that is associated with the message.</td>
</tr>
</tbody>
</table>

Optional Attributes: None
Required sub-elements: None
Optional sub-elements: None

Examples:
<ibmfs:resultMsg level="WARN" code="WCM0014">Default Content GUID does not point to a content item. Default content set to null.</ibmfs:resultMsg>
A WCMException was thrown when attempting to move the content.

**documentId**

**Table 111. documentId element**

<table>
<thead>
<tr>
<th>Element parameters:</th>
<th>Details for this element:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Allowable Values</td>
<td>This element contains the ID that was generated by Web Content Manager.</td>
</tr>
<tr>
<td>Required Attributes</td>
<td>None</td>
</tr>
<tr>
<td>Optional Attributes</td>
<td>None</td>
</tr>
<tr>
<td>Required sub-elements</td>
<td>None</td>
</tr>
<tr>
<td>Optional sub-elements</td>
<td>None</td>
</tr>
</tbody>
</table>

Example:

```xml
<ibmfs:documentId>
com.ibm.workplace.wcm.api.WCM_Content/Connections Announcement/f6f60a00498c0d759ee9ffe695695374/PUBLISHED
</ibmfs:documentId>
```

**WebDAV**

With WebDAV for IBM WebSphere Portal, you can use standard operating system tools to create, modify, and delete web content rather than the standard authoring portlet.

Before you can use WebDAV with web content, you will need to set up a WebDAV client. After your client is set up, you can access the web content libraries with WebDAV using the following URL:

http://server:port/portal_context_root/mycontenthandler/dav/content/libraries/

For example:

http://www.example.com:10039/wps/mycontenthandler/dav/content/libraries/

By leveraging tools like file system explorers, WebDAV enables you to work with your web content items through familiar, everyday actions. Here are a few examples:

- You can create components or presentation templates simply by dragging a file into a corresponding folder.
- You can perform actions on several items at once. For example, you can create five images at the same time by dragging five image files into the image component folder. This creates five separate image components, and for each image component the file name is used for the component's name and title.
- Modifying items is also straightforward through a WebDAV client. For example, you can open a presentation template using your preferred HTML editor, make changes to it, and then save the changes. The WebDAV client takes care of accessing the web content library, downloading the template, and then uploading the changes.

In addition to modifying the actual content of an item, you can also modify any item's metadata or access control settings by modifying XML files that define the
item’s metadata and access control characteristics. You can also drag an existing XML file into the appropriate folder, enabling you to easily set the same data for different items.

You can create, modify, or delete the following items: libraries, taxonomies, categories, site areas, folders, components, and presentation templates.

**Note:** Be aware that the following features are not supported when using WebDAV with web content:
- Content items, with the exception of managing metadata and access control
- Authoring templates, with the exception of managing metadata and access control
- Nested items within site areas
- Server-side copy and move
- Unauthenticated users
- Exporting of web content libraries with WebDAV to be imported into another web content server

When using WebDAV with web content, be aware of the following considerations.

**Locked item support**
Locking or unlocking an item through WebDAV will lock or unlock the item in Web Content Manager and the JCR database. Because some items are represented by multiple files and folders, locking or unlocking one of these files causes locking or unlocking of the other associated files at the same time. If you lock an item, folders and files related to the content of the item, its metadata, and its access control settings are also locked.

**Workflow support**
There is no representation of a workflow itself in the WebDAV tree, but if a file is part of a workflow and the workflow indicates that the file is in a state that allows users to modify it, WebDAV will allow you to modify the file as well.

**File names and file type suffixes**
Files representing data items are always named exactly like the corresponding content item. For example if you have an image component named `myImage`, the corresponding image file is also named `myImage`, without any suffix indicating the file type, such as `.gif` or `.jpg`. This can sometimes cause a problem when opening the file through WebDAV because the appropriate application for editing the file cannot be chosen automatically. To account for this, you can either rename the component itself to include the file type (for example, `myImage.gif`), or you can manually start the editing application and open the file from within the client.

**Missing items**
If an item no longer displays or can no longer be modified, this could be due to a changed state for the item in the web content server where the item is stored. For example creating or modifying an item on the server could lead to a changed state that prevents you from accessing this item with WebDAV, depending on how workflow is set up. Expiration is another reason an item’s state might change and so affect whether you can access the item with WebDAV.

**Configuring a HTTP server front end**
When you use an HTTP server front end to work with WebDAV, you need
to set **Accept content for all requests** to true for the Web server plugin in the WebSphere Integrated Solutions Console under **Web servers > webserver1 > Plug-in properties > Request and response.**

**Related tasks:**

**Working with WebDAV clients**
WebDAV is an HTTP extension framework with a plug point for the access and management of hierarchical data, for example, in content management systems. WebDAV stores the data in collections and allows you to work with the data in a user interface view that is similar to that of a file system. Various tools are available for integrating WebDAV resources into the client file system, known as WebDAV clients. To use WebDAV for WebSphere Portal, you must first set up your WebDAV clients.

**Web content items in the WebDAV tree**
The WebDAV tree containing your Web content items begins at the WebDAV root `/libraries/`, which displays all libraries to which you have access. All Web content items within the libraries are organized with folders and files.

Items that do not pertain to content, such as site areas or categories, are represented as folders that only contain the item's metadata folder and any child items like other site areas or categories. No data files are present.

```
sites
  - wcm.siteArea.siteArea1
    - meta-data
    - wcm.siteArea.siteArea1.1
      - meta-data
      - wcm.siteArea.siteArea1.1.1
        - meta-data
        - wcm.siteArea.siteArea1.1.1.1
          - meta-data
          - wcm.siteArea.siteArea1.1.1.1.1
    - wcm.siteArea.siteArea1.1.2
      - meta-data
  - wcm.siteArea.siteArea2
    - meta-data
```

Data-oriented items like image components or presentation templates are represented as files so you can manipulate them with drag and drop operations. The corresponding metadata for the item is managed the same way as for the non-data items but in separate subfolders within the metadata folder.

```
wcm.comps.image
  image1.jpg
  image2.jpg
  - meta-data
    - wcm.comp.image1.jpg
    - wcm.comp.image2.jpg
```

In addition to folders that represent actual Web content items, there are folders in the WebDAV tree to structure the data or to allow for better scalability. For each library there are folders for components, presentation templates, sites, and taxonomies.

```
libraries
  - wcm.library.my_library
    - authoringTemplates
    - components
    - presentationTemplates
    - sites
    - taxonomies
  - wcm.library.contentlibrary
```
Within the components folder there are subfolders for the component types for better scalability and management of the different types of components.

libraries
- wcm.library.my_library
  - components
    - wcm.comps.authoring.tools
    - wcm.comps.component.references
    - wcm.comps.data.and.time
    - wcm.comps.federated.content
    - wcm.comps.file
    - wcm.comps.html
    - wcm.comps.image
    - wcm.comps.jsp
    - wcm.comps.link
    - wcm.comps.menu
    - wcm.comps.navigator
    - wcm.comps.number
    - wcm.comps.page.navigation
    - wcm.comps.personalization
    - wcm.comps.richt.text
    - wcm.comps.search
    - wcm.comps.short.text
    - wcm.comps.style.sheet
    - wcm.comps.taxonomy
    - wcm.comps.text
    - wcm.comps.user.name
    - wcm.comps.user.selection

To organize your authoring templates, presentation templates, and components, you can create custom Web content folders. These Web content folders are represented as folders in WebDAV and contain a set of component types structured in the same way as the root components folder. Here an example of a component folder structure:

- components
  - CustomComponentFolder1
    - wcm.comps.authoring.tools
    - wcm.comps.component.references
    - wcm.comps.data.and.time
    - wcm.comps.federated.content
    - wcm.comps.file
    .
    .
    .
    - wcm.comps.user.selection
  - CustomComponentFolder2
    - wcm.comps.authoring.tools
    - wcm.comps.component.references
    - wcm.comps.data.and.time
    - wcm.comps.federated.content
    - wcm.comps.file
    .
    .
    .
    - wcm.comps.user.selection
  - wcm.comps.authoring.tools
  - wcm.comps.component.references
  - wcm.comps.data.and.time
  - wcm.comps.federated.content
  - wcm.comps.file
Metadata and access control for Web content items in WebDAV
WebDAV uses XML files to represent metadata and access control information for a
Web content item. You can make changes to an item's metadata and access control
settings by modifying these files, and you can specify settings for multiple files by
copying the XML files to their appropriate locations in the WebDAV tree.

Metadata
An item's metadata is represented by the meta-data.xml file, which describes
identification information for the item, including the name and title for the item
and the list of authors and owners associated with the item.

Here is a sample meta-data.xml file:
<?xml version="1.0" encoding="UTF-8"?>
<meta-data>
    <item>
        <title lang="en" value="test1.JPG"/>
        <description lang="en" value="description"/>
        <wcm-group id="authors">
            <member DN="all_auth_portal_users" type="group"/>
            <member DN="uid=wpsadmin,o=defaultWIMFileBasedRealm" type="user"/>
        </wcm-group>
        <wcm-group id="owners">
            <member DN="uid=wpsadmin,o=defaultWIMFileBasedRealm" type="user"/>
            <member DN="all_users" type="group"/>
        </wcm-group>
    </item>
</meta-data>

Access control
An item's access control information is represented by the following files:
• access-control-system.xml: Contains access control settings for the system
  specified by the administrator.
• access-control-user.xml: Contains access control settings defined by the user.

In addition to these item-specific files, the access-control.xml file is provided for
folders that represent resource types, like the components folder, and contains
access control settings for the resource type.

Here is a sample access-control.xml file for resource access control settings:
<?xml version="1.0" encoding="UTF-8"?>
<access-control>
    <resource-config>
        <role-block role-type="Editor" type="inheritance"/>
        <role-block role-type="User" type="inheritance"/>
        <role-block role-type="Editor" type="propagation"/>
        <role-block role-type="User" type="propagation"/>
    </resource-config>
    <role-list>
        <role type="Administrator">
            <member DN="uid=wpsadmin,o=defaultWIMFileBasedRealm" type="user"/>
        </role>
        <role type="Contributor">
            <member DN="all_auth_portal_users" type="group"/>
        </role>
    </role-list>
</access-control>
Here is a sample access-control-system.xml file for an item's administrator-defined access control settings:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<access-control>
    <resource-config>
        <role-block role-type="Contributor" type="inheritance"/>
        <role-block role-type="Manager" type="inheritance"/>
    </resource-config>
    <role-list>
        <role type="Editor">
            <member DN="authors" type="virtual"/>
        </role>
        <role type="User">
            <member DN="all_auth_portal_users" type="group"/>
        </role>
    </role-list>
</access-control>
```

Here is a sample access-control-user.xml file for an item's user-defined access control settings:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<access-control>
    <role-list>
        <role type="Contributor">
            <member DN="owners" type="virtual"/>
        </role>
        <role type="Editor">
            <member DN="all_auth_portal_users" type="group"/>
        </role>
        <role type="Manager">
            <member DN="uid=wpsadmin,o=defaultWIMFileBasedRealm" type="user"/>
        </role>
    </role-list>
</access-control>
```

**Related tasks:**

"Managing metadata and access control settings for authoring templates with WebDAV" on page 332

With WebDAV you can change the metadata information for an authoring portlet or update the template's access control settings.

**Creating taxonomies and categories with WebDAV**

Taxonomies and categories are profiling methods used to group content items, and you can work with taxonomies and categories directly through WebDAV. Taxonomies and categories are represented in WebDAV as folders, and you can set up your taxonomy by creating and nesting folders.

All taxonomies and categories for a given library are listed under the taxonomies folder for that library.

```
taxonomies
    - wcm.taxonomy.taxonomy1
      - meta-data
        access-control-system.xml
        access-control-user.xml
        meta-data.xml
```
To create a new taxonomy or category for your library, create a new folder using with the `wcm.taxonomy` prefix or the `wcm.category` prefix. Taxonomies can only be created under the generic taxonomies folder, while categories can be created in either a `wcm.taxonomy.*` folder or a `wcm.category.*` folder.

**Important:** Some WebDAV clients create a folder with a default name, such as New Folder, and as soon as you enter the name of the new folder, the client sends a request to rename the already created folder. Because taxonomy and category folders require a corresponding prefix for creation, this client behavior does not work. If your WebDAV client uses this method to create new folders, you can first create the new taxonomy or category folder locally and then copy it into the WebDAV tree.

**Deleting taxonomies and categories:** To delete taxonomies or categories simply delete the corresponding folder. Note that taxonomies or categories containing categories cannot be deleted until you have also first deleted the child items. Also if a category is still being referenced by another item, it cannot be deleted until you have first removed the corresponding references using the authoring portlet.

**Managing content with site areas in WebDAV**

Site areas are used to organize content items in your Web content system. In WebDAV site areas are represented as folders, and you can set up your site structure by creating and nesting folders. A content items within a site area is represented as a folder containing the metadata and access control settings for the content item.

All site areas and content items for a given library are listed under the `sites` folder for that library.
Note: Support for content items is limited to modifying the metadata and access control settings. You cannot create or delete content items using WebDAV.

To create a new site area for your library, create a new folder using with the `wcm.siteArea` prefix.

**Important:** Some WebDAV clients create a folder with a default name, such as New Folder, and as soon as you enter the name of the new folder, the client sends a request to rename the already created folder. Because site area folders require a corresponding prefix for creation, this client behavior does not work. If your WebDAV client uses this method to create new folders, you can first create the new site area folder locally and then copy it into the WebDAV tree.

**Deleting site areas:** To delete site areas simply delete the corresponding folder. Note that site areas containing site areas or content items cannot be deleted until you have also first deleted the child items. To delete child content items before deleting a site area, you must use the authoring portlet rather than WebDAV.

**Creating components with WebDAV**

Components are used to store elements in your Web content system, and you can use WebDAV to create and manage components. Each component type is represented as a folder in WebDAV, with individual components being represented as files in the appropriate component folder.

All components for a given library are listed as folders under the components folder for that library. Within the components folder, you can also create custom folders that you can use to organize your components. Like the root components folder, custom folders contain folders for each type of component.
Components are data-oriented items and represented as files and metadata folders.

Important: Although displayed in WebDAV, the following components cannot be created or modified through WebDAV and are represented by empty files:

- Authoring tools
- Component references
- Federated content
- JSP
- Menu
- Navigator
To make changes to these components, you must use the authoring portlet.

**Link component limitation:** Currently link components are not fully supported by WebDAV. The WebDAV file representing the link component contains only the URL of the link itself but no other information, such as the link text. For example, if you use WebDAV to modify a link component containing an HTML representation of `<a href='www.lotus.com'>lotus software</a>` and change the URL to `www.ibm.com`, the link text will still be rendered as *lotus software*, because that information cannot be modified with WebDAV.

To create components for your library, drag one or more files into the appropriate component type folder. When you create a new component in this way, the object's file name is used as the name and title of the new component, and the file's content is stored as the component's data. In addition the user authenticated with the WebDAV client is specified as the author and owner of the new component. For example, you could drag an HTML file into the `wcm.comps.html` folder for a new HTML component or into the `wcm.comps.rich.text` folder for a new rich text element.

**Important:** Placing an incompatible file into a component type folder (for example, putting a JPEG file into the `wcm.comps.html` folder) can cause errors during component creation and might result in an unusable component.

**Updating components:** To update an existing component, you can simply replace the corresponding file in the WebDAV tree with a new file that has the same name. For example you can place `myCoolPic.jpg` into the image components folder that already contains `myCoolPic.jpg`, and the component will automatically be updated with the new file's content. If you place a file with a different name, a new component with that name is created.

**Creating presentation templates in WebDAV**

With WebDAV you can create and maintain presentation templates to define the layout and appearance characteristics of Web pages used to display content. You can also create nested image components for use with the presentation templates. Presentation templates are stored in a folder with nested image components in an associated folder.

All presentation templates for a given library are listed under the `presentationTemplates` folder for that library. Because they are data-oriented items, presentation templates are represented as files and *meta-data* folders. Nested image components are stored in a folder named after its associated presentation template—for example, `template_name_files`.

```
libraries
  - wcm.library.my_library
    - presentationTemplates
      template1.html
      myTemplate.html
    - meta-data
      - wcm.presentationTemplate.template1.html
```
1. To create presentation templates for your library, drag one or more files into the presentationTemplates folder. When you create a new presentation template in this way, the object's file name is used as the name and title of the new template, and the file's content is stored as the template's data. In addition the user authenticated with the WebDAV client is specified as the author and owner of the new template.

**Important:** Placing an incompatible file into the presentationTemplates folder (for example, a JPEG file) can cause errors during template creation and might result in an unusable presentation template.

**Deleting presentation templates:** To delete a presentation template simply delete the corresponding file. If the presentation template is being referenced by another item, such as a site area, it cannot be deleted until you have first removed the corresponding references using the authoring portlet.

**Updating presentation templates:** To update an existing presentation template, you can simply replace the corresponding file in the WebDAV tree with a new file that has the same name. For example you can place myTemplate.html into the presentationTemplates folder, replacing the myTemplate.html file that is already there, and the presentation template will automatically be updated with the new file's content. If you place a file with a different name, a new template with that name is created.

2. Create any nested image components for your presentation template by adding the image files to the template_name_files folder for your template. For example, if your template is template1.html, you would add the image files to the template1.html_files folder.

**Note:** When you add an image to the nested components folder, a temporary image is created initially, and the image is only permanently added to the list of nested components when a reference to that image is added to the presentation template's HTML code. This is done to prevent orphaned components within the presentation template.

3. If you have added a nested image component, update the presentation template's HTML code to reference the component according to the relative WebDAV path to the component.

For example, to reference a nested image component, you would update the template1.html file with the following code:

```html
<img src="./template1.html_files/nestedImage.jpg" border="0" title="my nested image"/>
```

To reference a standard image component, you would use HTML code similar to the following example:

```html
<img src="../components/wcm.comps.image/test2.jpg" border="0" title="my test image"/>
```
Managing metadata and access control settings for authoring templates with WebDAV

With WebDAV you can change the metadata information for an authoring portlet or update the template’s access control settings.

All authoring templates for a given library are listed under the authoringTemplates folder for that library. Because they are data-oriented items, authoring templates are represented as files and meta-data folders.

libraries
- wcm.library.my_library
  - authoringTemplates
    auth_template1.html
    myAuthTemplate.html
  - meta-data
    - wcm.presentationTemplate.auth_template1.html
      access-control-system.xml
      access-control-user.xml
      meta-data.xml
    - wcm.presentationTemplate.myAuthTemplate.html
      access-control-system.xml
      access-control-user.xml
      access-control.xml
      meta-data.xml

Note: You cannot modify the authoring template itself in WebDAV. To make changes to the authoring template, use the authoring portlet.

1. To change the access control settings for an authoring template, edit the access-control-system.xml file for administrator settings or the access-control-user.xml file for user-defined settings.

2. To change the metadata for an authoring template, edit the meta-data.xml file for the authoring template.

Related concepts:
“Metadata and access control for Web content items in WebDAV” on page 325

WebDAV uses XML files to represent metadata and access control information for a Web content item. You can make changes to an item’s metadata and access control settings by modifying these files, and you can specify settings for multiple files by copying the XML files to their appropriate locations in the WebDAV tree.

Blogs

Use blogs and blog libraries to provide news and commentary on a variety of subjects pertinent to your intranet and extranet sites. Blogs and blog libraries typically combine text with graphics and links to other blogs and web sites. Entries that you create and post are arranged in reverse-chronological order, with the newest entry displayed first. Readers can post comments about your entries, fostering discussions and online networking. You can manage your own blog entries and comment on other blog entries. You can also incorporate tagging and rating as you would with other WebSphere Portal content.

Tip: Use a blog for sharing information on a single subject. Use a blog library to share information on multiple discussions.
Related concepts:

Tagging and rating

Get an overview of the administrative tasks related to tagging and rating.

Learn about the template libraries used by blogs and blog libraries

Blogs and blog libraries use the template libraries provided by IBM Web Content Manager. Blog libraries use Web Resources v70 library and Blog v70 library. Blogs use the Web Resources v70 and Blog Solo v70 library. The page hierarchy that is provided for these components is the common one defined by the Web Content Manager template libraries.

Note: The Web Content Manager libraries described in this topic only work with the theme customizer used in the default theme. Customization to these libraries will affect all blogs and blog libraries on the site.

Web Resources v70 library

The Web Resources v70 library provides the authoring and presentation templates for blogs and blog libraries. All blogs on the site use this single shared library. Changes to the main shared library affects all blogs.

Blog Solo v70 library

When you add a blog to a page, a copy of the Blog Solo v70 library is created using the name that you provided when adding a blog to a page. This copy of the Blog Solo v70 library is used to store posts and comments.

Blog v70 library

When you add a blog library to a page, a copy of the Blog v70 library is created using the name that you provided when adding the blog library to a page. The copy of the Blog v70 library is used to store posts and comments when new pages are added to the blog library.

Adding a blog or blog library to a page

With Editor access to the portal or the portal page, you can add a blog or blog library to a page. Choose a blog to collaborate with your team on a single topic. Choose a blog library to collaborate with your team on multiple topics in a centralized view.

Refer to the WebSphere Portal Family Wiki for an example of how you can modify your custom theme to enable the capability of adding blogs and wikis to a page.

1. Create a project or access an existing project, and open the project in edit mode.
2. If you have not already done so, create a page for the blog or blog library or open an existing page.
3. Click the Content tab, then click Communications.
4. Click Blog Library to add a team blog or click Blog to create a single-topic blog.
5. Click Add to page.
6. Provide a name. Then click Add.

The name is used for the portlet title, library display title, and library name. It can contain alphanumeric characters, spaces, and the following characters: $ -
7. Click **Save Draft**.
   This action copies the appropriate IBM Web Content Manager blog template library and places an instance of the web content viewer on the page.

After you publish the site with a blog library, click **Create Blog** and **Create Post** to add content. Click **Edit** to modify content. To delete a post, click **Delete**.

After you publish the site with a blog, click **Create Post** to add content. Click **Edit** to modify content. To delete a post, click **Delete**.

**Note:** Tagged web content displayed in the web content viewer is only available when there is a single instance of this portlet on the page. When you click on a tag result, the Tag Center broadcasts the information on what content should display in the viewer using a public render parameter. If you have multiple instances of web content being displayed in the web content viewer, then these instances will display the content that you tagged rather than display the original content of these instances.

**Related concepts:**

“Web content viewer best practices and limitations” on page 577
View some best practices and limitations for using web content viewers.

**Adding existing blogs or blog libraries to a page**

If you created a blog or blog library for another page and now want to use it again, add a web content viewer to the new page and edit its settings to point to the existing blog or blog library.

1. Create a project or access an existing project, and open the project in edit mode.
2. If you have not already done so, create a page for the existing blog or blog library or open an existing page.
3. Click the **Content** tab.
4. Click **All** in the list of categories, and search for the portlet **Web Content Viewer (JSR 286)**.
5. Add the web content viewer to your page by clicking the plus sign (+) or dragging the viewer onto the page.
6. Click **Save**.
7. Click the drop-down menu located in the web content viewer title bar, and select **Edit Shared Settings**.
8. Under **Content Type**, select **Content Item**.
9. Under **Content**, click **Edit**.
10. Click **Libraries**.
11. Click the library that contains the blog or blog library you want to add.
12. Expand the selected library to see the blog or blog library you want to use.
13. Select the blog or blog library and click **OK**.
14. Click **Apply** to save your changes to the web content viewer.
15. Click **Advanced Options**.
16. Ensure the **Broadcast links to** option is set to **None** or **This Portal page**.
17. Ensure the **Receive links from** option is set to **This portlet only** or **Other portlets and this portlet**.
18. Click **OK**.
Note: Tagged web content displayed in the web content viewer is only available when there is a single instance of this portlet on the page. When you click a tag result, the Tag Center broadcasts the information on what content should display in the viewer using a public render parameter. If you have multiple instances of web content being displayed in the web content viewer, then these instances display the content that you tagged rather than display the original content of these instances.

Related concepts:
“Web content viewer best practices and limitations” on page 577
View some best practices and limitations for using web content viewers.

Assigning blog access to users
The Portal administrator can assign Editor access to you if you need to create and manage blogs within the site. If you are given Editor access to a blog, you can create or modify posts in that blog. If you are given Editor rights to a blog library, you can create and modify blogs and you can create and modify posts in that blog library. If you have Manager rights to a blog, you can create, modify, and delete posts and delete comments in that blog. If you are given Manager rights to a blog library, you can create, modify, and delete blogs and you can create, modify, and delete posts and delete comments within the blogs.

For an example of assigning access to users, refer to the following instructions for adding users to the Editor role:
1. Go to Administration > Portal Content > Web Content Libraries.
2. Navigate to the library that contains the blog you want to manage and click Set permissions.
3. Click Edit Role for the Editor role.
4. Click Add to assign users or groups to the Editor role. Search for the users or groups that belong to this role.

For example, the settings in the following table allow all authenticated users to create new blogs, but restricts them to posting only in blogs they created. Use the Set Permissions and Library Resources options from the Web Content Library page to assign the following settings:

Table 112. An example of access settings for a blog

<table>
<thead>
<tr>
<th>Access settings</th>
<th>Group</th>
<th>Access level</th>
<th>Other settings</th>
</tr>
</thead>
<tbody>
<tr>
<td>Presentation</td>
<td>All Authenticated Portal Users</td>
<td>Contributor</td>
<td>Do not select &quot;Allow propagation&quot;</td>
</tr>
<tr>
<td>Templates</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Presentation</td>
<td>All Authenticated Portal Users</td>
<td>User</td>
<td></td>
</tr>
<tr>
<td>Templates</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Content</td>
<td>All Authenticated Portal Users</td>
<td>Editor</td>
<td></td>
</tr>
</tbody>
</table>

Viewing blogs and blog posts
You can view the blogs created in a specific library and you can view the posts in a blog by descending order. If you are a contributor you can view blog pages. By default, all portal users can view content in a blog or blog post once it has been created.

Perform the following steps to view blogs and blog posts:
1. Navigate to the page that contains the blog or blog library.
2. **(Blog library only)** Click **List of Blogs** to view all of the blogs created in a specific library. Blogs are listed in descending order from first to last and display the name of the person who created the blog (Editor) and the date and time when the blog was created.
   a. Move through the list of blogs using the page controls.
   b. Specify how many blogs to list on a page by choosing to **Show 10, 20, 50 or 100**.
   c. If you have Editor access, you can click **Create Blog**.
   d. Click the title of a blog to open the blog.

3. **(Blog or Blog Library)** View the posts, which are listed in descending order from first to last and display the name of the person who created the post (User) and the date and time when the post was created.
   a. Click **Latest Blog Posts** to view the most recent blog content.
   b. Move through the list of posts using the page controls.
   c. Specify how many posts to list on a page by choosing to **Show 10, 20, 50, or 100**.
   d. If you have Editor access, you can **Create Post**. You can edit and delete posts that you own.
   e. If you have User access, you can comment on a post.

After viewing posts, you can add comments to share your thoughts. To add a comment, click **Add a Comment**.

**Deleting blogs or blog libraries**
If you are the owner of a blog site, you have Manager access and can delete any blog, pages, comments, or posts on that site.

To delete an individual blog, open the project that contains the blog library in a project. To delete an entire blog library, use the Portal Administration page.

1. To delete one blog from a library, navigate to the page that contains the blog library.
2. Click **List of Blogs**.
3. Navigate to the blog library that contains the blog you want to delete.
4. Click **Delete** and **OK**.
5. To delete an entire blog library, go to **Administration > Portal Content > Web Content Libraries**.
6. Navigate to the library you want to delete and click **Delete library** and **OK**.

**Wikis**

Use wikis to share community content on a variety of subjects pertinent to your intranet and extranet sites. Wikis typically combine text with graphics and links to other wikis and web sites. You can monitor and manage your own wiki articles.

**Learn about the template libraries used by wikis**
Wikis use the template libraries provided by IBM Web Content Manager. Wikis use the Web Resources v70 library and the Wiki v70 library. The page hierarchy that is provided for wikis is the common one defined by the Web Content Manager template libraries.

**Note:** The Web Content Manager libraries described in this topic only work with the theme customizer used in the default theme. Customization to these libraries affects all wikis on the site.
Web Resources v70 library

The Web Resources v70 library provides the authoring and presentation templates for wikis. All wikis on the site use this single shared library. Changes to the main shared library affects all wikis.

Wiki template

When you add a wiki to a page, a copy of the Wiki v70 library is created using the name that you provided when adding the wiki to a page. The copy of the wiki v70 library is used to store wiki page content when new pages are added to this wiki.

Adding a wiki to a page

With Editor access to the portal or the portal page, you can add a wiki to a page to quickly create and edit content in-line.

Refer to the WebSphere Portal Family Wiki for an example of how you can modify your custom theme to enable the capability of adding blogs and wikis to a page.

1. Create a project or access an existing project, and open the project in edit mode.
2. If you have not already done so, create a page for the wiki or open an existing page.
3. Click the Content tab, then click Communications.
4. Click Wiki.
5. Click Add to page.
6. Provide a name. Then click Add.
   The name is used for the portlet title, library display title, and library name. It can contain alphanumeric characters, spaces, and the following characters: $ - _ . ! ( ) ,. If you use other characters, they are replaced by those that are supported by the IBM Web Content Manager library.
7. Click Save Draft.
   This action copies the appropriate Web Content Manager wiki template library and places an instance of the web content viewer on the page.

After you publish the site, click New Page to add content or click Edit to modify content. To delete a wiki page, click Delete.

Note: Tagged web content displayed in the web content viewer is only available when there is a single instance of this portlet on the page. When you click on a tag result, the Tag Center broadcasts the information on what content should display in the viewer using a public render parameter. If you have multiple instances of web content being displayed in the web content viewer, then these instances will display the content that you tagged rather than display the original content of these instances.

Related concepts:

Web content viewer best practices and limitations on page 577
View some best practices and limitations for using web content viewers.

Adding existing wikis to a page

If you created a wiki for another page and now want to use it again, add a web content viewer to the new page and edit its settings to point to the existing wiki.
Refer to the following steps only when you are adding a web content viewer portlet from the Page Properties menu and configuring this web content viewer to point to an existing wiki. To create a new wiki, use the Customize link to add a wiki to a page.

1. Create a project or access an existing project, and open the project in edit mode.
2. If you have not already done so, create a page for the existing wiki or open an existing page.
3. Click the Content tab.
4. Click All in the list of categories, and search for the portlet Web Content Viewer (JSR 286).
5. Add the web content viewer to your page by clicking the plus sign (+) or dragging the viewer onto the page.
6. Click Save.
7. Click the drop-down menu located in the web content viewer title bar, and select Edit Shared Settings.
8. Under Content Type, select Content Item.
9. Under Content, click Edit.
10. Click Libraries.
11. Click the library that contains the wiki you want to add.
12. Expand the selected library to find the wiki you want to use.
13. Select the wiki and click OK.
14. Click Apply to save your changes to the web content viewer.
15. Click Advanced Options.
16. Ensure the Broadcast links to option is set to None or This Portal page.
17. Ensure the Receive links from option is set to This portlet only or Other portlets and this portlet.
18. Click OK.

Note: Tagged web content displayed in the web content viewer is only available when there is a single instance of this portlet on the page. When you click a tag result, the Tag Center broadcasts the information on what content should display in the viewer using a public render parameter. If you have multiple instances of web content being displayed in the web content viewer, then these instances will display the content that you tagged rather than display the original content of these instances.

Related concepts:
“Web content viewer best practices and limitations” on page 577
View some best practices and limitations for using web content viewers.

Assigning wiki access to users
If you are the administrator, you have Manager access and can assign Editor access to other users who need to create and manage wikis within the site. If you are the owner of a wiki site, you have Manager access. As wiki site Manager, you can also delete any wiki page or wiki. If you have Editor access, you can create and edit any wiki site and wiki pages. All wiki editors can modify all pages of a wiki site.

If you are a contributor you can view wiki pages. By default, all portal users can view content in a wiki once it has been created.
For an example of assigning access to users, refer to the following instructions for adding users to the Editor role:

1. Go to Administration > Portal Content > Web Content Libraries.
2. Navigate to the library that contains the wiki you want to manage and click Set Permissions.
3. Click Edit Role for the Editor role.
4. Click Add to assign users or groups to the Editor role. Search for the users or groups that belong to this role.

Deleting wikis
If you are the owner of a wiki site, you have Manager access and can delete any wiki or wiki pages on that site.

You can delete a wiki and all its pages by deleting the wiki from a project.
1. Create a project or access an existing project, and open the project in edit mode.
2. Open the page that contains the wiki.
3. From the site toolbar, click More > Delete Page.
4. Click OK.
5. Publish the project.

Purging deleted wiki pages
When you delete a wiki posting, it is removed from the site, but remains in the delete view in IBM Web Content Manager. Users with Administrator access to libraries can purge these deleted wiki pages, which optimizes performance of the wiki library. Purging the deleted wiki pages removes all occurrences, including all versions. You cannot restore deleted items after you purge them.

To purge deleted wiki pages, complete the following steps:
1. Select Applications > Content > Web Content Management.
2. In the Library Explorer, select the wiki library that contains the pages that you want to purge.

   Tip: If you do not see your library, click Preferences > Configure > Library Selection to add the library to the selection list.
3. Select Item Views > Deleted Items.
4. Select the appropriate wiki page and click Purge. Click OK.

Item management
IBM Web Content Manager includes a range of features that enable you manage the web content items used in your system.

Working with folders
You use folders to group sets of item types into logical groupings.

Folders are used to group sets of item types into logical groupings. This is useful when you have large numbers of items in your library and want to distinguish between different groups of items within each item type view.

For example, you could create the following component folders to distinguish between different types of image components:
- Photos
You can create folders for the following item types:

- authoring templates
- presentation templates
- components

The folders you create are displayed in the authoring portlet within each item type view.

- Only a single item type can be stored per folder. For example, if you create a folder under the Presentation Template view, only presentation templates can be saved in that folder.
- You can create folders under existing folders allowing you to create a hierarchical structure of folders.
- You add items to folders selecting a folder as a location when you first create an item, or by moving or copying an item into an existing folder.

**Related tasks:**

- Creating a folder
  
  You create a folder when you need to store a set of the same type of items in a logical grouping.

**Managing versions of items**

You can configure your system to either automatically save a version of an item each time it is published, or allow users to select when to save a version of an item. You can restore items individually, or choose to restore a set of items within a library that either have the same label or were versioned at, or before, a specified date and time. Versioning is controlled through the Web Content Manager configuration services in the WebSphere Integrated Solutions Console and is enabled by default.

**Item level version control**

- If automatic versioning is enabled, a version is saved each time you save an item, or if the item is participating in a workflow, each time the item's state changes to published. You cannot save versions of draft items.
- If manual versioning is enabled, users with editor access or higher can manually save versions of published items. You can restore a version by viewing an item's version history and selecting a version to restore.
- Any hierarchical relationships to other items are not saved. For example, if you save a version of a site area that has child site areas and you delete the child site areas, the child site areas will not be restored if you restore the parent site area.
- If you have links to external resources within an item, a copy of the external item is not saved when a version is created. If you delete the external resource, you cannot use the Web Content Manager version feature to restore the external resource.

**Item names:** An item name is independent of version control. For example, changing the name of version 4 will also change the name of all versions of that item.
Library level version control

- You cannot choose to save a version of an entire library. Only those items that have had versions saved using item level version controls are saved within a library.
- You can restore a set of items from within a library that either have the same label or were versioned at, or before, a specified date and time.

You can apply a label to the most recent set of saved versions in a library at any time and then restore all the items with the same label at a later date.

Note: Labels that you apply to versions are not syndicated to subscribers.

Clearing version history

The clear versions tool can be used to clear version histories from items. It is configurable and can be used to clear versions within a specific date range, for specific item-types or libraries, or to keep only a certain number of most recent versions.

Related tasks:
“Clearing version history” on page 596
You use the clear versions tool to clear the version history of an item.

Related reference:
Web Content Manager configuration services
Configuration services for IBM Web Content Manager contain settings for the general operation of the web content system, including settings for messaging, pre-rendering, and searching.

Locked and draft items

As you work in collaboration with other users, you will encounter items that are locked by other users, either because they are being edited by another user, or a draft item has been created.

Working with locked items

When a user is editing an item, the item is locked to other users. This means other users cannot edit the item until the current user closes the item being edited. A lock symbol is displayed against items that are currently locked.

Users with administrator access can unlock items currently being edited by a user by selecting an item and then clicking the Unlock button.

Users with manager access can also unlock items so long as they have manager access to both the item and the library it is stored in.

Hierarchical item locking options

Locking of site areas, taxonomies, and categories are configurable and is not enabled by default.

When locking is enabled for site areas you cannot create any children within the locked site area. For example, if a site area is locked, you cannot create any site areas or content items within that site area until it is unlocked. This applies only to
direct children of a locked parent. Items that are descendants of a child of a locked
parent are not affected.

**Working with draft items**

When a new draft of a published item is created, a tick is displayed against the
published item.

**Deleting an item**

You cannot always immediately delete items. Sometimes you need to perform
steps to prepare an item for deletion.

To delete an item:
1. Select an item in an index.
2. Click Delete.

You can view a list of deleted items in either the Item Views > Deleted Items or
Personal Views > My Deleted Items views.

**Referential integrity**

When deleting items that are referenced by other items, you need to resolve any
references that is broken by deleting the item. A dialog opens listing the items that
cannot be deleted:

- Select an item from the list and click **Edit References**.
  - Select an item and click **Replace Reference** to replace the item you are
currently deleting with a different item. For example, if you are deleting a
text component that is referenced within a presentation template, you can
replace the reference to the text component you are deleting with a different
component.
  - Select an item and click **Clear Reference** to remove the reference to the item
you are deleting. For example, if you are deleting a presentation template that
is mapped to an authoring template, you can choose to clear the reference to
the authoring template.

**Note:**

- The referential integrity options available when deleting items depend on the
level of referential integrity applied to different items.
- There are occasions where only one option is available.
- Other items may not be able to be deleted at all.
- References to authoring templates can only be changed by using the **Apply**
  Template feature on the item itself.

**Restoring items**

You can restore a deleted item by restoring a version of an item. See Managing
Versions of items for further information.

**Purging items**

Users who have been granted manager access or higher to a library can purge
deleted items by selecting deleted items in the ‘All Items’ view and then clicking
**Purge.** This removes all occurrences of the selected item, including all versions. You cannot restore purged items.

**Viewing item references**

The "view references" tool assists you to view and manage all the links between items.

There are two ways to access the View References button:

1. Open an item, and click the View References button.
2. Select an item in an index and then click More Actions > View References.

To view references:

Click the View References button.

1. To view the items that reference an item, select Show references to item.
2. To view the items that are referenced by an item, select Show references from item.

**Personalization**

Portal Personalization provides automatic customization of website content for individual users and user groups.

Personalization can recognize a specific user based on a profile or can determine characteristics of a user based on previous purchases, products or pages viewed, and so forth. Personalization then selects content that is appropriate for that profile. If a person has a high salary range, Personalization can be configured to retrieve information about a commercial website premium product. If an individual belongs to a particular geographic region, content specific to that region may be targeted to the individual. The page is assembled with the proper personalized information, and the user sees her personalized page.

Personalization is composed of:

- **Personalization browser** - The Personalization user interface:
  - registers resource collections
  - authors rules, campaigns, and content spots
  - maps rules into content spots for a particular campaign

  Since objects are authored through the Personalization server, the Personalization browser can display rules in production as well as view rules in a staging environment.

- **Rules engine** - The rules engine executes rules created in the Personalization browser. A programming interface exists for Personalization to invoke rules, Personalization rules may be invoked through the Personalized List portlet, or rules may be invoked through Web Content Manager Personalization components. Rules associated with pages or portlets through Portal Administration are also automatically triggered.

- **LikeMinds Recommendation engine** - The recommendation engine evaluates recommendation rules created in the Personalization browser.

- **Resource engine** - The resource engine resolves the queries produced by rules into content pieces to be returned. Content for Personalization is created and
approved using whatever content management tool you choose, or may come from an LDAP or any other database. Content is accessed via a set of Resource Collection classes.

- **A logging framework** - The logging framework is used to record information about website usage to the feedback database and the recommendation engine. It is entirely up to the site developers to decide what information is logged.

The engines are sometimes collectively referred to as the Personalization run time server.

The engine identifies the particular user. Personalization retrieves that person’s profile. For example, a person may have a salary range included in her profile. Personalization then selects content that is appropriate for that profile. If a person has a high salary range, region code, or other information, Personalization can be configured to retrieve information about a commercial website premium product. The page is assembled with the proper personalized information. The user sees her personalized page.

**Types of Personalization**

There are three types of Personalization:

**Simple filtering**
- A site displays content based on predefined groups of site visitors. For example, if a site visitor is in the Human Resources department, the site provides access to URLs containing Human Resources policy manuals.

**Rules engines**
- In a rules based system, the site owner defines a set of business rules which determine what category of content is shown when a certain profile type visits the site. An example would be: Display all four wheel drive SUVs to visitors in the northeast in the 21 to 35 age group.

This approach has the advantage of driving the site’s behavior with the business objectives of the site owner. The site owner is usually the owner of a marketing campaign or some other business manager.

**Collaborative filtering**
- A site visitor rates a selection of products, explicitly or implicitly. Those ratings are compared with the ratings offered by other visitors. Software algorithms detect similarities. For example, a visitor receives book recommendations based on the similar purchases of others.

**Rules versus collaborative filtering**

When complex filtering is required, a rule-based system may work better than collaborative filtering, and vice versa. The following table details examples where one type of personalization is better than the other.

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Which filtering type to use</th>
<th>Reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>If the number of items offered and users who purchase them are rather low.</td>
<td>Rules</td>
<td>Very little room to compute user similarity necessary for collaborative filtering.</td>
</tr>
</tbody>
</table>
Table 113. When to use rules filtering versus collaborative filtering. (continued)

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Which filtering type to use</th>
<th>Reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>If price points are high or purchasing frequency is low.</td>
<td>Rules</td>
<td>Finite, limited arenas - collaborative filtering fails because of the inherent lack of diversity.</td>
</tr>
<tr>
<td>If there is a pre-existing dependency between items. Example: Disability policy required for homeowner</td>
<td>Rules</td>
<td>Recommending a disability policy just because collaborative filtering says many others &quot;like this user&quot; also bought a policy is incorrect--one must have the homeowner policy first.</td>
</tr>
<tr>
<td>If number of items offered and users who purchase them are rather high.</td>
<td>Collaborative</td>
<td>Cannot write rules covering all items.</td>
</tr>
<tr>
<td>If price points are low, all quite dissimilar, or the products offered have a wide range of user appeal.</td>
<td>Collaborative</td>
<td>The wide variance fits the collaborative filtering approach. Collaborative filtering also lowers the risk of making &quot;bad&quot; recommendations.</td>
</tr>
<tr>
<td>When not much information is gathered about the user, but the user can be identified, possibly by a login or cookie.</td>
<td>Collaborative</td>
<td>In this case, user attributes on which to base rules may be lacking. Collaborative filtering can compare the user's experiences on the site to other users.</td>
</tr>
</tbody>
</table>

Related concepts:
“Developing a personalized portlet” on page 455

This exercise demonstrates how to use Personalization features of WebSphere Portal and Rational Application Developer to build your first personalized portlet. Your final result is a working portlet that uses Personalization rules and content spots to display personal news based on user attributes (or profiles).

Related tasks:

Technotes for personalization

How a site is personalized

Use this topic to understand how to define a personalized list of new articles for a website, such as a section of an intranet site for targeted employee bulletins or where the content of the site is tailored to the particular user.

Developing a personalized portlet or website

1. A web developer defines an area of a site that needs Personalization. This area may be a personalized list of new articles appearing in a website, a place on an intranet site for targeted employee bulletins such as changes to benefit plans, a product list on a commerce website, or any other place where the content of the site should be tailored to the particular user.

2. A set of Resource Collections and Application Objects are defined. Together, these objects make up the business vocabulary that represents the terms and objects upon which Personalization decisions are based. These objects may be defined in the web page by pointing to an IBM Java Content Repository item
type. These objects may be generated through a set of Personalization wizards provided with IBM Rational® Application Developer; or they may be developed according to a set of public programming interfaces.

3. The Resource Collections and Application Objects are registered to the Personalization server through the Personalization browser by importing .hrf files. These files define Resource Collections. The developer can also manually create the Resource Collection and Application Object references in the Personalization Browser.

**Note:** The classes used for the Resource Collections and Application Objects must be on the classpath for both the application being personalized and for the Personalization browser web application. One way to achieve this task is to use a shared library placed either on the application server, or on the individual web applications. If you are using stock resource collections provided with IBM WebSphere Portal, such as the Portal User Resource Collection or the Java Content Repository Resource Collection, these classpaths are already registered properly.

4. Programmers now use the objects and terms defined through the Resource Collections and Application Objects to write rules and map those rules to content spots using campaigns.

5. Portlet and website developers may either configure a Personalization Rule Display portlet to show the rule or content spot or may call into the Personalization programming interfaces to execute the appropriate rules or content spots which the programmers have defined.

**Note:** The Content Spot provides a way for the site developers to develop personalized pages before the rules are authored as well as a way to more loosely tie a particular rule to a page. It is then up to a programmer to "map" the Content Spot to a Rule using a Rule Mapping within a Campaign in a Personalization browser.

### Using Personalization Rules

1. A user navigates to the page containing Personalization rules or content spots.
2. The application invokes Personalization to find content or make decisions.
3. Personalization identifies the correct rules to execute. If a Content Spot with the name given to execute is not found, a rule is looked for directly with this name.
4. The Personalization server searches for all Rule Mappings for the Content Spot. The server looks for Rule Mappings which have started, but not yet ended.
5. The Rule Mappings are ordered based on the priority and split values. The rules associated with each mapping are executed in turn until a rule returns content.

**Note:** It is possible to configure Personalization so that only the first rule (higher priority) will get executed.

6. For each rule executed, Personalization retrieves the user's profile and evaluates the rule to select the content which meets the conditions of the rule. The rules engine will invoke the resource engine as necessary to retrieve content pieces.
7. The content is returned to the web page, and displays for the user.
Personalization terms

The concepts and principles for working with Portal Personalization require an understanding of terminology.

Resources, resource instances, and resource collections

Before you can personalize IBM WebSphere Portal resources, you need to understanding the terms for portal objects stored in the content repository.

A resource is a Java class that defines the properties of a user or content object. In database terms, it is analogous to the database schema that defines the column names and types for a database table. Resource classes must implement the com.ibm.websphere.personalization.resources.Resource interface.

A resource instance is an instance of the resource class. Again, using a database analogy, the resource instance is similar to a row of a database table because it contains actual values for each property defined by the resource.

A resource collection is a Java class that represents and allows access to a collection of resource instances. It is similar to a database table with a fixed schema and a number of rows. Resource collection classes must implement the com.ibm.websphere.personalization.resources.ResourceDomain3 interface. Rational Application Developer provides a wizard that can create resource collections that store data in SQL databases or LDAP repositories. The classes that can make up the resource collection are as follows:

Resource Class
  An instance of com.ibm.websphere.personalization.resources.

Resource Manager Class

Domain Class

Translator Class
  An instance of com.ibm.websphere.personalization.resources.AuthIDTranslator.

For more details, refer to the Javadoc API documentation for Personalization APIs. You can provide your own implementation of these classes or use the RAD Personalization Wizard to generate classes that query against SQL or LDAP repositories.

While resources, resource instances, and resource collections are easily mapped to familiar database concepts, it is important to note that the actual content store they refer to does not have to be a database table. It can be a file system, an LDAP repository, an XML store, or virtually any content store accessible by Java.
User resources
Your Web site visitors want to quickly access the Web content that meets their needs and interests. The needs and interests of the site visitors are stored as properties in the user profile data store.

Many sites obtain the user needs and preferences by using an HTML input form. The input form includes information that seldom changes (such as the user name and address) and information that must be updated over time (such as gift wish lists). In addition to explicit information provided by the site visitor in the HTML input form, a business might also programmatically update the user profile data store with information obtained from other sources. These sources could include an analysis of a user’s actions on the Web site and data from a user’s non-Web interactions with the business.

Although users are unique, personalizing Web content does not often require creating totally unique content for each user. Users can be profiled or grouped into categories that facilitate personalization. For example, on an internal Web site, certain information might be appropriate for managers, while other content may be suitable for salespersons.

Note: The Portal User Collection resource collection included with Personalization cannot currently be used to select lists of users in select content rules. The resource collection may be used to profile the user in a profiler rule. This applies to both the user resource collection which is installed with Personalization as well as any custom resource collections created with the manager class com.ibm.websphere.personalization.resources.wps.UserManager.

Content resources
Web content resources consist of data that is formatted and displayed on Web pages. Examples include news articles, products, statistics and educational materials.

The content is displayed when a user requests a JSP or servlet that dynamically generates and formats a Web page in the appropriate format, such as HTML or XML. When personalizing your Web site, you only have to customize the content you want to selectively display (or suppress) based on the user. For example, a given page might consist of 40% static content and 60% variable content based on particular user characteristics or business conditions, while other pages might be 100% static content.

Content resources can be file content or structured content.

Attribute Based Administration
Attribute based administration provides a facility to customize the layout of a page for individual authenticated users by using rules to show or hide pages or portlets. This implementation tells the portal to show or hide pages and portlets based on dynamic characteristics that are determined at runtime. Attribute Based Administration is only available for authenticated users. For anonymous users, all pages are shown.

Portal Personalization rules can be used to control whether a page is displayed in the site navigation; this is managed by choosing a rule appropriate for the user attribute you want to enable to see the page. If the rule returns true, the page or portlet will be shown, otherwise, it will be hidden. If Personalization is not installed or is not enabled in the properties settings, you will not see this option.
Access Control and Visibility Rules

Access Control and Visibility Rules both impact what appears on a portal page or portlet. Access Control determines what a user is allowed to see on a page or in a portlet. In order to see pages and portlets, a user must be explicitly defined as a member of the group that has access. The groups are arranged in a hierarchy and each group is assigned roles such as administrators or editors.

Visibility rules determine what a user will see, or what has been targeted towards a user, and Access Control is based on group membership, visibility rules use any type of information, including LDAP attributes, or time of day. For example if you want to hide a portlet for an individual in a certain geography, store the location as an attribute in LDAP, and assign a visibility rule hiding the portlet. For example, a user may have access to the revenue figures for all divisions for the entire year, but these figures should not be displayed prominently except when they are first released. For a week after the figures are first released, the figures for the employee's division should show prominently on their home page. The visibility rule hides figures for divisions the employee is not in and only shows the employee's figures the week after they are released.

Access Control takes precedence over visibility rules. You must have access to a page or portlet prior to applying visibility rules. Access Control also determines if a page or portlet will be returned in a search; if a user does not have access, he will not be able to see the portlet or page in search results. If a user has access to a portlet or page, but has the visibility rule set to hide the page or portlet, it will show up in search results.

Assigning attribute based administration rules to pages and portlets:

Attribute based administration rules can be assigned manually in the portal in the Edit Properties and Edit Layout portlets, or through the XML configuration interface. The rule must be present on the system in order to assign it to a page or portlet. You can usually use Personalization Publish to accomplish this.

- In order to map rules, non-administrator users must be given at least user access to the rule that is being mapped and edit access to the page or portlet where the rule is being mapped.
- Pages or portlets on derived pages show an inherited visibility rule if no rule is defined for them. You cannot clear the inherited visibility rules on derived pages.

Using the XML configuration interface, you can assign a rule to a page or portlet. Set the parameter com.ibm.portal.navigation.rule to indicate the rule to assign to the page or portlet. The value of the parameter should be the unique id or UUID of the rule. The UUID can be found by exporting the rule in the Personalization user interface and inspecting the exported XML for the jcr:uuid parameter.

For example, to assign a rule with the UUID 7ce9d5004ee31f41b0d3b944c9f7965c to a page or portlet, add the following parameter to the content-node in the XML access script:

<parameter name="com.ibm.portal.navigation.rule" type="string" update="set"><![CDATA[7ce9d5004ee31f41b0d3b944c9f7965c]]></parameter>
Related concepts:
The XML configuration interface
Use the XML configuration interface for exchanging portal configurations.

Changing the error condition behavior:

If an error occurs when locating or using a rule assigned to a page or portlet, by default that page or portlet will be hidden. If a user or expected application does not exist, the system will continue. This behavior may be appropriate, but in a development environment, you may need to change this behavior for testing purposes. Update the PersonalizationService.properties file to override this behavior globally.

The property rulesEngine.visibilityDefault specifies whether a page or portlet renders if there is a problem with the assigned rule. Changing the property value to show means the page or portlet will display even if the assigned rule cannot be found or if there is a problem with the rule.

The property rulesEngine.throwObjectNotFoundException specifies what happens if an object such as a user is not found when needed during rule execution. This may occur when Personalization cannot find the current user or when an expected application object does not exist on the session or request at the expected key. When set to false, a null user or object is not treated as an error but is instead only printed to the logs as a warning. Personalization will continue as if the requested attribute of the null object is itself null.

For example, if no user object is found and rulesEngine.throwObjectNotFoundException is set to false, a rule such as Show page or portlet when user.name is null would return show. A null user is treated as if the user name is null. However, if no user object was found and rulesEngine.throwObjectNotFoundException is set to true, this same rule would throw an exception. If this rule was used to determine the visibility of a page or portlet, the ultimate result would depend upon the value of the rulesEngine.visibilityDefault property, which would decide what occurs if an exception is thrown during processing of a rule in attribute based administration.

1. Open the PersonalizationService.properties file. This file is located in the following directory:

   wp_profile_root/PortalServer/config/config/services/
   PersonalizationService.properties

2. Find the rulesEngine.visibilityDefault property.

3. Set the value of this property to show to enable portlets or pages to render if an error occurs.

4. Find the rulesEngine.throwObjectNotFoundException property.

5. Set the value of this property to true to throw an exception if the object is not found.

Rules
Rules are used to define how your Web site interacts with individual and groups of Web site visitors. Rules are composed of easy-to-read logic statements that, in their final form, specify how to evaluate various conditions and what actions to take based on those conditions.

Actions:
Actions use simple evaluation statements to select content to use or display, or to set information. Basic arithmetic calculations (addition, subtraction, multiplication,
division) that follow order of operations can be used on either side of the evaluation; parentheses are not supported. Actions can be combined with profilers into bindings.

The following action items are supported:

**Select actions**
Select data or content. Select actions retrieve data from a data store, typically for display on a web page. Select actions also can be used within bindings to exclude certain content, that is, filtering a subset of returned content from a superset.

**Update actions**
Update content or objects on the request. Update actions cannot select content. Update actions are used to store content or data in the user profile, an application object, or other data stores.

**Email actions**
Send email messages using a web page as the body. Email actions cannot select content. An email action sends an email message to a recipient or list of recipients. An email action assigned to a content spot sends an email message when the content spot is triggered. For example, at the time a website visitor views a page with the content spot.

The email action editor allows the fields that identify the recipients (primary, copied and blind copied), the subject line, and the sender to be identified by either explicit text or by resources attributes. The email body is a separate file, such as a text file, an HTML file, or a JSP, that must be accessible from the email server via a URI. An emailed JSP can contain content spots for personalizing the email message for the user who triggers the content spot.

Although email actions differ somewhat from email promotions, both have prerequisites that must be in place and working before email can be sent.

*Example: Simple select content action:*

View an example for a simple select content action.

The select content action shown here, Get Bank News By Role, queries all records within a content resource entitled News and returns those marked as being for the current user's role.

The content resource YourCoNews represents news articles in the data store. Each record has several different fields (for example, Title, Abstract, Author, Body), including a field entitled Role. In the data store, this field is marked to indicate the role of the visitor to whom it applies.

**Simple select content action**

```
Select NewsArticle
  whose Role is current User.Role
  order as is
```

*Example: Simple update action:*

View an example of an update action that is part of a Web site that allows visitors to manage certain information and preferences about themselves.
When executed, the following update action will write to the fields (Income Group, Role) in the data store for the record associated with the current user (the current Web site visitor), using data contained in the current user's session variables, such as `incomeGroup` and `role`.

**Update action**

```
Update
  current Portal Users.Income Group set to current Session.incomeGroup
  current Portal Users.Role set to current Session.role
  current Portal Users.Last Name set to current Session.lastName
  current Portal Users.Title set to current Session.title
```

**Example: Simple email action:**

This email action rule example is typical of one that might be used after a website visitor submits a form indicating interest in an item or service.

The email action rule can be attached to a content spot that, when triggered by the visitor viewing the page with the spot, sends the email indicated by the `bodyURI` field to that visitor. This email is also blind copied to someone within the sample company.

**Email action rule**

```
To: current Portal Users.Email Address
From: Rates@YourCo.com
    bcc: Mortgage_Broker@YourCo.com

Subject: Today's Mortgage Rates
Body URI /email/mortgage-rates.jsp
```

**Related information:**

[“Email administration” on page 367](#)

View the steps you need to complete before your run time server can send personalized email.

**Profilers:**

Profilers are typically used to categorize an individual (usually the current site visitor) according to his or her user properties.

Besides a user's properties, profilers can be used to define other conditions based on such factors as the current date and time, the type of browser the visitor uses, or other implicit and explicit application object properties. Profilers can also make decisions based on the current user's session attributes and request attributes and parameters, along with category and action counts.

Profilers can be constructed to define the conditions of arbitrarily named profiles, or can be defined in terms of other profilers. For example, you can create a profiler that will evaluate as true if a profile is in any, all, or none of a group of other profiles.

**Example: Simple profiler:**

View an example of a simple profiler that determines whether confidential company news articles will be shown to the current Web site visitor.

The profiler `User Clearance` is based on a user resource called `Personnel`. When the profiler was created, the name `User Clearance` and the settings, `Confidential`
and Regular, were arbitrarily defined for later reference within bindings. One side of the comparison line (current Personnel.Role) refers to a user resource named Personnel created from a Personnel table in a database. Role is a mapped value, defined when the resource was created, that points to the Personnel.Role column in the Personnel table. The values in the Role column in the database are either Employee, Executive, or Manager.

This completed profiler is used within a binding as a means of determining whether confidential company news articles will be shown to the current Web site visitor.

Simple profiler

User Clearance is

  Confidential when
  current Personnel.Role is Executive or
  current Personnel.Role is Manager

  Otherwise Regular

Example: Nested profiler:

View an example of a nested profiler.

A nested profiler is true if any, all, or none of the profilers within the profiler are true. You can categorize a condition as a combination of other conditions. For example, a Web site visitor can be profiled as a young male if a pre-existing gender profiler classifies the visitor as male, and a pre-existing age profiler classifies as the visitor as being in his teens or twenties.

Nested profiler

AgeGenderProfiler is

  YoungMale when
  GenderProfiler is Male and
  AgeProfiler is any of Teenager or Twenties

  Otherwise NotInTargetAudience

Example: Category Count (implicit profiling):

Get an overview of implementing category counts from a profiler that will contain profile definitions for movies, cooking, and sports; this overview sets the profile for the category with the highest count.

For the Category Count example, assume a repository of articles on sports, movies, and cooking is available for the user to view. Each time the user views an article, a record is logged to show his or her preference for that topic. For this to occur, each article must be a JSP that implements category beans. For example, the following code would appear on a sports article:

```html
<jsp:useBean class="com.ibm.wcp.analysis.beans.Category" id="category" scope="session">
  <jsp:useBean>
  <\category.log(request, "Articles/Sports");
```

These values were typed into the Attribute text field during the creation of this profiler after selecting current Category Count.

Note: A complete version of this profiler will contain profile definitions for movies and cooking for the cases where those category counts are greater than sports and each other.
Category count

ArticlePreference is
Sports when
- current Category Count.Articles.Sports is greater than current Category Count.Articles.Cooking and
- current Category Count.Articles.Sports is greater than current Category Count.Articles.Movies

Example: Browser capability:

The browser capability stock object allows you to profile the browser the current Web visitor is using to view your site. View an example profiler that checks the browser version to determine whether it is supported.

When using the browser capability stock object, there is a finite list of attributes available, but you must type the values for one side of the evaluation. The browser capability script files and the SinglePixel.gif image must be properly configured in the Web application.

The following example profiler checks the browser version to determine whether it is supported. Other possible checks include available plug-ins, whether Java is enabled, and whether cookies are enabled.

Check Browser is supported when

```
{
  (current Browser Capability.BrowserType is Netscape and
   current Browser Capability.FullVersion is greater than or equal to 6.2
  ) or
  (current Browser Capability.BrowserType is Internet Explorer and
   current Browser Capability.FullVersion is greater than or equal to 6.0
  )
}
```

Otherwise unsupported

Example: "Count of" (quantifiable condition):

View an example demonstrating the use of "Count of".

A quantifiable condition profiler is similar to implicit profiling because a profile is defined according to numbers of items. Quantifiable condition profilers do not require the use of logging beans, but require attributes of resources that will be quantified to be organized uniformly.

In the following profiler, counts are made for items in the session object shoppingCart used by the user resource Shopper. Here, shoppingCart is analogous to a table in a database and color would be a column of data. Each item within the table would be a row. For example:

<table>
<thead>
<tr>
<th>Item</th>
<th>quantity</th>
<th>size</th>
<th>color</th>
<th>price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gadget</td>
<td>1</td>
<td>L</td>
<td>red</td>
<td>$1.99</td>
</tr>
<tr>
<td>Gizmo</td>
<td>3</td>
<td>S</td>
<td>green</td>
<td>$0.95</td>
</tr>
</tbody>
</table>

Quantifiable condition profiler for the session object shoppingCart

ColorPreference is
Red when number of items matching (Shopper.shoppingCart.color is red) is greater than 5
Green when number of items matching (Shopper.shoppingCart.color is green) is greater than 5
Example: Request attributes and session attributes:

Request values passed to the JSP or stored in the HTTP session can be referred to from rules. To refer to them in a rule, you must know the variables and their possible values.

The following example profiles news articles as read or unread by the current user, based on the page request generated by a form on a JSP.

**Request attributes and session attributes**

User News is
- ReadNews when
  - current Request.ProcessNews is Read
- UnreadNews when
  - current Request.ProcessNews is Unread
- UnreadAllNews when
  - current Request.ProcessNews is UnreadAll

Example: Arithmetic operation:

View an example of a profiler that uses arithmetic operations.

To build an operation, select a combination of Resource.Attribute and other operands. You can do this on either side of the evaluation.

**Order of operations is in effect but parentheses cannot be used**

AgeProfiler is
- Youth when
  - current Date.Year - current User.BirthDate.Year is less than or equal to 25
- Adult when
  - current Date.Year - current User.BirthDate.Year is between 25 and 65
- Senior when
  - current Date.Year - current User.BirthDate.Year is greater than or equal to 65

**Bindings:**

Bindings combine actions and profilers to specify actions to perform when defined conditions are encountered. Returned content can be sorted or filtered prior to display or use. Actions can be set to be performed by always, exclude, and otherwise blocks. In addition, the total number of items used can be limited.

The always and exclude blocks will always be performed no matter what the outcome of the profiler execution, but the otherwise block will only be run if none of the profilers match. An otherwise block works the same as an always blocks in that it adds items to the results, but for an exclude block, you actually choose an action that defines items that you don't want to include in the results. For example, on a shopping site, you may want to exclude items that have already been purchased.

*Example: Simple binding:*

Because bindings couple the conditional processing of a profiler with the functional power of an action, the simplest form of a binding works like a conditional “if-then” clause.

Consider a simple binding example. If the current user is not a previous customer, then show a limited number of current offers. If the current user is a known customer, then show offers appropriate to their status level.
For example, the profiler Customer Type is used to check whether the current user is a known customer. If the customer is not a known customer, the action Get Limited Number of Offers is executed. If the profiler indicates the user is a known customer with a status of Gold or Platinum, then a different action is executed and different offers are retrieved for display.

**Simple binding**

When Customer Type is
   Not A Customer
      do Get Limited Number of Offers
   Gold or Platinum
      do Get Offers For User
      order as is

*Example: Multiple profilers and optional actions:*

View an example that demonstrates the use of a conditional "if-then" with an additional clause.

Consider an example: If the current user does not have Confidential status, then the action GetNonConfidentialNews is executed. The same results could be achieved in this example by placing the GetNonConfidentialNews action under Otherwise because these are the only two profiles possible within this profiler.

The action field under Otherwise remains as is. Since the UserClearance profiler places every user into one of two categories (Regular or Confidential), any action placed here would never be executed.

The GetSiteNews rule will always be executed. Any content the rule retrieves from the data store is added to the total return set.

The GetNewsAlreadyRead action works like any other action because it retrieves content from the data store. However, when the action is placed under Exclude, any content retrieved by this action is removed from the total return set.

*Note:* It must be possible to indicate an article has been read by a given user. When you click the Select Action menu, you will only see rules that are assigned a Select Action type. Binding rules are also Select Action type rules. Once a resourceCollection type is set for the binding, all of the action rules will be locked and will use the same collection type.

The order of the total return set is randomized and the number truncated to no more than 10 items. This effect takes place each time the rule is executed, so the news articles displayed on the Web page will change from page view to page view.

**Conditional "if-then" with an additional clause**

When UserClearance is
   Confidential
      do GetConfidentialNews
   Regular
      do GetNonConfidentialNews
Always
   GetSiteNews
Exclude
   GetNewsAlreadyRead
   order randomly
   show 10 items
Example: Nested bindings (simple):

When creating a binding, it is possible to use a binding in any of the do action areas. This is known as a nested binding.

In this example, the Always action (Get Top Products) in Get Products By Location is actually another binding. When the nested binding is placed with Always, it has the effect of the boolean operator or.

For example: The total rule returns content that meets the conditions in the earlier part of the binding or that meets the conditions in the later part.

Note: It is possible for a nested binding to contain nested bindings.

Get Products By Location:

**Nested binding**

When User Location is Lab
   do Get Test Products
   Factory
   do Get Available Products
   Field
   do Get All Products
   Otherwise
   do Get Future Products
Always
Get Top Products
order as is

Get Top Products:
When User Role is Manager
   do Get Top Selling Products
   Exec
   do Get Top Selling Products
   Employee
   do Get Top Overstocked Products
   order as is

Example: Nested bindings (advanced):

View an example that demonstrates advanced nested bindings.

This example is similar to the Nested bindings (simple) example, with the exception of the addition of the binding Get Top Products to the actions performed when the User Location profiler is Lab. Multiple actions can be grouped by selecting multiple actions or bindings simultaneously within the rule editor. Selecting multiple actions or bindings here has the effect of the boolean operator “and”, which returns the intersection of the data sets.

For example: the current user location must be in the lab and an executive. Therefore, in addition to test products, executives and managers in the lab will receive information about the best selling products, and employees at the lab will get the most popular overstocked items. Factory or field workers will not see either best selling or most popular overstocked products.

Modified Get Products By Location:
Advanced nested bindings

When User Location is
Lab
   do Get Top Products and
   do Get Test Products
Factory
   do Get Available Products
Field
   do Get All Products
Otherwise
   do Get Future Products
order as is

Recommend Content:

You use Recommend Content rules (also referred to as recommendation rules) to recommend content to your Web site visitors. Recommendation rules, powered by LikeMinds, recommend content based on users' past interactions with your Web site.

When creating a Recommend Content rule, you specify one of three recommendation methods. The recommendation methods are:

how the current user navigated the site
   This method is associated with the LikeMinds Clickstream Engine.

preferences explicitly expressed by the user
   Use this recommendation method to generate recommendations based on users' ratings of items. This method is associated with the LikeMinds Preference engine.

   Items map to a piece of content and are represented by resources and resource collections. Your Web site captures ratings using the Rating bean. The Rating bean collects the rating, the item resource, and resource collection, and then logs the data for LikeMinds to use later.

association with content returned from a rule
   Use this recommendation method to generate recommendations based on market-basket analysis. This method associates items that a current Web site user has interest in (such as an item in their shopping cart) with items that others users have had interest in or have purchased. This method is associated with the LikeMinds Item Affinity engine.

   Item affinity rules make use of the LikeMinds transaction data being collected. They offer a method for generating recommendations from a known set of resources (actually the results of another rule returning the same resource type).

Note: Before using Recommend Content rules, check with your system administrator to see which LikeMinds engines are configured and running on the production run-time server.

Previewing a Recommend Content rule:

To preview results, the production LikeMinds database must contain data, including items, users, and transactions (ratings or actions). The problem of initial data priming is commonly called coldstart.

Visibility Rules:
Visibility rules determine what a user will see, or what has been targeted towards a user. Visibility rules can be assigned to pages and portlets and will be triggered automatically by the portal as needed.

Visibility rules use any type of information, including LDAP attributes, time of day, or session information. For example, if you want to hide a portlet for an individual in a certain geography, store the location as an attribute in LDAP, and assign a visibility rule hiding the portlet. A user may have access to the revenue figures for all divisions the entire year, but these figures should not be displayed prominently except when they are first released. For a week after the figures are first released, the figures for the employee's division should show prominently on their home page. The visibility rule hides figures for divisions the employee is not in and only shows the employee's figures the week after they are released.

Visibility rules can be assigned to pages and portlets and will be triggered automatically by the portal as needed. Through the APIs, visibility rules behave like profiler rules where the only two possible profiles are show and hide. This allows visibility rules to be invoked programmatically and used in any custom application just as you would call a profiler rule.

**Related concepts:**

"Attribute Based Administration" on page 348
Attribute based administration provides a facility to customize the layout of a page for individual authenticated users by using rules to show or hide pages or portlets. This implementation tells the portal to show or hide pages and portlets based on dynamic characteristics that are determined at runtime. Attribute Based Administration is only available for authenticated users. For anonymous users, all pages are shown.

**Example: Show page or portlet:**

View an example of a visibility rule, Show Page that shows the specified page or portlet only during the specified time period, and only to users in the Midwest. For all other dates and users, the page or portlet is hidden.

**Visibility rule**

Show page or portlet when

- current Date.Date is between December 12, 2006 and December 19, 2006
- current LdapUsers.Geography is Midwest Region

Otherwise hide

**Rule elements:**

Learn about the options in the rule editor for the different types of rules.

**Arithmetic expressions:**

Arithmetic expressions allow you to perform mathematical operations on resource attributes as part of your rule. When you choose this option, you can select multiple resource attributes, values, and operators (addition, subtraction, multiplication, or division) to use between them.

An example use of an arithmetic expression is a profiler that profiles Web site visitors according to age. In the data you record for each visitor, it is more practical to store date of birth (which does not change), than to store age. In the evaluation
in the profiler, you can use an arithmetic expression to calculate the visitor's age by subtracting the current user's year of birth from the current year (current Date.year).

Arithmetic expressions are calculated according to traditional order of operations (multiplication and division are calculated before addition and subtraction. For example, 3+2^2–1/2 evaluates to 6.5). It is not possible to group expressions using parentheses.

Count of (quantifiable conditions):

Create an evaluation based on a count or tally of attributes that meet your criteria.

In the Specify a Resource Attribute drop down, you have the option to select Use Number of Items in List. Selecting this option allows you to create an evaluation based on a count or tally of attributes that meet your criteria. When you select this option, you must select a resource and the attribute of that resource that are to be tallied when the rule executes.

Note: This option can only be used in profiler and visibility rules.

Because the tally is made at the time the rule is triggered, it could produce different results at different times during the session if used on an application object or a current resource. For example, you might profile a visitor's color preference as red by creating an evaluation that checks to see if the number of red items in the user's shopping cart is greater than 5. The rule syntax for this evaluation could be:

Count of (shoppingCart.item.color is red) is greater than 5

Although you can make counts of any data type, the tally must be compared against a value or resource attribute that has a data type of number, decimal number, or integer.

Current Action Count or Action Name:

Current action count, like current category count, is a way to base a profiler or rule on the number of times a Web site visitor has performed certain actions. These actions must be logged using the action logging beans in order to be accessible by the rule. Current action name inspects the names of the actions logged.

Current Browser Capability:

Browser Capability is an application object that allows you to profile a Web site visitor based on the attributes or capabilities of the browser being used. When applicable, it appears in the rule editor as an option when you select Resource.Attribute.

Purpose

Browser Capability currently supports these attributes:

AcceptLanguage
  Returns the value of the header 'accept-language' from the request object.

AcceptMimeTypes
  Returns the value of the header 'accept' from the request object.
Agent  Returns the value of the header 'user-agent' from the request object. This is
a lowercase string that contains information about the client software,
usually the browser name or version.

BrowserType  Returns the browser type. Choices are available for supported browsers.

FullVersion  Returns the version of the browser to one point of precision. For instance,
6.1 and 6.1.1 are both returned as 6.1.

MajorVersion  Returns the first digit of the browser version. For instance, 6.0, 6.1 and
6.1.1 are all returned as 6.

Current Date:

The current Date resource contains several attributes you can use for comparison
(date, day, month, time, timestamp, weekday, and year). To set values, you may
enter a specific value or reference the value of a specific attribute of the same type.

Current Request Attributes:

Use current Request Attributes to inspect request attributes which can be set on
the current JSP. You must know the name of the request attribute to use it in a
rule. This request is the request passed into the content spot executing the rule. For
eexample, you would use the portlet request to set the current Request attribute for
portlets. The portlet request is not shared among portlets. For jsp's directly within a
Web application, the current Request Attribute is the HTTP request of the Web
application.

Consider the following code that can be inserted into a JSP to set a request
attribute:
<% request.setAttribute("user", userObject); %>
where userObject is of any Object type

An example rule condition constructed to evaluate the previous example might be:
when current Request Attributes.user is equal to rob

All data types are supported.

Current Request Parameters:

Use current Request Parameters to inspect data contained within the query string
(the variables and values that appear after a question mark on a URL).

To understand current Request parameters, consider the example URL
http://hostname/page.jsp?var1=rob&var2=expert. In this example, the request
parameters are var1 and var2. Typically these are passed by GET and POST
methods associated with forms. You must know the name of the request parameter
that is passed by the page to use it in a rule.

Given the following <jsp:forward> command:
<jsp:forward page="/servlet/login" />
<jsp:param name="user" value="rob" />
</jsp:forward>
an example rule condition constructed to evaluate this example might be:
when current Request.user is rob

Only data types Text and List are supported.

*Current Session Attributes:*

Use *current Session Attributes* to inspect parameters stored within the current session object for the Web site visitor. You must know the name of the parameter to use it in a rule. All data types are supported. The current session object is the session associated with the request passed into the content spot executing the rule. For example, the current session object is the portlet session, which is unique to the portlet. For jsps within a WAR, the current session object is the the http session.

*do Action:*

Within a binding, you can couple actions with profilers so certain tasks are performed when certain conditions are met. You can also indicate actions to be done under other conditions. Use the *Do action, Otherwise do action, Always do action*, and *Exclude do action* elements.

*Do action* allows you to choose one or more actions in your project. You can also select another profiler and profile to define a combination of conditions to evaluate. These actions run when the condition in the preceding profile (or set of profiles) are met.

*Note:* If there are multiple actions in a binding, they must all work with resources of the same type.

*Otherwise do action* allows you to choose one or more actions that run when none of the preceding conditions in the profile (or set of profiles) are met. Within the otherwise clause, you can also select another profiler and profile to define a combination of conditions to evaluate.

*Always do action* allows you to choose one or more actions in your project that will execute whether or not any of the preceding conditions are met.

*Exclude do action* allows you to identify one or more actions in your project that will execute, and whose results returned will be removed from the result set generated by the other actions in the binding.

*Note:* Exclude takes precedence over Always.

*Include Only:*

*Include Only* is a choice within the Select Action and Recommendation Rule structures. You can select action or binding rule to be used as the include only cause. When the content is selected for the main select action or recommended rule, it is only returned if it would be selected by the Include Only clause’s action rule.

*is:*

Select *is* to evaluate the relationship between two sides of a conditional statement.
When using `is`, either side of the conditional statement can typically be the content returned by a resource attribute, value, or arithmetic expression. If the resource attribute is of the data type List (array, vector, or enumeration), the available evaluations become `includes` and `includes any of`. Otherwise, the choices are:

- `includes`
- `includes any of`
- `is between`
- `is between but not equal to`
- `is empty`
- `is`
- `is greater than`
- `is greater than or equal to`
- `is included in`
- `is less than or equal to`
- `is less than`
- `is not empty`
- `is not`

### Is Empty/Is Not Empty

The evaluations `is empty` and `is not empty` allow a rule to check for the existence of a null value or an empty list. When using either of these evaluations, one side of the evaluation is unnecessary and is removed.

**Table 115. Examples of Is Empty or Is not Empty evaluations**

<table>
<thead>
<tr>
<th>Left side of Evaluation</th>
<th>Evaluation</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resource Attribute (non-list type)</td>
<td>is empty</td>
<td>true if attribute is null, otherwise false</td>
</tr>
<tr>
<td>Resource Attribute (non-list type)</td>
<td>is not empty</td>
<td>false if attribute is null, otherwise true</td>
</tr>
<tr>
<td>Resource Attribute (list type)</td>
<td>is empty</td>
<td>true if list is empty, otherwise false</td>
</tr>
<tr>
<td>Resource Attribute (list type)</td>
<td>is not empty</td>
<td>false if list is empty, otherwise true</td>
</tr>
<tr>
<td>Request Attributes or Session Attributes (non-list type)</td>
<td>is empty</td>
<td>false if attribute/parameter exists and value is not null; true if attribute/parameter does not exist or value is null</td>
</tr>
<tr>
<td>Request Attributes or Session Attributes (non-list type)</td>
<td>is not empty</td>
<td>true if attribute/parameter exists and value is not null; false if attribute/parameter does not exist or value is null</td>
</tr>
<tr>
<td>Request Attributes or Session Attributes (list type)</td>
<td>is empty</td>
<td>true if attribute/parameter does not exist or list is empty; false if attribute/parameter exists and list has data</td>
</tr>
<tr>
<td>Request Attributes or Session Attributes (list type)</td>
<td>is not empty</td>
<td>false if attribute/parameter does not exist or list is empty; true if attribute/parameter exists and list has data</td>
</tr>
</tbody>
</table>
Profiler evaluations

If you choose to evaluate a profiler instead of a resource attribute in the Specify a Resource Attribute window, the available evaluations are:

- is
- is all of
- is any of
- is not
- is not any of

On one side of the evaluation, the possible choices are the profiles that are defined within that profiler. You may select one or more profiles for the result of the evaluation.

Order as is:

Order as is is used to specify the order you want selected content to be returned and used. The default, order as is, will return data in the order it is stored in the repository. By clicking the order as is link, you can also choose order randomly or order by.

Order by allows you to sort content by any of its attributes, sort by more than one attribute (and specify the order the attributes are used to sort), and specify whether you want each attribute in ascending or descending order. Order randomly returns data in a different order each time the rule executes.

Profile:

Profile is an arbitrary name (of your choice) that provides information about the Web site visitor, the date and time the visit occurs, or other circumstances or conditions.

To better understand profiles, consider an example. Suppose you want to differentiate your Web visitors according to whether they are able to view confidential information. You might use two profiles in this scenario: Confidential and Regular.

When you create a profile within a profiler, type an arbitrary, but descriptive name. Be as accurate as possible to avoid duplication or confusion with other profiles. When selecting a profile (for example, within a binding), you choose from a list of available profile names that match the profiler rule.

Profiler:

Profiler is a choice within the binding rule structure, and within the Specify a Resource Attribute window when constructing a profiler. Within a binding, you identify specific actions to perform that are based on the one profile within a profiler that evaluates as true.

You may also select quick profiler and specify a simple evaluation in-line instead of using a separate profiler.

Selecting Profiler within a profile allows you to identify other profilers (in effect, nesting profilers within one another) in order to create a profiler that can evaluate as true when multiple profilers or other evaluations are true.
Quick Profiler:

Quick profilers are created within bindings to perform simple evaluations. By using a quick profiler, you can avoid creating simple profilers as separate rules.

When you select the profile link within a binding and choose the quick profiler option, the structure for the line will change to an evaluation. The subsequent line of the binding will be the action to perform if the quick profiler evaluates as true.

sender:

sender must be a valid email address, list of email addresses (comma separated), or a resource attribute containing valid email addresses to whom the email will be sent.

set to:

Learn about the set to action and alternatives to set to.

Set to is the default action within an update action rule. Set to will modify the attribute of a resource, request object, or session object according to the value you specify in the expression.

Alternatives to set to include:
• append
• decrement by
• divide by
• increment by
• multiply by
• prepend
• remove
• remove all

value:

Value is the placeholder for the result of an evaluation. This value can be one you enter, the value of another resource attribute, or an arithmetic expression.

The value must be compatible with the data type of the other side of the expression or evaluation. For example, if you are evaluating an attribute that has the type Number, you can only compare it to resource attributes of the type Number or Decimal Number. The rule editor prevents you from choosing other resource attributes with incompatible types.

Note: Making comparisons against resources in a database respects the column type and size. Therefore, to compare a value to a column typed as CHAR(10), you must include all 10 characters. For example, assume you have a table with a column named DAY that is typed as CHAR(10). A row in the table has the value of 'Monday' rather than 'Monday' in the DAY column because DAY is compared against a profiler condition, and must have all 10 characters defined. However, if the column is typed as VARCHAR, the value in the profiler condition can be 'Monday' (without the four additional blanks).
Mapped values

Resources may be created using mapped values instead of actual values specified in the data store. This facilitates the creation of rules that are easier to understand. For example, if a column in the database held the integer values of 1, 2, or 3 indicating Yes, No, or Maybe, the resource can be configured to map integer values to words. If mapped values have been created for a resource, the mapped values will be used in the rule editor instead of the actual values. For more information on creating value mappings, refer to the documentation in Rational Application Developer for creating resources using the Personalization resource wizard.

Dynamic properties

In addition to predefined resource properties, you can enter properties of a resource that are not in the list. If you know the resource to handle dynamically, specify the name of the property. If the resource manages properties dynamically, the values are retrieved when the rule is evaluated.

Email:

View the prerequisites for creating an email action or promotion within the Personalization workspace.

An email promotion is an email message automatically sent to a defined list of recipients by an executing rule. Email promotions can be sent once or repeatedly on regular intervals. The body of the email message is derived from a file on the server, such as a text file, an HTML file, or even a JSP containing content spots for rules. This file can be chosen from the authoring repository. The list of recipients can be derived dynamically from a rule. Email promotions are implemented by using a rule event to trigger an email rule on a schedule.

Before you can create an email action or promotion within the Personalization workspace, you need the following:

• A user resource. This resource contains information about potential email recipients, and can be created with the Personalization resource wizard in Rational Application Developer. The resource must include a string representing the email address of the recipients (in the form “username@domain”) and must be published to the workspace server and to the run time environment.

• An email body. The body of the email is a flat text, HTML or JSP file that must exist on a server accessible from the run time environment. Typically, this file is created in Rational Application Developer. The location of the file in the run time environment must be specified as a URI when creating the email rule. The email body HTML or JSP page must be on the server which is sending the email.

• An email rule. The email rule specifies to whom the email should be sent, who is sending it, and identifies the email body as a URI.

• A rule to determine recipients (for emails triggered on a schedule or periodically, optional for email actions). This rule can be a select content action or a binding, but must return a collection of recipients from the user resource.
that was previously mentioned. The action or rule you create becomes an option of the To list. If your actions or rules are not properly defined, the To list displays “No Matching Rules”.

It is possible to create email actions or promotions that are sent to a predefined address or list of addresses, simply by typing them into the To field when creating the email action or promotion.

- A rule event (for emails triggered on a schedule or periodically). The rule event binds the email rule to the rule which determines the recipients. The rule event says that at a given time or times in the future, a specific rule should be executed once for each user in a list. That list is determined by the rule to determine recipients.

For more information about configuring email activities, refer to Configuring email activity accounts.

**Email administration:**

View the steps you need to complete before your run time server can send personalized email.

Before your run time server can send personalized email:

- Verify that the pznscheduler.ear is as an Enterprise Application.
- Have a properly configured and operating SMTP email server.
  JavaMail provides the SMTP required to send email. You can manage email responses from customers and outgoing error conditions (such as an unknown email address) using a standard email client.
- Configure a Mail Provider using the WebSphere Integrated Solutions Console.
  1. Click Resources > Mail > Mail Providers.
  2. Create a mail session to use with Personalization. By default, Personalization looks for a mail session in mail/personalizationMailSession/jndi. The JNDI name used is configurable in the PersonalizationService.properties file if you want to use Personalization with an existing mail session you have configured. If you are creating a new mail session, you must specify a Mail Transport Host. This is the mail server Personalization uses to send email. If your mail server is secured, you must specify a Mail Transport User ID and Mail Transport Password.
  3. Restart the server on which the email rules execute for the changes to take effect.
- Additional configuration is available through the PersonalizationService.properties file. Using this file, you configure how often Personalization checks for rule events that run scheduled or repeating emails. You can also configure the mail session being used.

**Content spots**

A content spot is a placeholder or slot for a rule on a Web page. When the page is viewed, the content spot uses its rule mapping to determine which rule to execute. When the rule is executed, any actions defined within the rule take place. Each content spot has a unique name. A content spot’s content type must be defined when it is created and should not be changed.

Content spots are created by developers using the Content Spot wizard in Rational Application Developer and also in the Personalization workspace by selecting New > Content Spot. After creating the spot, the developer can place it on a JSP, or invoke it programmatically from any Java class.
To make the spot available to the Personalization engine, it should be on the classpath of any application which invokes it. If you are using a portlet or Web project in Rational Application Developer, the classpath information updates automatically when you deploy the application.

To make the content spot available to the Personalization authoring portlets, the content spot must be created in the workspace by selecting **New > Content Spot**. The name given to the content spot in the authoring portlet should match the display name given to it in the Rational Application Developer wizard or the name by which it is invoked using the `com.ibm.websphere.personalization.ContentSpot` programming interface. Content spots may be placed into folders by either using a display name which fully qualifies this folder, or by setting the execution scope to match the folder at runtime. So, if you want your content spot to be called `MyDataSpot` in a folder called `ProductData`, then the content spot's display name should be specified in the wizard as `ProductData/MyDataSpot`.

Users of the Personalization workspace specify which rule to place in a content spot. This is also known as mapping the rule to the content spot, or creating a rule mapping. When finished, a workspace user with authority to publish rules and rule mappings publishes them from the workspace server to the run-time environment. Publishing is optional, and is used to move objects between servers. Content spots, rules and all other objects created in Personalization are live as soon as they are created. Rule mappings can be changed at any time and are effective immediately upon publication, or upon the rule mapping start date, whichever comes later. Rule mappings expire on the rule mapping end date.

Content spots can be accessed in the workspace through the Personalization authoring portlets. You can see a list of all the content spots in the project, along with their content type and the name of their mapped rules, by navigating the browser view.

Use of content spots is optional. The `com.ibm.websphere.personalization.ContentSpot` class can be used to directly execute a rule by the rule name.

For additional information, see “Programmatically invoking rules” on page 526.

**Rule spot mappings**

For a rule to be used on your Web site, it must be mapped to an existing content spot. A rule spot mapping is merely an association between a content spot and a rule. Changing the rule that is executed in the run-time environment is as easy as mapping a different rule to a content spot.

You create rule spot mappings within the Personalization workspace. There are two views for rule mappings: **Rule Mappings by Campaign** and **Rule Mappings by Content Spot**. The **Rule Mappings by Campaign** view shows all the rule mappings under a selected campaign from a drop-down menu, and their mapped content spots and rules. This view includes the Default Mappings option which simply shows all the default mappings of each content spot. The **Rule Mappings by Content Spot** view shows all the rule mappings under a particular content spot, including the default mapping and any mappings under campaigns. You can change the personalization behavior of your Web site by mapping a different rule to a given content spot.

Although only one rule is executed when a content spot bean is invoked, you can have multiple rules simultaneously mapped to a content spot by using campaigns.
When you create a campaign, you can create a separate set of rule spot mappings for any or all of the content spots in your project.

When multiple campaigns are simultaneously active, campaign priorities and splits are used to determine the rule to execute. When multiple active campaigns have the same priority, the splits are used to calculate a percentage chance that one mapping will be used instead of the others.

Splits can be changed for each rule spot mapping.

Rule spot mappings can be duplicated and moved from one campaign to another. The start and end date of the rule spot mapping may both be modified if they fall outside the range of dates for the campaign to which the spot mapping is moved. Multiple mappings can be added to the same spot within a campaign.

**Campaigns**

Campaigns are a means of organizing and implementing sets of personalization behavior. A useful analogy is an advertising campaign, which targets specific audiences with high-priority information for a specified period of time. Campaigns achieve this by allowing you to preferentially display campaign-related content in the content spots of a Web site. To accomplish such a goal, a campaign contains a set of rule-to-content spot mappings, start dates, and stop dates.

Users can create and manage campaigns through the Personalization Authoring Portlet. Campaigns are live as soon as their start date is reached and they may be published to other servers together with rules. To create a campaign, select **New > Campaign**. To add rule mappings to a campaign, select the campaign, and select **New > Rule Mapping**.

When a campaign is active in the run-time environment, the rule mappings take precedence over the default rule mappings for content spots the campaign references. For example, a seasonal campaign might contain certain rule mappings that result in the display of special offers to a Web site visitor. A campaign can contain rule mappings for some or all of the content spots on a site.

It is possible to have multiple campaigns active simultaneously. When this happens, the priority settings of the active campaigns dictate which campaign's rule mapping will be used. The campaign with the highest priority 'wins' and its rule mappings are used. In the event that multiple active campaigns have the same priority setting, the rule mapping used for a given content spot is determined randomly according to the relative split ratios.

**Application object**

An application object is a java object existing at a known location in the request context. Defining an application object involves specifying the object’s class name (as a Java class), and specifying a key (string key into a session attribute) to find it in the request context. Personalization calls methods on these objects during rule execution and uses their results in its decision making. Application Objects that implement the `SelfInitializingApplicationObject` interface are automatically instantiated as needed by Personalization.

**Request Context**

This is the interface used to access various attributes for rules. For HTTP contexts, it provides access to the `HttpRequest` and `HttpSession` attributes. For non-HTTP contexts, it provides the same interface to a surrogate for the request and session.
The request context, and any request values accessed via the request context, are only valid for the life of the request.

The Request Context string is used in caching lookups. The lookup is created by using a user-specified string combined with query values as the lookup key. The user-specified string should uniquely represent the current Request Context. This key is stored under ibm.wcp.cache.user.key as a request attribute or as a session attribute with the request attribute taking precedence.

**Accessing the Request Context**

The Request Context provides the Personalization rules engine with the data and environmental information needed for rules processing. In other words, the Request Context contains all the input to execute Personalization content spots and rules. This includes simple inputs like request and session data, and more advanced input like the user object.

You can access the Request Context from a content spot by first using the setRequest method of the content spot to back the content spot with a request, and then by calling getContext to retrieve the context. You can also use the Request Context to call directly into the ResourceDomain3 and ResourceManager3 APIs.

The Request Context allows you to retrieve session, request, portlet attribute, date, cookie, and other data and environmental information from the resource layer.

The Request Context includes:

- **Session**
  The session information identifies the HttpSession object that is associated with the current user.

- **Request date**
  This request date is the date the HTTP request was received. This information supports rules that have date-dependent actions.

Since Personalization uses the Request Context to contain all the rule input, the Request Context must be set onto the content spot prior to rule execution. The code with the content spot’s useBean tag must be similar to:

```jsp
<jsp:useBean id="gold_promo_bean" class="yourco.goldpromo.BannerSpot" />
<% gold_promo_bean.setRequest (request); %>
```

In the previous section, the jsp:useBean tag constructs the yourco.goldpromo.BannerSpot class and stores an instance of that class into the local variable gold_promo_bean. The next line calls setRequest to put the HttpServletRequest or PortletRequest onto the newly constructed content spot bean. The content spot then implicitly constructs a Request Context which is backed by the given HttpServletRequest or PortletRequest. This Request Context then provides access to that request’s parameters and attributes and attributes of the session through the com.ibm.websphere.personalization.RequestContext interface.

In some cases, it may be useful to call into a content spot without having access to an HttpServletRequest or PortletRequest. The interface com.ibm.websphere.personalization.PznRequestObjectInterface can be used in these situations. A implementation of this class called
com.ibm.websphere.personalization.PznRequestObjectImplementor is provided for convenience.

**Query framework**
The query framework is an object representation of a query. The framework is not specifically oriented toward querying either object or relational databases. A set of common operators is defined, but what an attribute represents is determined by the interpreter of the query.

Since the framework makes no assumptions about how the name of an attribute is translated to the object it represents, object trees and graphs, relational models, or almost any other data structure could be queried with the framework. The query framework is used to pass query information from the runtime engine into the resource engine. It is up to the resource collection to interpret the query. Traditionally, this interpretation of the query object is done through a callback class, which is essentially a translator from the query framework into a protocol-specific query language.

**The Personalization interface**
The Portal Personalization user interface consists of three portlets: the Personalization Navigator, Personalization Editor, and the Personalized List.

These portlets are automatically installed with IBM WebSphere Portal. Although personalization services can be used in portlets supporting remote WSRP services, the personalization portlets do not support being used as a remote WSRP service.

**Personalization Navigator**
The Personalization Navigator is the main navigation interface. Users explore the repository with a tree structure—directories display on one side of the window, and elements within a selected directory display in the other part of the window. The view can be modified to eliminate the side of the window with the tree structure, and instead list all directories and elements in a hierarchical list. A drop-down list enables users to show all elements in Personalization, or filter the view to display a single element type (such as rules or campaigns). The following table shows which views are available:

<table>
<thead>
<tr>
<th>List view</th>
<th>Description</th>
<th>Properties shown</th>
</tr>
</thead>
<tbody>
<tr>
<td>Browse Personalization</td>
<td>This tree view will show only artifacts created by Personalization.</td>
<td>Name, Creator, Last Modified, Node Type</td>
</tr>
<tr>
<td>Resources (tree)</td>
<td>For example, rules, campaigns, resource collections.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Name, Creator, Last Modified, Node Type</td>
<td>Name, Creator, Last Modified, Node Type</td>
</tr>
<tr>
<td></td>
<td>This view will show only Personalization rules.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>This view will also show the type of each rule; for example, Select Action</td>
<td></td>
</tr>
<tr>
<td></td>
<td>or Profiler.</td>
<td></td>
</tr>
</tbody>
</table>

Table 116. Available views in the Personalization Navigator

Chapter 5. Setting up a site 371
### Table 116. Available views in the Personalization Navigator (continued)

<table>
<thead>
<tr>
<th>List view</th>
<th>Description</th>
<th>Properties shown</th>
</tr>
</thead>
<tbody>
<tr>
<td>Campaigns (list)</td>
<td>Campaigns will be a specialized list view. In the action bar for the view, there will be a scope drop down. The scope drop down will allow the user to pick an execution scope. The campaigns from the scope context of that execution scope will be shown. When the scope setting is global as by default, there will be no drop down and all campaigns will be shown. As with all views, the Edit Mappings action will be available when a campaign is selected.</td>
<td>Name, Priority, Split, Start Date, End Date</td>
</tr>
<tr>
<td>Rule Mappings by Campaign (list)</td>
<td>This view is launched either by the drop down for selecting a view, or by the Edit Mappings action for a campaign. The view shows a list of mappings for a given campaign.</td>
<td>Spot Name, Rule Name, Content Type, Split, Start Date, End Date</td>
</tr>
<tr>
<td>Collections (list)</td>
<td>This view will allow a user to see all resource collections and application objects created under a collection.</td>
<td>Collection Name, IBM Java Content Repository(True / False), Collection Type (User / Content)</td>
</tr>
<tr>
<td>Events</td>
<td>This view will allow a user to see all rule events on the system.</td>
<td></td>
</tr>
</tbody>
</table>

In addition to navigation, the Personalization Navigator is where users control the repository content. Users can move, copy, import and delete elements, and create new elements. When users create a new element (by selecting **New** from the Personalization Navigator menu), they are automatically taken to the Personalization Editor.

You can also access the Resource Permissions portlet to set access control on individual personalization items from the Personalization Navigator by clicking **Edit Access > Extra Actions**.

**Personalization Editor**

This portlet is where users edit element content or information. Selecting a new element from the Personalization Navigator activates the edit mode, where users enter data, depending on the element chosen. To edit an existing element, users highlight the item in the Personalization Navigator. They return to the Personalization Editor and click **Edit** on the Edit item tab.
Personalized List

The Personalized List portlet allows a user to display personalized content without having to build a custom JSP portlet. Each portlet can display a list of resources and show details for each returned resource. Groups of related resources may be categorized for easy viewing. When a more detailed view of a piece of content is required, a custom detail JSP may be specified. Different instances of the portlet may be used across WebSphere Portal to quickly and easily deploy customized information to users.

Publishing personalization rules overview

WebSphere Portal Personalization sends published rules across HTTP to a servlet which resides on each personalization server. This servlet can receive publishing data or initiate new publishing jobs. When a user begins a publishing job from the personalization authoring environment, the local servlet is provided with the set of information necessary to complete the job. The local servlet contacts the destination endpoint servlet (which could be the same servlet) and sends its data to it. The destination servlet reports success or failure.

Publishing considerations

If you are publishing personalization rules in a clustered server configuration, to an IPv6 host, or using Web Content resource collection classes, there are unique considerations that you should be familiar with.

Clusters

Publishing to or from a clustered environment requires no special configuration. The specific cluster member that will perform the publishing task is chosen by the same rules that apply to incoming Web requests (because the publishing mechanism uses HTTP messages). At the end of a successful publishing job, Personalization flushes its caches for that workspace to ensure that any subsequent personalized content will be as current as possible.

When you first used the Personalization authoring portlets on a cluster to publish objects, the Publish Status dialog (accessed through More Actions > Publish > View Status) only shows information about the publish jobs initiated on that cluster member. To make all publishing jobs visible, set the pzn.publishServlet.url parameter described previously to be a specific cluster member. Set the URL to point to a single machine at the internal HTTP port: The default port number for HTTP is 10040, and the default port number for HTTPS is 10035.

For example, supposed the cluster head is visible at http://intranet.yourco.com, and the cluster members are accessible at http://intranet01.yourco.com and http://intranet02.yourco.com. If you set the publish servlet URL parameter to http://intranet01.yourco.com:10040/wps/pznpublish/pznpublishservlet you force all publishing requests to run on this single machine.

Note: Publish to a single node in the cluster as opposed to the cluster head, making sure you do not pass through a Web server.

IPv6 hosts

The server which initiates the publish command must have the IPv6 protocol stack installed and available. When publishing from the command line using pznload to an IPv6 host, you may need to set the system environment variable
IBM_JAVA_OPTIONS to a value of 
-Djava.net.preferIPv4Stack=false  
-Djava.net.preferIPv6Addresses=true on the system where pznload is run.

**Resource collection classes**

You use resource collections in Personalization to access LDAP, IBM Content Manager, the Portal user object, or other custom sources of data.

The DB2 Content Manager run-time edition and the WebSphere Portal user resource collection classes are installed in the Personalization shared library. Therefore, you do not need to move these classes between systems because they are already installed with Personalization.

For LDAP resources, Rational Application Developer provides a wizard to generate classes which implement the resource collection interfaces.

To use the authoring portlet, all resource collection classes must be in the class path of the Personalization authoring portlet. The rule editor uses these classes to display the list of attributes belonging to the collection. If the resource collection classes are not found by the rule editor, you could see the following message in a JavaScript alert.

![Image](image.png)

*Figure 3. Message displayed when resource classes cannot be found*

The resource collection classes must also exist on the class path of the application invoking the Personalization rules. The Personalization rules engine finds the resource collection classes using the class path of the application which invokes the rules. If you use the Personalized List portlet to display rule results, this application is the Personalized List application pznruleportlet.war in the Personalization Lists.ear.

So, the classes should be accessible to both the rule editor and the personalized application. An application server shared library is the easiest way to accomplish this. You can configure the shared library using the WebSphere Integrated Solutions Console. For more information, see the sections on the shared library in the WebSphere Application Server Information Center.

You handle updates and additions to the resource collection classes just as you would handle updates to any application binary or JSP. These classes are not affected by Personalization publishing. The definition of the resource collection which Personalization uses to associate a resource collection with its classes is stored in the Content Manager repository. Initially represented by the .hrf file, this definition is published along with the rules and campaigns.

**Publishing personalization rules**

After developing personalization rules, you publish the rules.

1. To begin publishing personalization rules, you create an object in the authoring environment which describes the target endpoint. This endpoint definition is referred to as a publish server and is created and managed in a manner similar
to creating and managing rules and campaigns.

The server requires one field, which is the URL associated with the publish servlet for that endpoint. The publish server may also define which workspace will receive publishing data. Personalization operates in the default Content Manager run-time edition workspace after installation. If the target workspace field is empty, then the publish server uses the default workspace. (You need to set the workspace field if you are configuring scenario three described previously.)

The last option is whether or not to delete remote objects that have been deleted on the local system. The default is Smart Delete, which simply removes items that are no longer present. If you do not have delete permission on the remote server you could select the Leave deleted resources on server option.

2. After you create a publish server, you can publish either the entire workspace or a set of objects within it. You specify either of these options by selecting the **More Actions > Publish submenu**.
The Publish page displays what will be published. This page requires the user to choose a destination publish server and any necessary authentication information. If the remote system is secured and is not a member of the current server’s Single Sign-On domain you can enter a user name and password in the provided fields. The values for user and password are stored in the WebSphere Portal credential vault and are not accessible to any other user.

3. Finally, click **Publish** to launch the publish job.
If the local system is able to locate and authenticate with the remote publish server, you are returned to the main navigator view, and you see the Personalization message EJPVP20001I. Then, the publish job runs as a background process on the local server.

4. Click the **View the details of this job** link to open the publish status window to see information about the progress and success or failure of the publish job.
Publishing and deleting personalization rules using a script

You can use a WebSphere Portal Personalization provided script, pznload, to export, publish, and delete Personalization rules on local or remote servers. You can script the delivery of rules and campaigns from staging to production, or the offline publishing between disconnected systems (such as when production servers are secured behind a firewall). You can use this function to quickly revert production servers to an earlier state.

Publishing via the command-line is a two step process. First, you export the personalization rules you want to transfer from the authoring environment to a remote system. After exporting and saving the required objects, you use the pznload script to send this data to the appropriate server.

1. You can export the personalization rules on the site or you can run the pznload command. Select one of the following methods to export personalization objects from the site:
   - Click More Actions > Export in the Personalization Navigator. You are prompted for a location to save a nodes file. This file contains an XML representation of all the currently selected personalization objects. You can export entire folders.
Open a command prompt and run the following command, where `out` designates the location of the exported data on your local system and `targetPath` is the object (and children) that will be exported:

```
pznload --export --out filename --serverUrl url --targetPath path --targetWorkspace workspace --username username --password password
```

2. Choose the appropriate option to send this data to the appropriate server:

   The `pznload` script is located in the `PortalServer_root/pzn/prereq.pzn/publish/` directory.

   **Tip:** This program accepts a number of command line options and a set of nodes files to publish. Invoke `pznload` with the `--help` option to see a list of all options. The most important arguments are described here:

   - **serverUrl**
     The URL of the remote publish servlet. If you do not specify a value the program will attempt to connect to a WebSphere Portal server running on the local machine.

   - **targetWorkspace**
     The name of the workspace to publish to. The default workspace name on all IBM Content Manager run time edition installations is `ROOTWORKSPACE`.

   - **targetPath**
     The location in the target workspace which will be the parent for the published nodes. The target path must exist prior to publishing. Example: If the Export function was used on the folder `/Projects/HR` website, then the target path should be specified as `/Projects` so that the published resources are once again located in `/Projects/HR` website.

   - **username**
     A valid user on the target system with sufficient access rights.

   - **password**
     The password for the user

---

*Figure 8. Exporting a folder to the file system*
• **Windows:** `pznload.bat --serverUrl url --targetPath path --targetWorkspace workspace --username username --password password`

• **AIX/Solaris/Linux:** `./pznload.sh --serverUrl url --targetPath path --targetWorkspace workspace --username username --password password`

• **IBM i:** `pznload.sh --serverUrl url --targetPath path --targetWorkspace workspace --username username --password password`

• **z/OS:** Complete the following steps to send this data to the appropriate server:

  **Note:** Before performing this task, you may want to view and print the appropriate IBM WebSphere Portal Enable for z/OS worksheets located on the wiki to help you define your variables.
  
a. Start the WebSphere Portal Customization Dialog.
b. In the Portal configuration panel, select **Application configuration tasks**.
c. Select **Configure Personalization**.
d. Select **Define variables**. Reminder: Press F1 to display the help panel if you need assistance defining the variables.
e. Generate the customization jobs.
f. Follow the Customization Dialog instructions for submitting the customization jobs.

3. To delete objects, run the following command where **TargetPath** is the object (and all associated children) that will be deleted.

   `pznload --delete --targetPath path --serverUrl url --targetPath path --targetWorkspace workspace --username username --password password`

After a publish is started, you see status messages in the command console.

**Related tasks:**

- [WebSphere Portal Enable for zOS worksheets](#)

**Publishing personalization rules over SSL**

WebSphere Portal Personalization uses the built-in SSL capabilities of WebSphere Application Server to provide secure publishing across unprotected networks. Your personalized portal can benefit from the full range of authentication repositories supported by WebSphere Application Server security.

In some environments even SSL publishing may not be secure enough. The pznload command-line program lets you fully control the transportation of the rules and campaigns during publish. You can encrypt the exported .nodes file and send it using email, or you can use another secure channel such as physical media transported between the staging and production servers.

1. Enable SSL between the personalization servers. To enable Personalization publishing over SSL, see the Personalization Navigator’s inline help: click the question mark, and scroll down to the end of the page to locate the link to the help topic on publishing.

2. Alter the publish servlet URL for secure publishing. If the remote server is not using the default HTTPS port of 443, modify the URL by adding a colon and the port number immediately after the host name.
3. Configure the personalization server from which you will be publishing to use the HTTPS protocol. To determine whether a particular URL is valid, point your browser to that location and enter your username and password for the system. If you see the message Publish servlet available and all SSL certificates have been properly imported, you should be able to publish. You can change this URL to redirect all publish jobs through a specific cluster member. If you encounter an error message that indicates the publish service was not available, the local publish servlet may not be configured correctly. To configure the local publish servlet URL:

a. From the Portal Administration page, select Portlet Management > Portlets.
b. Locate the Personalization Navigator portlet in the list.
c. Click Configure portlet to configure the portlet.
d. Add a new portlet parameter whose name is pzn.publishServlet.url and specify the appropriate value.
If a Personalization server is configured to use a nonstandard HTTPS port or context root, or if you see messages such as EJPVP20002E: The local publish service was not available when publishing from the authoring environment, the local publish servlet URL might be incorrect.

**Monitoring the status of publishing**

After you start a job to publish a personalization rule, you can monitor the status to make sure the publish completes successfully.

All publish jobs that are currently running or have been completed are displayed in the status view. If an error occurs, to get more information, turn on the Java Run time Environment tracing for WebSphere on the client system or examine the error and trace logs on the server system.

1. To see the status of all current publish jobs, select **More Actions > Publish > View Status**.
2. After a job has completed (successfully or otherwise) a close icon displays that you can click to remove the job from the list of monitored jobs. If you click this icon, you can no longer view the status of that job.

The Web Content resource collection

The Web Content resource collection is installed and configured out of the box. This predefined collection allows you to write rules that select lists of content from IBM Web Content Manager. Rules specify which Web content to show in a Portal Personalization component in Web Content Manager.

The Personalization component is similar to a menu component in that its content is decided by a rule. The Personalization component specifies how the content returned from the rule is presented and points to a rule to decide what content to display. That rule may make use of the Web Content resource collection to select a list of Web content.

Personalization rules that you create using the Personalization editor are not managed in Web Content Manager and so are not available for versioning or included in syndication. These rules must be published using pznload or by publishing with Personalization.

The Web Content resource collection allows rules based on the following attributes:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Author</td>
<td>The author of the content as set in Web Content Manager. This attribute value is stored as a Distinguished name; for example, uid=wpsadmin,o=defaultWimFileBasedRealm.</td>
</tr>
<tr>
<td>Authoring template</td>
<td>The authoring template used to create the Web content. The value may be selected using an authoring template picker.</td>
</tr>
</tbody>
</table>
Table 117. Attributes used in Personalization rules for Web Content resource collections (continued). The names and descriptions of Web Content attributes that you can use in Personalization rules

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Authoring Template Properties</td>
<td>All authoring template properties that are available to be used within rules.</td>
</tr>
<tr>
<td>Category</td>
<td>The categories to which the piece of Web content belongs. If you are matching this attribute to a value from another resource collection or application object, the format for the value should be /parentcategory/childcategory.</td>
</tr>
<tr>
<td>Creation date</td>
<td>The date the piece of Web content was created.</td>
</tr>
<tr>
<td>Creator</td>
<td>The person who first created this piece of Web content. This attribute value is stored as a Distinguished name; for example, uid=wpsadmin,o=defaultWimFileBasedRealm.</td>
</tr>
<tr>
<td>Description</td>
<td>The description provided for the piece of Web content.</td>
</tr>
<tr>
<td>Expiration date</td>
<td>The date the content is set to expire in Web Content Manager.</td>
</tr>
<tr>
<td>Full text</td>
<td>Use this attribute to search the full text of a piece of Web content. It should be used sparingly. This attribute may not perform as quickly as other attributes when used in rules, especially when used in conjunction with other attributes.</td>
</tr>
<tr>
<td>Keywords</td>
<td>The keywords stored on the piece of Web content.</td>
</tr>
<tr>
<td>Last modified date</td>
<td>The date a modification last occurred on the piece of content.</td>
</tr>
<tr>
<td>Last modifier</td>
<td>The last person to modify the piece of content. This attribute value is stored as a Distinguished name; for example, uid=wpsadmin,o=defaultWimFileBasedRealm.</td>
</tr>
<tr>
<td>Name</td>
<td>The name of the piece of Web content as specified in Web Content Manager. This property uses case-insensitive matching. For example, a piece of content with a name of &quot;SampleContent&quot; matches &quot;SampleContent,&quot; &quot;samplecontent,&quot; &quot;SAMPLECONTENT,&quot; and other variations.</td>
</tr>
</tbody>
</table>
Table 117. Attributes used in Personalization rules for Web Content resource collections (continued). The names and descriptions of Web Content attributes that you can use in Personalization rules

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Position</td>
<td>The zero-based numeric position of the piece of content among its siblings in the nodes of the content hierarchy. Web Content resource collections use absolute positioning and relative positioning to support the ordering of content items. The algorithm used to generate position numbers results in fractional and negative values that control the display order of a set of child content items under a given area of a site. For more information, refer to the description of the position attribute in the Command reference for the Portal Scripting Interface.</td>
</tr>
<tr>
<td>Publish date</td>
<td>The publish date as specified in Web Content Manager.</td>
</tr>
<tr>
<td>Location</td>
<td>The document library or site area from which to select content. If specified as a string, it should be in the format /Library/Site/SiteArea. You can also specify <em>/Site/SiteArea to search by site area in all document libraries. Rules from previous versions with the Site Area attribute will automatically reference Location. If the site area was specified as a string, ensure the string starts with /Library or an asterisk (</em>).</td>
</tr>
<tr>
<td>Title</td>
<td>The display title as specified in Web Content Manager.</td>
</tr>
<tr>
<td>Unique Identifier</td>
<td>Use this attribute to select pieces of Web Content.</td>
</tr>
</tbody>
</table>

When a text, short text, numeric, or date component is added to an authoring template in Web Content Manager, that component will appear as an attribute on the resource collection under the Authoring Template Properties menu item. For example, if an authoring template is created for "Benefit Announcement" which includes an "enrollment begin" date component, you will have an "Enrollment begin" attribute on the Web Content resource collection. This new attribute would appear under a submenu for "Benefit Announcement". This will allow you to write rules based on custom metadata you attach to Web content. The performance of the standard metadata will do better compared to the performance of rules using attributes added to authoring templates. The use of keywords and categories should be considered since these are part of the standard metadata of Web content.

If too many authoring template properties have been designed, the Authoring Template Properties menu may become too large to be easily used. Once there are more than 15 authoring template properties, they are replaced with a chooser to select properties. This threshold can be adjusted by changing the value of wcm.authoringTemplate.menu.threshold in wp_profile_root/PortalServer/config/services/PersonalizationService.properties.
Personalization rules querying on short text run faster than those querying on text. Short text components can only store 254 characters. Text components can store an unlimited number of characters, but personalization rules will only see the first 254 characters. Anything after that is ignored when the rule runs.

Note: All attributes which store a reference to a person are stored as distinguished names. This format will match against the value of the Distinguish Name attribute on the Portal User resource collection. For instance, to select all documents authored by the user viewing the portlet, you would write, "Author is current Portal Users.Distinguished Name."

The Portal User resource collection

Portal Personalization comes with a Portal User resource collection. This collection uses public APIs provided by IBM WebSphere Portal to access user information.

The collection allows rules to be written based on all properties of the WebSphere Portal user.

You can also use the Portal User resource collection to profile based on the property extension database. Use the Manage Properties menu option in the rule editor to add or remove dynamic or Lookaside properties from the collection. You may add any properties to this collection, but for them to function at runtime some values must be stored in Virtual Member Manager for the properties being used and the user executing the rule. The Manage Properties screen can also be used if the server on which you author rules has a different Virtual Member Manager configuration than the one on which the rules are deployed.

The resource collection is not new to WebSphere Portal Version 8.0, but it now shows the list of Portal User attributes automatically. You no longer have to add each WebSphere Portal property you want to use as a dynamic property in the Personalization rule editor. You do not need to generate Java code to use this collection or configure a security ID translator as is often required for Personalization User collections. You cannot currently write rules to select or recommend a list of users from this collection. This collection works with update rules as long as your repository allows writes. The collection works as if it is an LDAP collection that is automatically configured for the LDAP server.
You may continue to use your existing LDAP or custom user resource collections, and even use them in the same rules as the Portal User Collection. This is useful if you have multiple user repositories, you a have repository that is generally not used for Portal and only used for Personalization rules, or you requirements for attribute value translation (‘CHI’ should be interpreted as ‘Chicago’ for instance).

**Using the Groups Attribute**

The Groups attribute on the Portal User object exposes the distinguished names (dn) of the groups. An example of a distinguished name of a group is cn=wpsadmins,o=defaultWimFileBasedRealm, though the exact form will vary by your installation. Using distinguished name allows for more exact matching of groups, since it is possible for two groups to share a common name, such as wpsadmins. The `includes` operator may be used for inexact string matching, but will perform slightly slower. When possible, use the `is` operator and match to the distinguished name of the group.

**Adding and extending user attributes**

You can make attributes from your user registry available to the personalization portlet as required. However, in most cases the schema for your user registry does not match the default schema for Virtual Member Manager (VMM). For this reason, you must first extend the default VMM schema by adding attributes that you must then map from the VMM schema to your user registry.

Do the following to make user attributes available:

1. Extend the default VMM schema by adding attributes that you can map to your user registry. For instructions, see the following topic in the Installation section of this Information Center: *Adding attributes*. Refer to the topic that corresponds to your operating system and environment configuration.

2. Map the attributes you added to the VMM schema to the attributes in your user registry. For instructions, see the following topic in the Installation section of this Information Center: *Mapping attributes*. Refer to the topic that corresponds to your operating system and environment configuration.

**Configuring which properties show up for the Portal User Collection**

The Personalization rule editor discovers the list of properties to show in rules through a public API (com.ibm.portal.um.PumaProfile). The list exposed by this API is configurable in your Member Manager configuration.

You may have a long list of properties that are available on the portal user, but do not want all of them to show up in the Personalization rule editor. For this purpose, set the property `wmm.property.hide` in the file `PortalServer/config/config/services/PersonalizationService.properties` as illustrated in this example:

```properties
# Use this configuration property to control which WMM properties show
# in the Personalization rule editor. wmm.property.hide will only
# hide those properties which are introspected from the WMM configuration.
wmm.property.hide=mobile,pager,roomNumber,secretary,carLicense,telephoneNumber,facsimileTelephoneNumber,
seeAlso,userPassword,ibm-firstWorkDayOfWeek,ibm-alternativeCalendar,ibm-preferredCalendar,
ibm-firstDayOfWeek,ibm-primaryEmail,ibm-otherEmail,ibm-generationQualifier,labeledURI,createTimestamp,
modifyTimestamp,ibm-middleName,ibm-timeZone,initials,jpegPhoto,WCM\:USERDATA,groups
```

Chapter 5. Setting up a site  387
The Portal User collection and the Personalization Server installed without Portal

The Portal User collection can only be used in rules running within a WebSphere Portal installation. The system will not prevent you from publishing rules using this collection to a Personalization Server installed outside of WebSphere Portal, but the rules will not function in this environment since Virtual Member Manager is not available. When WebSphere Portal is installed, the security context of the logged in user must have access (through Portal resource permissions) to the user information being accessed through the collection.

LikeMinds Recommendations

Personalization contains a dynamic recommendation system based on LikeMinds. LikeMinds is software that is used with your e-commerce applications. LikeMinds analyzes user interactions that occur on your Web site and generates real time predictions and recommendations to your Web site users.

Real time predictions are generated by three LikeMinds engines using recommendation rules within Personalization. These rules, called recommend content, base their predictions on transactions logged through Personalization’s rating and action beans.

When a user visits your Web site, rating and action beans log captured transactional data. If your e-commerce Web site is set up so that users can rate content (or products), you use Rating beans to capture rating data. Similarly, if you use shopping cart technology, you use action logging beans to capture content affinity behavior to capture shopping activity. Both rating and action data is stored in your database. For example, the following types of transactions may be recorded:

- Products a user has purchased
- Items added or removed from a shopping basket
- A history of the user’s navigation throughout the application
- Products that go best with a product that the user has already selected
- Any action or series of actions that are meaningful for a site

Using recommend content rules, LikeMinds surfaces results through a set of recommendation engines. These engines predict relevant content for users based on their past Web browsing habits.

Typically, after a user has rated a minimum number of items or completed a minimum number of transaction activities, that user is assigned a set of mentors. A mentor is a specially designated user who has visited the e-commerce application a number of times, and whose profile is similar to the user’s. LikeMinds uses a technique called collaborative filtering to build a mentor’s profile for each user to predict how much a user will like particular items and which items that user will enjoy, buy, or add to their shopping cart.

Predicting a matching product to go with a user’s selected product, independent of actual user preferences, is accomplished by the discovery of probable pairs of product matches to be recommended. This concept is called item affinity and uses a family of algorithms different from collaborative filtering. While collaborative filtering uses its algorithms to discern the highly variable affinities between individual Web-surfers, the item affinity approach looks at relationships that can exist between items.
You can use Like Minds in a variety of situations, including:
- eRetailer promotion and personalization Web sites
- Financial portal content recommendation and personalization Web sites
- Help desk and/or on-line technical support content recommendation Web sites
- Gift recommendations for eRetailer
- Music, movie, book, or other product rating and recommendations
- Travel bureau trip planners

**Like Minds Recommendation Engine architecture**

Get an overview of the functioning and architecture of the Like Minds Recommendation Engine. In order to build your own recommendation application you need to customize the Like Minds Recommendation Engine settings to work with your database and Web applications.

The Like Minds Recommendation Engine captures data based on user actions. From this set of actions, Like Minds bases its construction of mentor sets and subsequent predictions. User actions can include:
- A history of the user's navigation throughout the application
- Products that go best with a product that the user has already selected
- Products purchased
- Items added to or removed from a shopping basket

The current user preferences are collected by the engine in the so-called rating vector and used to identify those people most like the current user.

People with a behavior similar to the current user become mentors for that user. The Like Minds Recommendation Engine assigns a numeric weight to each mentor based on the level of similarity of the rating vector to yours. The more the mentor's rating values resemble a user and the more products the mentor has rated, the greater the weight.

The Like Minds Recommendation Engine assembles a set of recommendations by finding the products each mentor recommends and creating a prediction vector containing the predicted rating of each product. With each predicted rating, it also stores a numeric value representing the confidence for the rating.

The confidence values determine the quality of predictions. The Like Minds Recommendation Engine assigns a confidence level to each recommendation based on how many users have rated the recommended item and how similar the ratings are to each other.

A user is assigned a set of mentors only after he has rated a minimum number of items or completed a minimum number of transaction activities.

If you want your application to predict a matching product to go with a user's selected product, you can configure probable pairs of product matches to be recommended. This concept is called item affinity.

Depending on the type of recommendation you want to extract, the specific engines must be configured, such as the Preference Engine or Item Affinity Engine. The following image shows the details of the overall Like Minds Recommendation Engine architecture.
How LikeMinds generates recommendations
Learn how LikeMinds generates recommendations when a user logs on and navigates through your web site.

When a user logs on and navigates through your Web site, LikeMinds follows these steps to generate recommendations for that user:

1. Personalization Rating beans and action logging beans create a record for new users in the Lps_User_Data table. The Lps_User_Data table stores the following types of information about the user: the user’s resource ID, a user ID, the number of items the user has rated or selected, and so on.

2. The Personalization Rating beans and action logging beans log data for that user as that user navigates your Web site. The profile data is first stored in the server’s cache, then the server writes all the new data to the database. The Lps_User_Rating table stores the user’s explicit preferences; the Lps_User_Trx table stores the user’s clickstream and purchase behavior. The Lps_User_Trx table also stores item affinity input data.

3. The application can then query LikeMinds for recommendations. Recommendation queries are transaction data-specific.
• Preference recommendations are surfaced by the recommend content rule. To receive Preference recommendations, your application must record users' explicit preferences (ratings) using the Rating bean.

• Clickstream recommendations are surfaced by the recommend content rule. To receive Clickstream recommendations, your application must record users' clickstream behaviors (that is, product detail views, shopping basket inserts, and so on) using the Action bean.

• Item Affinity recommendations are in the form: “this mouth guard is a likely product to go along with the hockey puck the user has just added to his or her shopping cart”. To receive Item Affinity recommendations, your application must record users' likely product pair matches. In other words, your application must capture the current "content/product" context using the Action bean, in order to return those items most associated with that "content/product".

4. Depending on the engine you are using, the following step occurs next:

• **Preference and Clickstream engines**: For a new user, if your application queries LikeMinds for recommendations before mentors have been assigned for that type of data (that is, Preference, Clickstream, or Item Affinity), the server will assign mentors from a cached pool of mentors. If the server is unable to, for lack of profile data, match cached mentors to this user, the server will provide an empty set of recommendations. An important distinction, profile data means the transaction data for the current user and not the attributes of that user.

• **Item Affinity engine**: If your application is predicting item affinity product pair matches, it will collect data based on a set of definitions that you create, called an item affinity set. For input data, the item affinity set uses transactions from a specified input table.

5. Depending on the engine, the following step occurs next:

• **Preference and Clickstream engines**: Once a user's profile is stored in the database, the sifter utility can calculate mentors for that user. The sifter is a background utility which assigns a set of mentors to each user.
  – Mentor assignments are specific to each type of data.
  – Mentor assignments are stored in the mentor table associated with this type of data.

• **Item Affinity engine**: The accumulator generates item affinity product pair matches by analyzing data in the item affinity set (same as transaction table) and recording its findings to an output table.

6. Depending on the engine, the next step is as follows:

• **Preference and Clickstream engines**: As new transaction data is recorded for a user, the user is prioritized for reprocessing by the sifter to calculate new mentor assignments. Users are prioritized by a calculated 'sift priority', reflecting the percentage of new or changed profile data for that visitor.

• **Item Affinity engine**: As new product selection behaviors are recorded in the transaction input table specified in the item affinity set definition, the accumulator uses this data to calculate new item affinity recommendations.

7. When your application runs LikeMinds rules, the following occurs, depending on the engine:

• **Preference and Clickstream engines**: LikeMinds looks up that user's mentors, and calculates recommendations.

• **Item Affinity engine**: LikeMinds calculates the most likely item-content pairs based on accumulated item transaction history.
The LikeMinds Recommendation Engines

LikeMinds Recommendation Engines communicate with a relational database and generate recommendations. Learn about the three types of recommendation engines, Preference engine, Clickstream engine, and Item Affinity engine.

Following is a description of the three types of recommendation engines:

- **Preference engine**: This engine generates recommendations using patented collaborative filtering algorithms based on users' item ratings.

- **Clickstream engine**: This engine, which also accesses transaction information, generates recommendations based on users' actions as they navigate a Web site; that is, the history of user “clicks” during website visits, and the items that users view, click, and add to their shopping carts.

- **Item Affinity engine**: This engine generates recommendations based on the history of the user's site browsing activity. It matches a currently selected product with a second product that the user would most likely want to purchase along with the first product. For example, if a user is purchasing groceries and adds French onion soup to the shopping cart, the Item Affinity engine could recommend Gruyere cheese to go with it.

**Preference Engine**:

The Preference Engine uses explicitly stated user preferences to make highly accurate recommendations for products and content that your website visitors like.

The Preference Engine enables customers to get exposure to items they might otherwise miss. For example, users shopping for gifts can get recommendations based on the gift recipient's shopping preferences.

The shipped with the LikeMinds Recommendation Engine presents a typical implementation of the Preference Engine:

- A user can rate a movie and LikeMinds returns a predicted rating for that movie.
  
  For example, a user can assign the movie *Fantasia* with a rating such as "I loved it". Internally the rating corresponds to a numerical value.

- A user can ask for a recommendation about the best bet, which is provided by a LikeMinds rule. This corresponds to asking which movie a user would like the most among all available movies.

**Clickstream Engine**:

Based on navigational data gathered as customers browse your Web site, the Clickstream Engine tracks clickstream (or rating) behavior and generates recommendations based on mentors who exhibit similar content/product affinities.

The Clickstream Engine tracks the pages that users have looked at. After analyzing all users' traffic patterns, it then makes content recommendations for each specific user, using data from relevant subsets of the user base.

**Item Affinity Engine**:

The Item Affinity Engine generates recommendations based on any transactional history available, such as shopping cart activity, external legacy transactions, and web transaction completely unrelated to shopping cart activity (page views, product inquiries, searches, and so on).
An Item Affinity Engine can, for example, predict that a user who purchases a
digital camera would likely want to purchase compact flash cards or a USB card
reader. The Item Affinity Engine also spots the less obvious connections—that
users who purchase beer are likely to purchase diapers at the same time, for
example.

The Item Affinity engine lets you track more than mere purchases measured at
check-out time. It can identify items the user only considered for purchase. For
example, it can know when a user only considered purchasing rye bread rather
than actually purchasing the rye bread. (This is measured by a shopping cart add,
followed by a shopping cart drop, of the rye bread.) In this case, the grocer could
not possibly know that the user considered purchasing rye bread during the
shopping session, since the rye bread was not in the shopping cart at check-out
time. Even though a considered purchase does not necessarily imply the same level
of item affinity as a completed purchase, it does convey item affinity information.

Unlike the other engines, Item Affinity Engine recommendations are based on
Market Basket Analysis statistics, not collaborative filtering. Market Basket
Analysis enables content affinity predictions even when cold-start situations
obscure the relevance of collaborative filtering. The Item Affinity Engine can be
used to provide improved automated recommendations, such as cross-sells, even
for first time visitors to the Web site.

**The LikeMinds utilities**

Get an overview of sifter, buildstats, buildvisit, and accumulator, the utilities
that support running of background processes along with the LikeMinds server.

The following utilities support the background processes operating on the database
when the LikeMinds server is running:

- **sifter:** This utility runs continuously to identify mentors for new users and
  recomputes the best set of mentors for existing users. The mentor set for a user
  may change as the sifter gathers more information about the user or the
  mentors. The sifter identifies mentors for the Preference, and Clickstream
  engines.

- **buildstats:** This utility runs once a day to update statistics for each item, such
  as the number of ratings or transactions, the average rating, the standard
  deviation in the average rating, and the default recommendation information.
  The Clickstream and Preference engines use buildstat. The Item Affinity engine
does not.

- **buildvisit:** This utility, which the Preference engine uses, runs daily to
  construct lists of items to be presented to users for rating. If your applications
do not use the Preference engine, buildvisit is not necessary.

- **accumulator:** For the Item Affinity engine, the accumulator (listed as lpsIAA in
  the util directory) accumulates the number of times every possible item-to-item
  combination occurs and writes its findings to an output table specified by the
  item affinity set. An item affinity set defines the type of data required to build
  an item-to-item combination.

**Configuring LikeMinds**

Use a suitable database modification tool or edit the likemindsdb.properties file
to configure your LikeMinds server installation.

Set the following general lps_cfg parameter information for the LikeMinds server
installation:

- Basic server information
Scheduling LikeMinds events
Server load management
Cache behavior
Recommendation behavior

All these parameters are set in the likemindsdb.properties file. This file is located in the directory `/wp_profile_root/pzn/config/runtime/likemindsdb.properties`. The file is in ASCII. To edit it, use an ASCII editor.

LikeMinds stores its configuration information in the lps_cfg table of its database, which is initialized with the data in the likemindsdb.properties file. To update this configuration, you can update this file and reload the configuration data, or you can use any database modification tool, to modify the LikeMinds configuration parameters. To update configuration values, complete these steps:
1. Stop the WebSphere_Portal server.
2. Edit the likemindsdb.properties file.
3. Choose the appropriate task to update the configuration:
   - **Windows**: `ConfigEngine.bat likeminds-load-config -DWasPassword=password`
   - **AIX/Solaris/Linux**: `./ConfigEngine.sh likeminds-load-config -DWasPassword=password`
   - **IBM i**: `ConfigEngine.sh likeminds-load-config -DWasPassword=password`
   - Run the `EJPSZL11` customization job, which the Customization Dialog created in your target control data set. This job runs the `likeminds-load-config` configuration task.
   - Use the DB modification tool to update the configuration directly.
4. Start the WebSphere_Portal server.

Estimating database size:

The size of your database depends on your application, as well as the number of users and items. View some general guidelines for estimating the size of your database, but your results may vary.

A database containing the seed data supplied with the Movie Site application may use just 250 MB, while the LikeMinds tables for a large site with millions of users may take up 10 GB. Following are some general guidelines:

The tables that contribute the most to the size of the database are as follows:
- **Lps_User_Rating**: This table normally dominates your space considerations. Users typically average 50 to 100 ratings. The seed users supplied with Movie Site average about 500 ratings.
- **Lps_User_Trx**: This table can grow very large, depending on the number of item affinity, clickstream, or purchase activities recorded from your applications.
- **Lps_MBA_Scored**: This table can grow quite large, depending on the number of products your site sells and the number of relationships you want to configure for each product. For example, if you have 1000 products listed in your Lps_Item_Data table, and you want to store 10 relationships for each product, an Lps_MBA_Scored table can grow to 10,000 rows.
- **Lps_User_Mentor**: The size of this table depends on the number of users and the number of mentors associated with each user (50 by default).
- **Lps_User_Data**: This table may contribute a large portion of the database size if you have large numbers of users who each have few ratings. This table is heavily indexed, which can affect performance.
- **Lps_Item_Data**: This table is normally fairly small, but may be significant if you store large amounts of data about each item.

The remaining tables are typically less than 100 KB each.

The following table gives typical numbers of rows, row sizes, and index sizes for a "typical" Microsoft SQL Server database with 5000 items and 100,000 users. The row sizes include only the fields required by LikeMinds, and they account for typical null fields and clustered index overhead. Sizes will vary for other database systems, especially for indexes.

<table>
<thead>
<tr>
<th>Table</th>
<th>Rows in Typical Site</th>
<th>Row Size (Bytes)</th>
<th>Total Size</th>
<th>Index Size (Bytes Per Row)</th>
<th>Total Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lps_User_Rating</td>
<td>8,000,000</td>
<td>25</td>
<td>200 MB</td>
<td>about 20</td>
<td>160 MB</td>
</tr>
<tr>
<td>Lps_User_Trx</td>
<td>8,000,000</td>
<td>32</td>
<td>256 MB</td>
<td>about 20</td>
<td>160 MB</td>
</tr>
<tr>
<td>Lps_User_Mentor</td>
<td>5,000,000</td>
<td>25</td>
<td>125 MB</td>
<td>about 20</td>
<td>100 MB</td>
</tr>
<tr>
<td>Lps_User_Data</td>
<td>100,000</td>
<td>typical: 100</td>
<td>100-400MB</td>
<td>about 100</td>
<td>10 MB</td>
</tr>
<tr>
<td></td>
<td></td>
<td>maximum: 400</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lps_Item_Data</td>
<td>5000</td>
<td>136 (for required fields only)</td>
<td>68 MB</td>
<td>4</td>
<td>2 MB</td>
</tr>
<tr>
<td>Lps_MBA_Scored</td>
<td>10,000</td>
<td>32</td>
<td>32 MB</td>
<td>about 20</td>
<td>1 MB</td>
</tr>
<tr>
<td>Lps_Genre_Data</td>
<td>10-1000</td>
<td>116</td>
<td>1160 KB-116 MB</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>Lps_Item_Genre</td>
<td>5000-20,000</td>
<td>12</td>
<td>60-240MB</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>Lps_User_Selector</td>
<td>25,000-100,000</td>
<td>12</td>
<td>30-120MB</td>
<td>4</td>
<td>100-400 MB</td>
</tr>
</tbody>
</table>

In your size estimate, remember to include space for transaction logging and rollback areas. Because the LikeMinds server commits frequently, the rollback area need not be especially large relative to the database. Allow space equal to the size of the database for transaction logs, since the LikeMinds server performs frequent updates.

**Database performance:**

View some guidelines for performance optimization in your LikeMinds database.

How you configure your database during installation has a significant impact on performance. Use the following guidelines to achieve the maximum performance in your LikeMinds database:
- **Hardware setup**: Ideally, you should have one machine (with two CPUs), preferably the fastest, dedicated to the database, and the WebSphere Portal
software on a separate machine. See the system requirements in the WebSphere Portal Information Center for more information.

- **load distribution**: You can install the LikeMinds utilities onto separate machines to distribute load. Refer to the WebSphere Portal Information Center for information on installing utilities to multiple machines.

- **sifter configuration**: You can configure the sifter to accommodate heavy loads or busy sites. Be aware that the sifter has heavy memory usage.

- **accumulator configuration**: As with the sifter, the accumulator has heavy memory usage, so you should schedule it to run during off-peak hours.

**Scheduling LikeMinds Events**

Use the `lps.schedule` setting to schedule events to be fired at specific dates and times.

The syntax for using the `lps.schedule` setting is as follows:

```
lps.schedule.<event name> = <schedule><event type><event args>
```

where:

- **event name**: Refers to a unique name for the event.

- **schedule**: Refers to the scheduling time, in five fields separated by spaces or tabs, and are integer patterns. These fields are:

<table>
<thead>
<tr>
<th>Field</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>minute</td>
<td>0–59</td>
</tr>
<tr>
<td>hour</td>
<td>0–23</td>
</tr>
<tr>
<td>day of the week</td>
<td>0-6, with 0 being Sunday</td>
</tr>
<tr>
<td>day of the month</td>
<td>1–31</td>
</tr>
<tr>
<td>month of the year</td>
<td>1–12</td>
</tr>
</tbody>
</table>

Table 119. Schedule Time Values

To specify all values of a particular field (for example, to schedule events every day of the week), use an asterisk (*), in the order listed in this table. For example:

```
0 2 1-2 * *
```

month of the year, in this case, all
day of the month, in this case, all
day of the week, in this case, Monday and Tuesday
hour, in this case, 2am
minute, in this case, 0

When you do specify actual values, you can enter either the value itself (such as 1 to schedule an event on Monday) or a range of values (for example, 1-2 to schedule events for Monday and Tuesday). When you want to specify days, you need to include both the day or the week field and the day of the month field. For examples of using these values, see
aspect name and aspect args: Refer to any of the following events and their arguments:

Table 120. Event Types and Arguments

<table>
<thead>
<tr>
<th>Event type</th>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lazyDBWrite</td>
<td>N/A</td>
<td>Specifies when all the changed information since the last read is written to the database. The default setting is: 12,24,36,48,00 **** lazyDBWrite</td>
</tr>
<tr>
<td>purgeUserCache</td>
<td>timeval</td>
<td>Specifies the time in seconds after permanent users have not been used before purging them from cache. If timeval is not specified, the setting from db.tune.user_cache_age_time is used.</td>
</tr>
<tr>
<td>purgeItemCache</td>
<td>timeval</td>
<td>Specifies the time in seconds after items have not been used before purging them from cache. If timeval is not specified, the setting from db.tune.item_cache_age_time is used.</td>
</tr>
<tr>
<td>syncCache</td>
<td>user</td>
<td>Refreshes all permanent user objects in cache.</td>
</tr>
<tr>
<td></td>
<td>item</td>
<td>Refreshes all item objects in cache.</td>
</tr>
<tr>
<td></td>
<td>engine</td>
<td>Refreshes all engine cache.</td>
</tr>
<tr>
<td>runBuildstats</td>
<td>verbose</td>
<td>Runs the buildstats utility. If verbose is specified, additional information is printed to the trace log.</td>
</tr>
<tr>
<td>runBuildvisit</td>
<td>verbose</td>
<td>Runs the buildvisit utility. If verbose is specified, additional information is printed to the trace log.</td>
</tr>
<tr>
<td>runAccumulator</td>
<td>verbose</td>
<td>Runs the accumulator utility. If verbose is specified, additional information is printed to the trace log.</td>
</tr>
</tbody>
</table>

Configuring the LikeMinds engines:

You can configure your LikeMinds recommendation engines to control aspects such as predictability, number of mentors used, which rating, transaction, or mentor set to use, and so on, for the recommendations returned.

The Preference and Clickstream engines rely on the sifter utility to assign mentors to users. The Item Affinity Engine uses the accumulator utility to collect item affinity data.

Configuring the Preference Engine:

The Preference Engine generates recommendations based on users' ratings of items. You can configure the following settings for the Preference Engine.

Number of mentors to use:

Set `<eng_instance_name>.db.engine.tune.num_mentors` to the maximum number of mentors to assign to a given user. The default is 50. For example:

```
movie_pref.db.engine.tune.num_mentors = 60
```
If you do not set this parameter, then the Preference Engine uses the value set for 
<mentor_set>.max_mentors = <number>.

Mentors to look for in cache:

If a user requests recommendations before the sifter has found mentors for that 
user, the Preference Engine checks the cache for mentors for the user. Set 
<eng_instance_name>.db.engine.tune.max_cached_mentors to the number of 
potential mentors which it should consider for a given user. The default is 500. For 
example:
movie_pref.db.engine.tune.max_cached_mentors = 600

Use of “average user” to improve recommendation confidence:

The Preference Engine can use an “average user,” whose ratings are the average of 
al users' ratings. The buildstats utility computes the average ratings. Configuring 
an average user improves the confidence level of recommendations for that user. 
To do so, set eng_instance_name.db.engine.tune.disable.avg_user to false.

However, remember that this feature can be costly in system resources. To disable 
it, set it to true. (The default setting is true.)

Archetypes:

To generate recommendations for items that no mentors have rated, the Preference 
Engine uses synthetic users called “archetypes.” These users possess a particular 
characteristic very strongly. For example, an archetype who only likes action 
movies might rate all action movies very highly. Recommendations from 
archetypes are added to recommendations produced by mentors.

Archetype configuration includes:

- “Setting the number of archetypes in cache”
- “Enabling or disabling the use of archetypes” on page 399

Guidelines for configurable recommendation dynamics:

You can use the following configuration parameters to control, or at least balance, 
effects of external factors.

- The number of mentors to use in making recommendations, using the following 
  configuration parameters:
  - <mentor_set>.min_mentor_ratings
  - <mentor_set>.max_mentor_ratings
  - <mentor_set>.min_mentor_transactions
  - <mentor_set>.max_mentor_transactions
- Balancing the mentor selection process with the following configuration 
  parameters:
  - <mentor_set>.mentor.pool.size
  - <mentor_set>.max_mentors

Setting the number of archetypes in cache:
The list of archetypes is kept in the cache for a specified number of uses before it is reloaded. To change the number of users, set `<eng_instance_name>.db.engine.tune.max_archetype_list_use`. The default is 100000.

For example:

```
movie_pref.db.engine.tune.max_archetype_list_use = 150000
```

**Enabling or disabling the use of archetypes:**

To turn on or off the use of archetypes, set `<eng_instance_name>.db.engine.tune.consult.archetype_for_list` to `true` or `false`. The default is `false`. For example:

```
movie_pref.db.engine.tune.consult.archetype_for_list = true
```

**User predictability:**

This section contains the following information regarding user predictability:

**Maximum number of mentors assigned to each user:**

Use the following setting to specify the number of mentors to be assigned to each user. The number of actual mentors can be less than the maximum setting but never greater than the value specified. For example:

```
<mentor_set_name>.max_mentors = 50
```

**Note:** Use a value between 50-100. See "Maximum number of mentors assigned to each user." for guidelines on setting `<mentor_set_name>.max_mentors`.

**Maximum ratings a user needs before becoming a mentor:**

The following setting specifies the maximum number of ratings a user can have to become a mentor. The sifter uses this setting if the Preference Engine is using that mentor set. You can specify the value as one of the following:

- A percentage of the total number of items in the database. For example:
  
  `<mentor_set_name>.max_mentor_ratings = 75%`

- The maximum number of ratings (that is, without a percentage). For example:
  ```
  <mentor_set_name>.max_mentor_ratings = 18
  ```

Use the following guidelines for setting `<mentor_set_name>.max_mentor_ratings`:

**Table 121. Guidelines for Setting the Maximum Ratings**

<table>
<thead>
<tr>
<th>Total Number of Items in Database</th>
<th>Suggested Setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt; 1000</td>
<td>&lt;= 100</td>
</tr>
<tr>
<td>1000-5000</td>
<td>&lt;= 500</td>
</tr>
<tr>
<td>5000-10,000</td>
<td>&lt;= 1500</td>
</tr>
<tr>
<td>10,000+</td>
<td>&lt;= 1500</td>
</tr>
</tbody>
</table>

**Note:** For more information on how the sifter makes use of `<mentor_set_name>.max_mentor_ratings`, see "Maximum number of mentors assigned to each user."
Maximum transactions a user needs before becoming a mentor:

The following setting specifies the maximum number of transactions a user can have before becoming a mentor. The sifter uses this setting if the Purchase or Clickstream Engine is using that mentor set. You can specify the value as one of the following:

- A percentage of the total number of items in the database. For example:
  `<mentor_set_name>.max_mentor_xactions = 75%`
- The maximum number of transactions (that is, without a percentage). For example:
  `<mentor_set_name>.max_mentor_xactions = 18`

Use the following guidelines for setting `<mentor_set_name>.max_mentor_transactions`

<table>
<thead>
<tr>
<th>Total Number of Items in Database</th>
<th>Suggested Setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt; 1000</td>
<td>&lt;= 100</td>
</tr>
<tr>
<td>1000–5000</td>
<td>&lt;= 500</td>
</tr>
<tr>
<td>5000–10,000</td>
<td>&lt;= 1500</td>
</tr>
<tr>
<td>10,000+</td>
<td>&lt;= 1500</td>
</tr>
</tbody>
</table>

Note: For more information on how the sifter makes use of `<mentor_set_name>.max_mentor_xactions`, see “Guidelines for configurable recommendation dynamics” on page 398.

Minimum number of ratings for user recommendations:

Set `<eng_instance_name>.engine.titan.predictable.min_ratings_cutoff` to the minimum number of ratings a user must make before the Preference Engine will make recommendations for that user. The default is 2. For example:

`movie_pref.engine.titan.predictable.min_ratings_cutoff = 3`

Recomputing Preference Engine predictions:

The `<eng_instance_name>.engine.titan.recomputation_bound` configuration parameter specifies the percentage change allowed in a user's ratings before the LikeMinds server recomputes the user's predictions. For example:

`music_pref.engine.titan.recomputation_bound = 10.0`

Ordinarily, the LikeMinds server generates predictions based on a user's mentors, which the sifter computes and makes available to the database. When a user has no mentors, perhaps because he or she has just arrived at the site, or when the user's ratings or transactions have changed beyond the percentage specified here, the LikeMinds server selects mentors from a reduced set of candidates and recomputes the user's predictions.

Use this setting with caution, as selecting mentors is a relatively expensive operation: a low percentage setting can lead to excessive CPU load with little or no gain in prediction quality. A higher setting will improve performance, but predictions may be less accurate.

Default Preference Engine recommendations:
The Preference Engine reads the value from the score field in the Lps_Item_Data table to determine which items are popular. Hence, a higher score means that the item is more popular. When buildstats runs, it updates the score field to the average rating for that item; it generates this value based on the rating set defined in the db.applic.rating.source parameter. To get reliable default recommendations, run the buildstats utility on a regular basis. By default, the installer sets buildstats to run once a day. Some applications use their own business logic to assign the score to items. If you want to override the score field value to use your application's scoring instead, run buildstats using the -noscore argument.

Configuring LikeMinds utilities: You can configure the following settings for the buildstats and buildvisit background utilities:

- Ratability parameters
- Repeated items in the visit list

You can use the buildstats utility for all of the Recommendation engines except for the Item Affinity Engine. The buildvisit utility is only used for the Preference Engine.

Configuring the sifter for mentor selection:

The sifter finds mentors for users, using the information from the rating or transaction data for the LikeMinds server engines. The sifter is used by all of the LikeMinds server engines except for the Item Affinity Engine, which uses the accumulator utility. Figure 12 illustrates how the sifter works:

1. The sifter reads user, ratings, and transactions data from the appropriate tables.
2. The sifter then uses the ratings or transactions to assign mentors to the user.
3. The sifter updates the Lps_User_Data table with the new mentor information.

Figure 12. How the Sifter Works

Sifter-specific mentor set configuration:

You can specify the following categories of sifter-specific configuration parameters for mentor sets:

- The engine to be used for mentor tables to be rebuilt
• Time interval for checking the sift priority
• Number of sift priority users per batch
• Maximum number of mentors assigned to each user
• Minimum ratings a user must have before becoming a mentor
• Maximum ratings a user must have before becoming a mentor
• Minimum transactions a user must have before becoming a mentor
• Maximum transactions a user must have before becoming a mentor
• How to set the mentor pool size
• sifter sleep time for when the Lps_User_Data table sift_pri field is 0
• Running multiple sifters
• Number of threads to sift users
• Recomputing (rebuilding) the mentor pool
• Pausing the sifter during heavy database access

Note: For more information on the sifter, see “Configuring the sifter for mentor selection” on page 401.

How the mentor selection process works:

In order to fully understand the sifter, it is important to have a clear idea of how the mentor selection process works, and how to set configuration parameters that increase the accuracy of your recommendations. You should understand the following terms before proceeding:

• **Collaborative Filtering** -- Collaborative filtering (CF) is a technology that calculates the similarity between users. It uses the behaviors of those who most closely resemble any given user as a functional basis for making predictions and recommendations for that user. Given that definition, the process by which “…the entire population of users is analyzed, their fitness as mentors is calculated, and they are assigned as mentors to individual users…” is critically important to the ultimate recommendations that come from the collaborative filtering approach. All of the Like Minds server engines, except the Item Affinity Engine, use collaborative filtering.

The data used to decide on the levels of similarity can come in a variety of forms. From a stream of self-supplied ratings made explicitly to get recommendations back, to clickstream events that comprise the sequence, duration, and outcome of a Web-surfer’s session, to data from a company’s legacy databases (such as transactions, demographics, or credit events)—all can form the necessary basis for making similarity calculations. From those similarity measures, this data can result in the measured recommendations from those users deemed most similar to any given user.

• **Mentor** -- A like-minded user that is used as the basis for recommendations for new users. Every user is assigned mentors by the sifter program, whose stored preferences are judged to be like-minded to the new user.

• **Mentor Pool** -- While the purpose of mentors is to form the basis for recommendations for those users deemed most similar to them, in its most basic form the mentor pool should reflect a representative sample of users in the transaction set for which the recommendations are required. And despite the clearly required emphasis on similarity, no recommendation process can make lucid suggestions without a concomitant space of dissimilarity. We might base our final recommendations on the similar, shared tastes discovered in our analysis of the users being considered for entry to the mentor pool, but it is
truly the confluence of similarity in purchases with some difference in the items purchased that make the LikeMinds server collaborative filtering-based recommendations possible.

- **Sifter** -- Creates mentors by analyzing stored user transactional data. The sifter runs in the background when you run the LikeMinds server.
- **Coverage** -- The volume or number of items rated or transactions performed.

**Mentor selection and assignment:**

Several factors determine the fitness of any user as a mentor. Similarity to any other user is the final arbiter of any mentor's fitness to make recommendations for a specific user. Yet it is **coverage** (the volume of number of items rated or transactions performed) that is most important when forming the mentor pool. (The mentor pool is a superset of the final mentors chosen to make the recommendations.) Although a case could be made for using the extent of the unshared purchase space as another dimension of dissimilarity, the shared space is where we find the most data (and the most predictive data) for making the required similarity distinctions within the user population.

In a nutshell, while similarity to the user is important, it is the ability of a mentor to contribute items outside of any user's typical purchasing space into the final pool of possible recommendations that qualifies the user as a possible mentor.

For this purpose, the first step the sifter uses in mentor assignment is to periodically create a new mentor pool (see lps_rtg_pool and lps_trx_pool) in an effort to collect a representative sample of experienced users who will then be considered as potential mentors for any user who requires recommendations.

The second step in the mentor selection process is to assign mentors from the mentor pool to be mentors for specific users. The challenge is to create a fair balance between the similarity and the coverage of users being considered as mentors. You can configure the sifter to emphasize similarity, coverage, or to automatically determine which dynamic to emphasize for each user in the final assignment of mentors to users.

**Number of sift priority users per batch:**

To specify how many users to sift during a single batch; for example:

\[<\text{mentor\_set\_name}>.pri\_list\_size = 10\]

**Pausing the sifter during heavy database:**

You can pause the sifter during times of heavy database access. This frees up database resources for other activities during heavy database activity. For example, to put the sifter asleep at 11 a.m.:

\[<\text{mentor\_set\_name}>.pause\_sifting\_at= 0 11 * * *\]

If you set \(<\text{mentor\_set\_name}>.pause\_sifting\_at\), use the following setting to wake up the sifter afterwards. For example, to wake the sifter up at 4 p.m.:

\[<\text{mentor\_set\_name}>.resume\_sifting\_at= 0 16 * * *\]

**Sifter performance considerations:**
The sifter places high processing demands on the system running it. For this reason it is important to tune it with consideration of its usage and environment. Use the following simple guidelines when tuning the sifter:

- Schedule the sifter to build the mentor pools during hours of the day when there is low traffic volume, for example, at 3 a.m.
- Tune the number of processing threads used by the process. You can configure this setting for each mentor set.
  
  To optimize the resource utilization of the sifter, set the number of threads to spawn to twice the number of CPUs in the machine that the accumulator is running on. The LikeMinds Recommendation Engine configuration value that controls this is `item_affinity_set.item_affinity.num_threads`.
- Distribute the sifter onto a separate server from the LikeMinds Recommendation Engine.
- Run multiple sifter processes distributed across multiple machines for greater scalability.

Recomputing (rebuilding) the mentor pool:

For better recommendations, you must recompute, or rebuild, the mentor pool at least once a day, ideally at a time when the database has little usage. This allows new users to become mentors and removes current mentors who are no longer good mentors. If you are running multiple instances of the sifter, use the following procedure to recompute the mentor pool:

1. Disable all instances of the sifter except for one, which you will use as the master sifter instance. Rebuilding the mentor pool uses a large amount of database resources, so it is better to use only one sifter to rebuild the mentor pool.
   
   For the sifter instance to be used as the master, type the following:
   
   `<mentor_set_name>.disable_build_mentor_pool = false`
   
   For each sifter instance to be disabled, type the following:
   
   `<mentor_set_name>.disable_build_mentor_pool = true`

2. Rebuild the mentor pool on the master sifter instance by specifying a time for the recomputation to take place.
   
   Because rebuilding the mentor pool requires heavy database access, schedule it for a time when the database has little usage. For example, the following setting is for 2 a.m.:
   
   `<mentor_set_name>.build_mentor_pool_at = 0 2 * * *`
   
   The times are interpreted as follows:

3. For each sifter instance you disabled, set a time to reload the mentor pool after it has been rebuilt.
The sifter will destroy the current mentor pool and reload the pool from the mentor pool table specified in the `<mentor_set_name>.mentor_pool.table` setting. Usually, you should set it for about an hour after the sifter instance has been reloaded.

For example, to reload the mentor pool at 3 a.m.:

```
<mentor_set_name>.reload_mentor_pool_at = 03***
```

**Running multiple sifters:**

Use the following settings to run multiple instances of the sifter and have all the sifters share some configuration parameters and override others. Running multiple sifters simultaneously allows you to distribute load on very large systems.

To override a particular default parameter, add a prefix to the parameter. For example, to reload the mentor pool for the MovieSift sifter instance at 4 a.m.:

```
MovieSift.<mentor_set_name>.reload_mentor_pool_at = 04***
```

If you run a sifter with the following setting,

```
sifter -config MovieSift -conf <lps config file>
```

that sifter will use the `MovieSift.<mentor_set_name>.reload_mentor_pool_at = 03***` setting.

**Preventing multiple sifters from sifting the same user:**

When you run multiple sifters simultaneously to distribute load, use the following setting to prevent multiple sifters from sifting the same user in the Lps_User_Data table. This setting limits the sifter to look at only a certain group of users, or eliminates a group of users from being sifted. For example, to allow the sifter to sift only users whose IDs are greater than 100000, and to ignore users whose IDs are less than 100000:

```
<mentor_set_name>.constraint = user_id > 100000
```

You can specify any field in the Lps_User_Data table for this parameter. For example, assuming there is an age column in the Lps_User_Data table, you could use the following setting to constraint the sifter to look at all users whose IDs are greater than 10000 and who are over 25 years of age:

```
<mentor_set_name>.constraint = user_id > 10000 and age > 25
```

**Number of threads to sift users:**

To set the number of threads to use for sifting users in lpsconfig:

```
<mentor_set_name>.num_sift_threads = 3
```

**Sifter sleep time when the Lps_User_Data sift_pri field is 0:**

The Lps_User_Data sift_pri field values must be greater than 0 in order for the sifter to get useful data (that is, updated mentors) for users. If set to true, the following setting causes the sifter to sleep for the specified number of seconds by `<mentor_set_name>.pri_check_interval` if the sift_pri field values in the Lps_User_Data table are 0. If you set the parameter to false, the sifter sifts random users when there are no users with sift_pri greater than 0.

```
<mentor_set_name>.sleep_on_no_sift_pri = true
```
**Note:** For information on `<mentor_set_name>.pri_check_interval`, see “Time interval for checking sift priority.”

**Time interval for checking sift priority:**

To specify the interval in seconds for the sift to check the sift priority value for a user, use the following setting. The default is 20 seconds.

 `<mentor_set_name>.pri_check_interval = 40`

**Note:** You can also use this setting when you are setting the time for the sift to sleep during busy periods of database access. See “Sifter performance considerations” on page 403.

**Ratability parameters:** The buildvisit background utility computes a ratability value for each item in the database. It is only used for the Preference Engine. The LikeMinds server presents items for rating in decreasing order of ratability; items with the same ratability are presented in random order.

The ratability value is computed as

\[ ratability = A_i + B_i + C_i \]

where \( A_i \) is the popularity of item \( i \):

\[ A_i = K \cdot (\log_{10}(num\_rating))^{rpow} \]

\( num\_rating \) is the number of ratings for the item.

To specify \( K \), set `db.ratability.num_rating.coefficient` (default is 1.0):

`db.ratability.num_rating.coefficient = 1.0`

To specify \( rpow \), the rating power, set `db.ratability.num_rating.power` (default is 1.0):

`db.ratability.num_rating.power = 1.0`

\( B_i \) is a measure of how controversial the item is:

\[ B_i = K \cdot (rating\_std\_dev)^{stdpow} \]

\( rating\_std\_dev \) is the standard deviation of the ratings for the item.

To specify \( K \), set `db.ratability.stddev.coefficient` (default is 1.0):

`db.ratability.stddev.coefficient = 1.0`

To specify `stdpow`, the standard deviation power, set `db.ratability.stddev.power` (default is 1.0):

`db.ratability.stddev.power = 1.0`

\( C_i \) is a weight based on the age of the item:

\[ C_i = K \cdot \exp(-\text{(current\_year - release\_year)}) \cdot \exp^\text{apow} \]

\( release\_year \) is the value of the year field in the `Lps_Item_Data` table for the item.

To specify \( K \), set `db.ratability.age.coefficient`. You can set this to zero if you do not want to consider age in determining ratability (default is 2.7):

`db.ratability.age.coefficient = 2.7`
To specify $p$, set `db.ratability.age.power` (default is 0.5):

```
db.ratability.age.power = 0.5
```

**Repeated items in visit list:** To specify whether the `buildvisit` utility should repeat an item in the visit list to be shown to a user, set `db.visitlist.ratability.duplication_threshold`. Items with ratability values greater than this threshold may be repeated.

For example:

```
db.visitlist.ratability.duplication_threshold = 6
```

**Configuring the Clickstream Engine:**

The Clickstream Engine generates recommendations based on a record of user shopping behavior as the user navigates through a Web site; that is, a history of user "clicks" during the user's website visit. It uses data such as the items the users view, click, and add to their shopping carts.

You can use the "User predictability for the Clickstream Engine" setting to specify recommendation behavior for the Clickstream Engine.

**User predictability for the Clickstream Engine:**

This section contains the following information regarding user predictability:

- The minimum number of clickstream activities required for a user before he or she can receive recommendations.
- The number of mentors the Clickstream Engine will examine before it checks to see whether the user is predictable.
- The percentage change allowed in a user's transactions before the LikeMinds server recomputes the user's predictions.
- Default recommendations.

**Minimum number of Clickstream activities for a user:**

Set `<eng_instance_name>.engine.saturn.tune.predictable.min_xactions_cutoff` to the minimum number of transactions a user must make before the Clickstream Engine will make recommendations for that user. The default is 10.

For example:

```
movie_click.engine.saturn.tune.predictable.min_xactions_cutoff = 2
```

**Minimum mentors the engine examines for predictability:** When it determines whether it can make recommendations for a given user, the Clickstream Engine must examine the user's mentors. `<eng_instance_name>.engine.titan.predictable.loop_check_cutoff` specifies the number of mentors the Clickstream Engine will examine before it checks to see whether the user is predictable. The default is 10.

For example:

```
movie_pref.engine.titan.predictable.loop_check_cutoff = 12
```

If the user is not predictable, the engine will examine another group of `<eng_instance_name>.engine.titan.predictable.loop_check_cutoff` mentors, and then check to see whether the user is predictable. It will continue until it finds that the user is predictable or it runs out of mentors.
Recomputing Clickstream Engine predictions:

The `<eng_instance_name>.engine.saturn.recomputation_bound` configuration parameter specifies the percentage change allowed in a user's transactions before the LikeMinds server recomputes the user's predictions. For example:

```
music_click.engine.saturn.recomputation_bound = 10.0
```

Ordinarily, the LikeMinds server generates predictions based on a user's mentors, which the `sifter` computes and makes available to the database. When a user has no mentors, perhaps because they have just arrived at the site, or when the user's ratings or transactions have changed beyond the percentage specified here, the LikeMinds server selects mentors from a reduced set of candidates and recomputes the user's predictions.

Use this setting with caution, as selecting mentors is a relatively expensive operation: a low percentage setting can lead to excessive CPU load with little or no gain in prediction quality. A higher setting will improve performance, but predictions may be less accurate.

Default Clickstream Engine recommendations:

The Clickstream Engine uses transaction data stored with a special user to generate default predictions. The `buildstats` utility generates clickstream transaction data, based on the transaction set named in the `db.applic.transaction.source` parameter. This transaction data is a summary of all clickstream transaction data for users, based on transaction data from all configured transaction sets. `buildstats`, the LikeMinds server, and other utilities identify the special user as one whose `Lps_User_Data.mentor_type` field is set to "z". There should be exactly one user with this setting, not more. `buildstats` will automatically generate this user if custom IDs are not configured; otherwise, you need to create the user by hand, since `buildstats` cannot know what restrictions apply to the custom IDs.

Run the `buildstats` utility on a regular basis. By default, the installer sets `buildstats` to run once a day. Some applications use their own business logic to assign the score to items. If you want to override the score field value to use your application's scoring instead, run `buildstats` using the `noscore` argument.

Configuring the Item Affinity Engine:

For every cross-selling transaction in the user's shopping history, the Item Affinity Engine derives its calculations from the following statistics:

- **Support**: This is the number of times an item pair is involved in the item affinity set you defined, divided by the total number of paired item affinity events. Item affinity events can be critical page views, purchases, shopping cart adds or drops, or combinations of these events. In other words, the support is the percentage of the time that the item pair occurred together, relative to all transactions.

- **Prediction**: Using the support statistic as a basis, the prediction is the number of times an item-to-item pair occurs together, divided by the singular support of the first item of that item pair. In other words, the prediction is the probability of the item pair, given the first item in the pair.

- **Confidence**: Using the prediction statistic as a basis, confidence is the prediction divided by the singular support of the second item of the pair for which the prediction is computed. That is, the confidence is the probability of the item pair given the first and second item in the item pair.
In general terms, you use the Item Affinity Engine as follows:

1. Define the shopping cart analysis requirements.
   This refers to the scope and filters for predictions you plan to use for the shopping cart analysis. This definition is very flexible. For example, the shopping cart can be based on the user's entire shopping history, on seasonal product sales, on the user's current session, user page views critical to an item's purchase, and so on.

   The type of purchases you expect from your users will determine the type of shopping basket you define. For example, a session-based shopping cart would be suitable for a grocery store site, where users tend to purchase a variety of items at once. A lifetime shopping cart is suitable for a site where users tend to purchase one large item occasionally, such as expensive electronic equipment.

2. Configure an item affinity set for the shopping cart analysis.
   The item affinity set defines your shopping cart analysis requirements. Similar to mentor sets or transaction sets, the item affinity set defines the type of data you want to collect. The item affinity set starts by collecting transactions from a transactions input table similar to Lps_User_Trx. You can include filters to query fields in the input table, specify the types of transactions to query, the number of highly associated items to consider, and so on, for the shopping cart definition. Finally, you specify an output table to which the results of the analysis are written. The output table must have the same fields and datatypes as the Lps_MBA_Scored table.

3. Run the accumulator utility to collect the item affinity set data and populate it into the output table specified in the item affinity set.
   Similar to the sifter, which is used for the other LikeMinds server engines, the accumulator uses the item affinity set data to make the support, prediction, and confidence calculations (described earlier). It writes its findings to the output table specified in the item affinity set.

   You should configure the accumulator to run at times of low database usage, since it can make heavy use of system resources.

**Specifying recommendation behavior:**

You can configure several parameters that affect the way the LikeMinds server generates recommendations.

You can set the following recommendation behavior:

- Allowable rating values
- Allowable confidence levels
- Prediction quality
- Best bets

The parameters in this section affect the way the LikeMinds server generates recommendations. If you change them, be sure that the results will still be appropriate for your application. Because the Movie Site application expects these parameters to be set to their default values, Movie Site application developers should use extreme caution when modifying them.

Parameters that apply to the background applications can also affect the way recommendations are generated. See “Configuring the sifter for mentor selection” on page 401.

**Allowable rating values:**
Learn about the parameters that govern the allowable range of ratings.

Specify the allowable range of ratings by setting `db.applic.param.lowestrating` (default is 0) to the lowest allowable rating and `db.applic.param.numberratingvalues` (default is 13) to the number of possible ratings. This setting also determines the recommendation value. This example specifies a rating scale of zero to twelve:

```
db.applic.param.lowestrating = 0
db.applic.param.numberratingvalues = 13
```

The parameters `db.applic.param.wontrate` (default is -1) and `db.applic.param.unrated` (default is -2) set special allowable rating values that are not part of the normal rating scale. Setting an item to unrated marks the rating for deletion. `wontrate` means that a user has specifically indicated that he or she does not plan to rate the item.

```
db.applic.param.wontrate = -3
db.applic.param.unrated = -2
```

The LikeMinds server ignores ratings which are neither in the specified range of rating values nor equal to `db.applic.param.wontrate` or `db.applic.param.unrated`.

**Note:** You can use these parameters for all of the Recommendation engines except for the Item Affinity Engine.

**Allowable confidence levels:**

The LikeMinds server assigns a confidence level to each recommendation based on how many users have rated the recommended item and how similar the ratings are to each other. "Confidence" refers to the accuracy of the prediction. Learn how to set allowable confidence levels.

You can specify the range of confidence values by setting `db.applic.param.lowestconfidence` (default is 1) to the lowest confidence level and `db.applic.param.numberconfidenceLevels` (default is 4) to the number of confidence levels. You can set confidence levels for all of the Recommendation engines.

```
db.applic.param.lowestconfidence = 1
db.applic.param.numberconfidenceLevels = 4
```

**Prediction quality values:**

The "quality" value refers to the degree a user will like an item. The LikeMinds server presents predictions in decreasing order of quality.

Recommendation quality is calculated using this equation:

```
quality = confidence^{mpow} \times (pred \ rating - mean \ pred \ rating) + K \times confidence^{pow}
```

where:

- **confidence** is the confidence value for the recommended item.
- **pred rating** is the predicted rating for the item.
- **mean pred rating** is the mean recommended rating for all items for the user.

To specify `mpow`, set `db.applic.param.BestBets.MultPower`:

```
db.applic.param.BestBets.MultPower = 0.5
```
To specify K, set `db.applic.param.BestBets.Coefficient`:

```
db.applic.param.BestBets.Coefficient = 0
```

To specify pow, set `db.applic.param.BestBets.Power`:

```
db.applic.param.BestBets.Power = 1.0
```

**Note:** You can use these parameters for all of the Recommendation engines except for the Item Affinity Engine.

**Best Bets values:**

The LikeMinds server generates recommendation vectors that include all recommendations for a given user in order from best quality to the least. You can configure the number of items to be returned as Best Bets and the maximum percentage of the total recommendation vector to include in the Best Bets list.

The LikeMinds server produces Best Bets by beginning with the highest-quality item in the recommendation vector and listing items in decreasing order of rating quality.

To specify how many items the LikeMinds server should return as Best Bets, set `db.applic.param.BestBets.Threshold`. The default value of this parameter is the number of rating levels (specified by the `db.applic.param.NumberRatingValues` parameter) divided by 2. For example, if `db.applic.param.NumberRatingValues` were set to 12, the default would be 6. You can specify a different value, for example:

```
db.applic.param.BestBets.Threshold = 7
```

Set `db.applic.param.BestBets.list.cutoff` to the maximum percentage of the total recommendation vector to include in the Best Bets list. For example:

```
db.applic.param.BestBets.list.cutoff = 50
```

**MovieSite Sample**

The MovieSite documentation focuses on six aspects of LikeMinds capabilities:

**Best Bets**

The Best Bets task provides a list of recommended movies for each user. These recommendations are calculated based on a user's previous ratings, as well as the ratings of other similar users, using collaborative filtering. The system determines users' similarities by comparing their ratings history among all users. For example, if both user A and user B have many similar ratings, but user B has rated many more different movies; user B might make a great mentor for user A. It is from the set of mentors (user B and other similar users) that LikeMinds makes recommendations, in this case a movie recommendation for user A.

**Get Items to Rate**

The Get-Movies-to-Rate task retrieves movies, which the user has not yet rated, from the database. These movies are selected based on the impact that a rating from the user would have on LikeMinds' ability to recommend movies (Best Bets). (See the task example.)

**View the User's Previous Ratings**

This task retrieves ratings that a user previously entered from the LikeMinds database. An item that has previously been rated can always be given a different rating. (See the task example.)
Log Item Ratings

The Personalization rating bean logs each user rating to the LikeMinds database. There are a few requirements to remember when using the rating bean:

- The session attribute “pzn.userName” must be set with the user resource ID. This is how LikeMinds pairs the user with an item rating.
- If you log an item that previously did not exist in the LikeMinds database table, an entry will automatically be created for that item.
- In the same way, if an item is logged for a user who does not exist in the LikeMinds database table, LikeMinds creates an entry for that user. This is how a “new user” is to be entered into the LikeMinds system.
- The resource collection that is sent as a parameter to the logging bean is case sensitive.

Database Access

Developers are responsible for maintaining their own database tables. In MovieSite's case, this includes MovieSite's user table, item table, and also genre tables. All MovieSite tables begin with MS* and LikeMinds tables begin with Lps*. To implement filtering, the filter ID columns must be added to the LPS_ITEM_DATA table. These filter IDs are populated when a movie is logged for the first time, or the IDs are set directly. For more information on filtering, see the section about filtering recommendations that follows.

Filtering Recommendations based on filter IDs

LikeMinds implements a filtering system for each of the Personalization rules that query LikeMinds. This system filters items at the LikeMinds level according to any number of filter ID's associated with each item. Only the items which satisfy the filter are surfaced to the content spot in the JSP.

Exploring Movie Site: Movie Site demonstrates the LikeMinds Recommendation Engine, guiding you through a Personalization scenario that is based on factual data from a popular site on the internet. The website uses a personalization solution to analyze visitor behavior and to recommend individualized content information and services while the visitor is actively engaged on the site.

Notes:

- The provided Resource Collections are for IBM DB2 Universal Database™ Enterprise Server Edition and DB2® for z/OS only. If you want to use the sample on other database types, you can regenerate them using RAD. For information about Resource Collections, see the topic Resources, resource instances, and resource collections.
- It is assumed that you are the Movie Site user, likeminds. This user has previously rated some movies and registered preferences as configured in the installation.

To set up the Movie Site sample, choose the appropriate option:

- AIX/Solaris/Linux: ./ConfigEngine.sh cfg-likeminds-samples
  -DWasPassword=password -DLikemindsDbPassword=password
- Windows: ConfigEngine.bat cfg-likeminds-samples -DWasPassword=password
  -DLikemindsDbPassword=password
- IBM i: ConfigEngine.sh cfg-likeminds-samples -DWasPassword=password
  -DLikemindsDbPassword=password
z/OS: The Movie Site sample is installed as part of the configuration of the LikeMinds utilities and samples. The Customization Dialog for Portal configuration provides the panels for Application configuration tasks, where you configure Personalization, which includes the installation and configuration of the LikeMinds samples. Refer to the worksheets for Configuring the LikeMinds samples before you begin to set up the Movie Site sample. The Customization Dialog for the installing and configuring the LikeMinds samples generates the EJPSPZLC and EJPSPZLR jobs for you to submit. After the EJPSPZLC job finishes, you can start the Movie Site sample application.

**Note:** Check the output for any error messages before proceeding with the next task. If any of the configuration tasks fail, verify the values in the wkplc.properties file.

Complete the following steps to start the application:
1. Open the WebSphere Integrated Solutions Console.
2. Go to http://machine:port_number/ibm/console where port_number is the port number for WebSphere Portal.
3. Click Applications > Application Types > WebSphere enterprise applications.
4. Select Movie Site Application, and click Start.
5. Rate movies using the Movie Site home page.

*Table 123. A quick demo of how to rate movies using the Movie Site home page*

<table>
<thead>
<tr>
<th>Step number</th>
<th>Description</th>
<th>Action to be taken</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Navigate to the Movie Site Home Page. For this demo we will concentrate on the recommendation engine which uses mathematical algorithms to provide collaborative filtering.</td>
<td>Open a separate browser window to the Movie Site home page. The URL is <a href="http://hostname:port/MovieSite">http://hostname:port/MovieSite</a> where port is the port number for WebSphere Portal</td>
</tr>
<tr>
<td>2</td>
<td>Log in as the LikeMinds user. Log on as the user, likeminds. This login name is the default user who has registered to the site and has rated some movies.</td>
<td>1. Type likeminds in the login field. 2. Type likeminds in the password field 3. Click Enter.</td>
</tr>
</tbody>
</table>
Table 123. A quick demo of how to rate movies using the Movie Site home page (continued)

<table>
<thead>
<tr>
<th>Step number</th>
<th>Description</th>
<th>Action to be taken</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>3</strong></td>
<td>Sample previous ratings. Look at the movie ratings by selecting <strong>Sample Previous Ratings</strong>. This section of the site contains a few sample movie ratings the user Likeminds previously placed.</td>
<td>1. Click <strong>Sample Previous Ratings</strong>. 2. Select <strong>Get more movies</strong> from the menu. 3. Click the <strong>To Lobby</strong> icon. You can show all of the movies or just certain rated movies by selecting one of the ratings. You can look at the ratings of other movies by clicking <strong>Get More Movies</strong>. This demo allows you to rate movies. For more movies, type the name of a movie into the search panel to see if specific movies are stored in the database.</td>
</tr>
<tr>
<td><strong>4</strong></td>
<td>Use the SuperRater to rate a movie. In Super-Rater mode, clicking the drop-down menu associated with each movie of the movie reveals a rating choice. You can rate any movies on the list and click <strong>Submit</strong> to log your rating choice. If you click the movie title it will link you to information about the movie using the imdb.com site. Notice the prediction of the recommendation engine of how much you, as the LikeMinds login user, would like this movie.</td>
<td>1. Click <strong>To Lobby</strong>. 2. Click <strong>Rate more movies</strong>. 3. Click <strong>Turn on Superrater</strong>. 4. Click the dropdown menu associated with the rating to see the options. 5. Click <strong>Movie title</strong>. <strong>Note:</strong> This opens a new window.</td>
</tr>
<tr>
<td><strong>5</strong></td>
<td>End demonstration.</td>
<td>Click <strong>To Lobby</strong>.</td>
</tr>
</tbody>
</table>

The LikeMinds collaborative filtering technology considers each user to be the center of their own preference cluster, instead of trying to find the best preordained preference category for every user, which is typical of profile-based approaches. LikeMinds recommendations can adapt to these changes much more quickly than aggregated, profile based techniques.

Choose the appropriate task to remove the application:

Windows: ConfigEngine.bat remove-likeminds-samples
- DWasPassword=password - DLikemindsDbPassword=password
AIXSolarisLinux: ./ConfigEngine.sh remove-likeminds-samples -DWasPassword=password -DLikemindsDbPassword=password
IBM i: ConfigEngine.sh remove-likeminds-samples -DWasPassword=password -DLikemindsDbPassword=password
z/OS: ./ConfigEngine.sh remove-likeminds-samples -DWasPassword=password -DLikemindsDbPassword=password

Note: Check the output for any error messages before proceeding with the next task. If any of the configuration tasks fail, verify the values in the wkplc.properties, wkplc_dbdomain.properties, and wkplc_dbtype.properties files.

Using the LikeMinds utilities

Learn about the four utilities LikeMinds uses to update the database, buildvisit (for the Preference Engine), sifter, buildstats, and lpsIAA (for the Item Affinity Engine's accumulator utility).

The LikeMinds utilities are run from the LikeMinds utility servlet.

You must configure these utilities to run at appropriate times. See “Scheduling LikeMinds Events” on page 396 for more information.

Sifter:

The sifter, which is used for all the Recommendation engines except for Item Affinity, assigns mentors to users and updates its assignments as users rate items. When no users are rating items, the sifter updates the mentor assignments of all users based on changes to the pool of available mentors.

The sifter must be running whenever the LikeMinds server is running. However, you can install the sifter across multiple machines to distribute load.

Buildstats:

The buildstats utility updates database statistics. You can use buildstats for all of the Recommendation engines except for the Item Affinity Engine.

When you install the LikeMinds utilities, the installer sets buildstats to run once a day, at 3:30 a.m. This is because the best time to run buildstats is daily, at a time when the LikeMinds server receives relatively little use. The buildstats utility connects directly to the LikeMinds database and processes the transactional or ratings data that is constantly being written to the LikeMinds database. Because this utility does not consume many resources, it is typically installed on the same machine as the LikeMinds server. buildstats performs a number of data analysis functions on the information in the LikeMinds database and persistently writes out the results into the LikeMinds schema. However, the functionality of this and the number of functions to test differ depending on the type of LikeMinds engines in use. If both a Preference and ClickStream are in use, buildstats will, by default, analyze both the ratings and the transactional data.

For ratings data, buildstats writes to the following fields in the Lps_Item_Data table:

- **num_rtg**: Total number of ratings for the item.
- **total_rtg**: Sum of all ratings for the item.
- **total_square_rtg**: Sum of the squares of all ratings for the item.
• **ratability**: Priority for an item’s presentation to users for rating. Ratability is non-negative value. Higher numbers indicate a higher priority for rating.
• **score**: Popularity or unpopularity of an item. If you prefer to have your own applications update this field, you can disable buildstats from writing to score.

For transactional data, buildstats writes to these fields in Lps_User_Trx:
• **value**: Data value associated with the transaction.
• **adj_count**: Adjusted count of transactions. If a new activity is recorded in the transaction table, this value increases. It may diminish over time.

**Accumulator:**

Use the accumulator utility if you are using the Item Affinity Engine. This utility accumulates the number of times every possible item-to-item combination occurs.

Similar to the sifter for the other LikeMinds engines, the accumulator runs in the background processing item event data. It writes this data to the table specified by the `<item_affinity_set>.item_affinity.output.table` setting. The accumulator is installed as part of the PZN_Utilities.ear. It is run through the LikeMinds scheduling mechanism (see [Scheduling LikeMinds Events](#) on page 396).

**Filtering LikeMinds recommendations**

When LikeMinds makes recommendations, it can make the recommendations based on all items in your resource collection, or it can limit the predictions to only items that have certain characteristics.

Tell LikeMinds about an item by including key/value pairs describing its characteristics as you log actions and ratings that occur against it. The format for the key/value pair is

Key = LMFilter.<item_characteristic>
value = <value>

For example, to tell LikeMinds that an item's color is blue, and its category is sports, you would add the 2 key/value pairs:

LMFilter.color,blue
LMFilter.category,sports

Specify which characteristics that you want LikeMinds to use in making predictions by setting request attributes in the RequestContext object immediately before the content spot that contains the LikeMinds rule. To specify characteristics for filtering, do the following:
• Add a request attribute that tells LikeMinds which characteristic or characteristics you want to filter on. The name of that attribute should be LMFilter and the value should be a string or an array of strings, where the string or each string in the array is a characteristic that you would like to filter on.

For example, to return predictions only from among items whose category is "clearance" and season is "spring" or "summer", you would add the following code before the content spot:

```java
com.ibm.websphere.personalization.RequestMapping.context =
com.ibm.servlet.personalization.context.PersonalizationContext.getRequestContext(httpRequest);
context.setRequestAttribute("LMFilter", new String[] { "LMFilter.category", "LMFilter.season" });
```
To return predictions only from items whose color is blue, you would add the following code before the content spot:

```java
context.setRequestAttribute("LMFilter.color", "blue");
```

### Feedback and analytics

Personalization provides a complete logging framework for collecting data on how visitors are using your Web site. If Feedback is enabled, data is automatically collected about each Personalization rule that is fired. In addition, development tools enable Web sites to collect a variety of data related to visitors' actions and behavior. By default this data is logged to a standard database schema for later analysis and reporting. The framework is also extensible, allowing Web sites to customize and supplement the way data is collected and stored to more fully meet their needs.

Personalization provides a complete logging framework for collecting data on how visitors are using your website. If Feedback is enabled, data is automatically collected about each Personalization rule that is fired. In addition, development tools enable Web sites to collect a variety of data related to visitors' actions and behavior. By default this data is logged to a standard database schema for later analysis and reporting. The framework is also extensible, allowing Web sites to customize and supplement the way data is collected and stored to more fully meet their needs. Using the Personalization Feedback framework you can collect data to help answer questions such as:

- How effective are the new campaign initiatives?
- Which items have the highest clickthrough ratio?
- What is the clickthrough ratio for health care products?
- Which items have the highest view to purchase ratio?
- What is the ratio for health care products?
- Which rules are resulting in the most number of downloads?
- During the month of December, which categories of pages were browsed most often (my special Christmas pages, or the regular part of my site)?

### Feedback subsystem overview

The Feedback listener captures data and writes it to the Feedback database schema for subsequent reporting, a capability which can be used to persist data in ways that specifically meet your requirements.

Logging occurs when a rule or logging bean is present in JSP or servlet a user visits. In either case, during execution an event containing data to be logged is generated and dispatched by the log manager to a set of listeners. Personalization supplies two such listeners. The first is the Feedback listener, which captures data and writes it to the Feedback database schema for subsequent reporting. The second is the LikeMinds listener (LMListener), which captures data for subsequent use by the LikeMinds recommendation engine. Web site developers can optionally extend the provided listeners or create additional custom listeners to persist data in ways that specifically meet their needs.
- Logging events are generated by Personalization rules or logging beans.
- The Feedback subsystem’s multithreaded implementation prevents HTTP clients from waiting.
- The Log Manager maintains internal queue of events.
- Event is dropped on queue then control returned to web client.
- Events are dispatched from the queue to all registered listeners.
- Listeners in turn persist and process the data contained in the logging event.

**Enable logging**
Create and set up the Feedback database. Then enable logging on the runtime server that hosts Feedback data.

Choose the appropriate option to create, set up, and enable logging of the Feedback database:

*Table 124. Operating system options to create, set up, and enable logging of the Feedback database*

<table>
<thead>
<tr>
<th>Operating system</th>
<th>Task</th>
</tr>
</thead>
</table>
| AIXSolarisLinux  | Complete the following steps:  
1. Open the `feedbackService.properties` file from the `wp_profile_root` PortalServer/config/config/services/ directory.  
2. Set `loggingEnabled` to true.  
3. Restart WebSphere Portal. |
### Table 124. Operating system options to create, set up, and enable logging of the Feedback database (continued)

<table>
<thead>
<tr>
<th>Operating system</th>
<th>Task</th>
</tr>
</thead>
<tbody>
<tr>
<td>IBM i</td>
<td>Complete the following steps: 1. Open the FeedbackService.properties file from the <code>wp_profile_root</code>/PortalServer/config/config/services/UserData directory. 2. Set <code>loggingEnabled</code> to true. 3. Restart WebSphere Portal.</td>
</tr>
<tr>
<td>Windows</td>
<td>Complete the following steps: 1. Open the FeedbackService.properties file from the <code>wp_profile_root</code>/PortalServer/config/config/services directory. 2. Set <code>loggingEnabled</code> to true. 3. Restart WebSphere Portal.</td>
</tr>
<tr>
<td>z/OS</td>
<td>To create and set up the Feedback database for WebSphere Portal for z/OS, follow the panels of the Customization Dialog and use the Feedback configuration worksheets.</td>
</tr>
</tbody>
</table>

#### Feedback properties file

The FeedbackService.properties file contains several customizable properties that are used by the Feedback component. These properties are used to enable custom log listeners, to tune the performance of the Feedback data collection system.

The FeedbackService.properties file is installed to the `wp_profile_root`/PortalServer/config/config/services directory.

To modify the properties:
- Edit the FeedbackService.properties file.
- Restart the IBM WebSphere Application Server where the Portal Personalization Runtime is installed.

**loggingEnabled**

Before information about the usage of your site can be recorded, logging must first be enabled. To enable logging, set the `loggingEnabled` property to true.

```
loggingEnabled = true
```

#### schemaName

It is possible to customize the name of the feedback schema used in the database. For most platforms the Personalization installer will create the feedback tables under a schema named FEEDBACK.

For IBM i installations, the `schemaName` property must be set to the name of the database library that contains the Feedback tables (for example, QWPS51).

```
schemaName = FEEDBACK
```
z/OS only: dbPlatform

Note: For IBM DB2 Universal Database for z/OS only.

If the Feedback database is installed on the DB2 for z/OS platform then the dbPlatform property must be set to 390 to identify this fact. It is unnecessary to set this property for all other database platforms.

dbPlatform = 390

logListeners

Log events are generated whenever the log method of a bean is called or whenever a rule is fired. Log listeners process these log events and can store the event data or perform other actions as a result of these events. The Feedback component provides two default log listeners, the LMLListener that collects data for use by the LikeMinds Recommendation Engine and the FeedbackListener that collects data for use in the Feedback reports. Custom log listeners can be used to modify the default behavior of the FeedbackListener or to provide a listener that processes the Feedback events in a user-specified manner.

After being developed, custom listener class files must be placed in the classpath of the server where Personalization run-time is installed. The class names of these listeners must be identified in the logListeners property. Multiple custom listener class names are separated by semicolons:

logListeners = LogListenerClassName1;LogListenerClassName2

logBufferSize

When log events are produced by logging beans or rules, they are placed on a process queue. Each of the registered log listeners consumes events from this queue. This enables listeners to consume log events at independent rates, minimally impacting the performance of other listeners. It also enables the Feedback component to queue events originating from Web page requests with minimal impact on the Web site performance.

The log event queue (or log buffer) has a default maximum size of 10,000 events. The optimal size of this buffer is affected by Web site volume, Web site usage spikes, and many other Web application performance factors. You can change the maximum size of the log buffer by modifying the logBufferSize property in the FeedbackService.properties file:

logBufferSize = 10000

logWarningLevel

As the log buffer begins to fill due to heavy Web site loads or slow log listener performance, a warning message can be logged to the WebSphere Integrated Solutions Console. The logWarningLevel specifies the log buffer load that will trigger warning messages. The logWarningLevel is specified as a percentage value between 0 and 100. Specifying a value of 0 will repeatedly output the current log buffer load whereas a percentage value of 100 will result in no warning messages as the buffer fills. Once the log buffer is full, an error message is logged to the WebSphere Integrated Solutions Console and logging is discontinued until room becomes available in the log buffer. The default value for the log warning level is

420

Web Content Manager Version 8.0
75%. You can change this warning level by modifying the `logWarningLevel` property in the `FeedbackServices.properties` file:

```
logWarningLevel = 75
```

**requestFlushInterval**

Multiple rules or logging beans can be placed on a single Web page or JSP. The Feedback component collects all the data from the rules and log beans on a page and then updates the Feedback database with all of the information for the page (HTTP request). Updating the Feedback database only once per page improves the performance of the Feedback component. It is necessary to determine when all of the data for a page request has been collected. Two mechanisms are used in the current release to determine when the log data for a page request should be written to the Feedback database. The first method is to flush all of the log data for a page request whenever a new page request is received from the same user session. The second method uses a time interval to determine when the response to a page request has been completed and, thus, all the logging data for the request has been generated.

The `requestFlushInterval` applies to this second method of flushing the logging data for a page request. It is the time interval during which no new logging data has been received during a user session after which the log data for the most recent request is assumed to be complete and is flushed to the database. The default value for the request flush interval is 10 seconds. You can modify this interval by changing the `requestFlushInterval` property (specified in milliseconds) in the `FeedbackService.properties` file:

```
requestFlushInterval = 10000
```

**inactiveListenerInterval**

It is possible for a log listener to become inactive due to a run-time error or resource deadlock. The result of such a deadlock would be an accumulation of data in the log buffer that will never be processed. Since the log buffer is finite in size, it will eventually reach capacity. Once it reaches capacity, no new data will be logged and all active listeners will be unable to collect and process additional log data. To protect against this scenario, the activity of all listeners is monitored. If log data is available for a listener to process and the listener remains inactive for a pre-determined, the log listener will be removed from the set of registered log listeners. The default period of inactivity is 2 minutes. You can modify this interval by changing the `inactiveListenerInterval` property (specified in milliseconds) in the `FeedbackService.properties` file:

```
inactiveListenerInterval = 120000
```

**workManager**

If running in an environment that supports it, the Feedback component uses the WebSphere Application Server `workmanager` system for asynchronous operations. Using the `workManager` property, you can specify the name of the `workmanager` that Feedback will use.

```
workManager = wm/default
```
**Rule logging**

If logging is enabled, rule information is automatically logged whenever a rule is executed.

When a content spot's rule retrieves content, a RuleEvent object is constructed from the request, campaign name, rule name, resource collection name, and items. This event object is routed to all registered log listeners and the logged data are stored in the Feedback schema.

**Logging beans**

Use logging beans to log data about a Web site visitor's actions, category interests, and ratings.

You can also use custom beans to log information specific to your Web application. You insert logging beans into your JSPs. The types of logging beans are:

- Action beans
- Rating beans
- Category beans
- CustomLog beans
- Page View beans

**Action beans:**

You use Action beans to log specific actions of your Web site visitors.

As a Web site visitor navigates your Web pages, an Action bean logs the visitor's events within the session. Action beans maintain action information for the current session including the actions logged and their corresponding log counts. This session data can be used within rules from your JSPs. Action information is also routed to all registered log listeners.

All actions are logged to the Feedback schema and to the user session. The actions logged include the pre-defined actions listed in Table 125 on page 423 and user defined actions. The actions that are logged to the Feedback schema and user session do not need to be pre-defined.

If you have LikeMinds installed, only the pre-defined actions listed in Table 125 on page 423 are logged to the LikeMinds schema. These actions must be defined to LikeMinds in the lps_trx_type table. (You can also add to the list of predefined actions by adding items to the lps_trx_type table.) The actions logged through an Action bean provide the transaction data necessary for the LikeMinds Clickstream Engine. This set of actions may be accessed after deployment and configuration of a Web application.

Actions can be logged with or without respect to a specific resource. For example, the “OrderCancel” pre-defined action does not apply to a specific resource whereas the “BrowseContent” pre-defined action applies to a specific content resource.

You can create rules based on the activity of the current session, for example:

```
WebSite Visitor is
  Undecided when
    current Action Count.ShoppingCartDelete > 0
  Otherwise Decided
```
<table>
<thead>
<tr>
<th>Action ID</th>
<th>Action Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>DetailedView</td>
</tr>
<tr>
<td>1001</td>
<td>Purchase</td>
</tr>
<tr>
<td>1002</td>
<td>Rated</td>
</tr>
<tr>
<td>1003</td>
<td>ShoppingCartInsert</td>
</tr>
<tr>
<td>1004</td>
<td>ShoppingCartDelete</td>
</tr>
<tr>
<td>1005</td>
<td>OrderCancel</td>
</tr>
<tr>
<td>1006</td>
<td>itemView</td>
</tr>
<tr>
<td>1007</td>
<td>WishlistAdd</td>
</tr>
<tr>
<td>1008</td>
<td>WishlistDelete</td>
</tr>
<tr>
<td>1009</td>
<td>BrowseContent</td>
</tr>
<tr>
<td>1010</td>
<td>Search</td>
</tr>
<tr>
<td>1011</td>
<td>ShoppingCartUpdate</td>
</tr>
</tbody>
</table>

**Implementing action logging:**

To implement action logging, insert an Action bean into your JSP. To log additional application data associated with the action, add key/value pair information to the log method call.

To implement action logging, insert an Action bean into your JSP, for example:

```html
<jsp:useBean class="com.ibm.wcp.analysis.beans.Action" id="action" scope="session"/>

<% // Note: Both the resource id and collection name were added to the request // by the referral page. action.log( request, request.getParameter( "resourceId" ), request.getParameter( "collectionName" ), "ItemView" ); %>
```

To log additional application data associated with the action, add key/value pair information to the log method call in. For example:

```html
<jsp:useBean class="com.ibm.wcm.analysis.beans.Action" id="action" scope="session"/>

<jsp:useBean class="ShoppingCart" id="cart" scope="session"></jsp:useBean>

<% // Log last item added to shopping cart with quantity and size data. Hashtable actionInfo = new Hashtable(); Product[] cartItems = cart.getItems(); actionInfo.put( "quantity", new String( cartItems[cartItems.length - 1].getQuantity() ) ); actionInfo.put( "size", cartItems[cartItems.length - 1].getSize() ); action.log( request, cartItems[cartItems.length - 1].getId(), null, "Shopping Cart Insert", actionInfo ); %>
```

**Action beans reference:**

Chapter 5. Setting up a site 423
View some additional information related to Action beans and associated methods.

The `com.ibm.wcp.analysis.beans.Action` bean method signatures are:

<table>
<thead>
<tr>
<th>Action bean method signatures</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>public void log(HttpServletRequest request, String actionName)</code></td>
<td>Logs a non-resource specific action</td>
</tr>
<tr>
<td><code>public void log(HttpServletRequest request, String resourceId, String collectionName, String actionName)</code></td>
<td>Logs a resource specific action</td>
</tr>
<tr>
<td><code>public void log(HttpServletRequest request, String resourceId, String collectionName, String actionName, String key, String value)</code></td>
<td>Logs a resource specific action with key/value action datum</td>
</tr>
<tr>
<td><code>public void log(HttpServletRequest request, String resourceId, String collectionName, String actionName, Hashtable keyValueData)</code></td>
<td>Logs a resource specific action with multiple key/value action data. Each key can have a single value specified by a string object or multiple values specified by a string array</td>
</tr>
</tbody>
</table>

Action beans should be instantiated as session beans. They maintain user action information for the current session including the actions logged by resource and their corresponding log counts.

The following methods are accessible from rules. These methods are provided through a custom application object.

```java
public String[] getActionNames(HttpServletRequest request);
public int getActionCount(HttpServletRequest request, String actionName);
```

Actions can be logged with or without respect to a specific resource. For example, the "OrderCancel" action does not apply to a specific resource whereas the "BrowseContent" action applies to a specific content resource.

If the call to the action log method specifies a resourceId and the collectionName is null, the name of a ResourceCollection is inferred. The ResourceCollection used will be any one containing a resource with the specified resourceId. The determination of the ResourceCollection used in this scenario is non-deterministic.

Note that these variants are suitable for user implementations supporting one and only one ResourceCollection per resource class. If an implementation utilizes multiple ResourceCollections for the same resource class, the collectionName should be specified.

**Category beans:**

Content categories enable you to classify Web user interests. You use Category beans to log content categories.

As a Web site visitor navigates your Web pages, a Category bean logs the different categories viewed by the visitor during the session. For example, if a Web site visitor goes to your Web site and views information about the weather, the category "Weather" can be logged. Another Web site visitor might be more
interested in sports, so you would log the "Sports" category for this user. Category beans maintain category information for the current session including the categories logged and their corresponding log counts. Category information is routed to all registered log listeners.

All categories are logged to the Feedback schema and to the user session. Category information is not logged to the LikeMinds schema.

You can create rules that access the category counts in the current session, for example:

```
News Interest is
    Weather when current Category Count.Weather > 5
    Sports when current Category Count.Sports > 5
    Otherwise Headlines
```

You can also create rules that access the category names from the current session, for example:

```
Select content whose News.Topics is included in current CategoryNames
    Category Names order as is
```

**Implementing category logging:**

To implement category logging, insert a Category bean into your JSP.

Consider the following code for inserting a category bean in your JSP.

```java
<jsp:useBean class="com.ibm.wcp.analysis.beans.Category" id="category" scope="session"/>
<% category.log( request, "Sports" ); %>
```

You can also pass an object to a JSP and implicitly log the content category. For example:

```java
<jsp:useBean class="com.ibm.wcp.analysis.beans.Category" id="category" scope="session"/>
<%
    NewsManager newsManager = new NewsManager();
    News news = newsManager.findById( request.getParameter( "newsId" ), null );
    category.log( request, news );
%>
```

In the previous example, the category bean logs the news category by querying the LoggableResource interface to get the category.

**Category beans reference:**

Learn about the various method signatures of Category beans.

The `com.ibm.wcp.analysis.beans.Category` bean method signatures are:

```
<table>
<thead>
<tr>
<th>CustomLog bean method signatures</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>public void log( HttpServletRequest request, String category )</td>
<td>Logs a single category literal.</td>
</tr>
<tr>
<td>public void log( HttpServletRequest request, String[] categories )</td>
<td>Logs an array of category literals.</td>
</tr>
</tbody>
</table>
```
Table 127. Descriptions for CustomLog bean method signatures (continued)

<table>
<thead>
<tr>
<th>CustomLog bean method signatures</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>public void log(HttpServletRequest request, LoggableResource resource)</td>
<td>Logs categories of interest by querying the LoggableResource interface of the object.</td>
</tr>
</tbody>
</table>

The LoggableResource interface can be implemented in addition to the Resource interface in order to facilitate logging. The categories of your resources can then be stored and retrieved from a database. By using this interface, you avoid the specification of category literals in your JSPs.

```java
public interface LoggableResource
{
    String[] getTopics();
}
```

Each of the topics returned by `getTopics` is logged as a category. If `getTopics` returns null, no category will be logged.

The log methods create a CategoryEvent with the request and category. The CategoryEvent is routed to all of the registered log listeners.

Category beans should be instantiated as session beans. They maintain category information for the current session including the categories logged and their corresponding log counts.

The following methods are accessible through rules. They can also be accessed directly through the category bean.

```java
public String[] getCategoryNames(HttpServletRequest request)
public int getCategoryCount(HttpServletRequest request, String category)
```

Rating beans:

The Rating bean is a specialized logging bean. You use Rating beans in applications that solicit user ratings for web content or other personalization resources.

The ratings logged through the Rating bean provide the user data necessary for the LikeMinds preference engine. This engine uses explicitly specified user ratings to predict user preferences. You can generate reports based on the ratings of your website content or resources.

**Note:** The concept of ratings logged through the Rating bean is different from the concept of ratings generated from tagging and rating. Tagging and rating allows users to assign ratings directly to an object to indicate likes and dislikes and is not used to collect data for future recommendations.

All rating data is logged to the Feedback schema and to the LikeMinds schema. Rating data are not logged to the user session.

**Implementing rating logging:**

To implement rating logging, insert a Rating bean into your JSP.
Following is an example that demonstrates implementation of rating logging by inserting a Rating bean in a JSP.

```jsp
<jsp:useBean class="com.ibm.wcp.analysis.beans.Rating" id="pref" scope="session" />
<% // Note: The mediaId, collectionName, and rating were added to the request // by the referring page.
    pref.log( request,
        request.getParameter( "mediaId" ),
        request.getParameter( "collectionName" ),
        request.getParameter( "rating" ));
%>
```

**Note:** Web applications implementing preference logging must provide a user interface (UI) to enable preference setting. Once retrieved from the UI, the preference settings can be logged with the Rating bean.

*Rating beans reference:*

Learn about the various method signatures of Rating beans.

The `com.ibm.wcp.analysis.beans.Rating` bean method signatures are:

<table>
<thead>
<tr>
<th>Rating bean method signatures</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>public void log(HttpServletRequest request, Resource resource, int prefRating)</code></td>
<td>Logs a rating for a Resource object. A ResourceCollection with the same resource type will be determined and used for logging the collection name. If you have multiple ResourceCollections containing objects of the same class, you should use the log method that accepts the resourceId and collectionName.</td>
</tr>
<tr>
<td><code>public void log(HttpServletRequest request, String resourceId, String collectionName, int prefRating)</code></td>
<td>Logs a rating using a resource id and collection name.</td>
</tr>
<tr>
<td><code>public void log(HttpServletRequest request, String resourceId, String collectionName, int prefRating, Hashtable ratingData)</code></td>
<td>Logs a resource-specific rating with multiple key/value rating data. Each key can have a single value specified by a string object, or multiple values specified by a string array.</td>
</tr>
</tbody>
</table>

The log methods generate a RatingEvent object with the request and rating data. These events are routed to all of the registered log listeners.

Rating beans should be instantiated as session beans; however, they do not maintain rating information for the current session.

*CustomLog beans:*

CustomLog beans give you the flexibility to log data that does not easily map to the other logging beans. You use CustomLog beans to log application specific data. You define the required key name(s) as well as the value(s) for that key.

Logged data is stored in the Feedback schema in the HITPARMS, KEY_VALUE_COMBO, KEY_VALUE_PAIR, KEY, and VALUE tables. If you use CustomLog beans, you may want to create custom report elements.
All custom log data is logged to the Feedback schema. Custom log data is not logged to the LikeMinds schema nor to the user session.

**Implementing custom logging:**

To implement custom logging, insert a CustomLog bean into your JSP.

The following code provides an example:

```jsp
<jsp:useBean class="com.ibm.wcp.analysis.beans.CustomLog" id="custom" scope="session" />
<% custom.log( request, "version", "1.0" ); %>
```

You can also log multiple values. For example:

```jsp
<jsp:useBean class="com.ibm.wcp.analysis.beans.CustomLog" id="custom" scope="session" />
<%
    Hashtable customInfo = new Hashtable();
    customInfo.put( "version", "1.0" );
    customInfo.put( "custLevel", new String[] { "gold", "preferred" } );
    customInfo.put( "custRegion", "West" );
    custom.log( request, customInfo );
%>
```

**CustomLog beans reference:**

View the CustomLog bean method signatures.

The `com.ibm.wcp.analysis.beans.CustomLog` bean method signatures are:

<table>
<thead>
<tr>
<th>CustomLog bean method signatures</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>public void log( HttpServletRequest request, String key, String value );</code></td>
<td>Logs a single pair of custom key/value data.</td>
</tr>
<tr>
<td><code>public void log( HttpServletRequest request, Hashtable keyValueData );</code></td>
<td>Logs multiple pairs of custom key/value data.</td>
</tr>
</tbody>
</table>

The log methods generate a CustomLogEvent with the request and custom data. The CustomLogEvent is routed to all of the registered log listeners.

CustomLog beans should be instantiated as session beans; however, they do not maintain any session data.

**PageView beans:**

You use PageView beans to log HTTP request data for JSPs (JavaServer pages).

The HTTP request data logged by PageView beans provides basic Web usage data that can be used during Web traffic analysis. The Action, Category, CustomLog, and Rating beans automatically collect the same basic HTTP data that is collected by the PageView bean. HTTP request data is also collected whenever a rule is executed. Use the PageView bean if the JSP of interest has no other logging beans or content spots, and you want to include the JSP in usage analysis.

**Implementing PageView logging:**
To implement Web usage logging, insert a PageView bean in your JSP.

Following is an example of implementing PageView logging:

```jsp
<jsp:useBean class="com.ibm.wcp.analysis.beans.PageView" id="pageView" scope="session" />
<%
    pageView.log(request);
%>

PageView beans reference:


The com.ibm.wcp.analysis.beans.PageView bean log method signature is:

```java
public void log(HttpServletRequest request)
```

The log method creates a new PageViewEvent with the request. The PageViewEvent is routed to all registered log listeners.

PageView beans should be instantiated as session beans; however, they will not maintain any session data.

**LogManager**

When data is logged by either logging beans or rules, log events are generated and are routed to a controller for processing. LogManager is the class that implements this controller. There is a single instance of the LogManager within the Personalization run-time. It is responsible for receiving all logged events and distributing these events to listener objects that implement the LogListener interface and are registered with the LogManager.

The LogManager queues log events as they are received. In order to protect listeners from performance degradation due to inefficiencies and/or problems in other listeners, a separate notification thread is managed for each active listener. A nanny thread monitors the notification threads and suspends any non-responsive listener. Since the "queue" is processed by multiple consumers, an event is removed from the queue after all of the notification threads have read the event. You can alter the settings that govern the non-responsive time interval and event queue size by modifying the FeedbackService.properties file, located in the

```java
wp_profile_root/PortalServer/config/config/services
```

**Listeners and persistence**

Log listeners process the log events originating from either logging beans or rules.

Log listeners process the log events originating from either logging beans or rules and will typically store the data in some fashion (such as a database). Separate LogListener implementations are provided for processing log data into the Personalization Feedback schema and for processing data into the LikeMinds schema. These listeners always run whenever logging is enabled. Additionally, it is possible to extend the provided listeners and also write custom listeners from scratch to process logging events and log data as you see fit.

A log listener is registered with and receives log events from the LogManager. All log listeners must either implement the LogManager interface or extend the LogAdapter class. The LogAdapter class provides a default (empty) implementation of each of the handleEvent methods in the LogListener interface.
The logging component uses cache and multi-threading to optimize performance, and to allow each listener to process events at a different rate without affecting other listeners.

**The log listener interface**

```java
public interface com.ibm.wcp.analysis.event.LogListener extends java.util.EventListener {

  /**
   * Method called by LogManager after adding the listener to the
   * set of active listeners. This method can be used to perform
   * initialization during startup.
   */
  public void startHandlingEvents();

  /**
   * Method called by LogManager when the listener is removed
   * from the set of active listeners. This method can be
   * used to perform cleanup during termination processing.
   */
  public void stopHandlingEvents();

  /**
   * Method to handle rule trigger events.
   */
  void handleEvent( RuleEvent event );

  /**
   * Method to handle category logging events.
   */
  void handleEvent( CategoryEvent event );

  /**
   * Method to handle action logging events.
   */
  void handleEvent( ActionEvent event );

  /**
   * Method to handle explicit rating events.
   */
  void handleEvent( RatingEvent event );

  /**
   * Method to handle custom logging events.
   */
  void handleEvent( CustomLogEvent event );

  /**
   * Method to handle basic Web traffic events.
   */
  public void handleEvent( PageViewEvent event );

  /**
   * Method to handle Personalization audit events.
   */
  void handleEvent( AuditEvent event );
}
```

**FeedbackListener:**
The FeedbackListener class routes data to the Feedback schema.

The FeedbackListener processes all log event types. Each page request (JSP invocation) maps to one row in the Feedback schema HIT_FACTS table. Data for each log event is stored in a separate entry in the Feedback schema HITPARMS table. This distinguishes the data related to each rule or log method call and enables the reporting of information specific to individual content spots or method call. This storage design correlates of all of the data logged by the rules or method calls in a JSP enabling page level reporting.

An instance of the FeedbackListener is active on all run-time servers whenever logging is enabled.

Related concepts:
“Feedback database schema” on page 442

The Feedback schema, within the Feedback database, stores data about your Web site visitors’ actions. This schema is closely aligned with the Tivoli Site Analyzer schema.

LMListener:

The LMListener routes data of interest to the LikeMinds database layer.

The LMListener only processes ActionEvent and RatingEvent data. These events result in LikeMinds addTransaction calls. Only action names defined in the LikeMinds lpx_trx_type table are logged by this listener. All other actions are ignored.

An instance of the LMListener is active after LikeMinds configuration tasks are run.

Custom log listeners:

When the log method of a logging bean is called, the feedback facility generates a log event. A log event is also generated when a rule is executed. Log listeners process these log events and either store the event data or perform custom processing with these events.

The Feedback component provides two default log listeners, the LMListener that collects data for use by the LikeMinds recommendation engine and the FeedbackListener that collects data for use in the feedback reports. Custom log listeners can be used to modify the default behavior of the FeedbackListener or to implement a listener that provides user-specific behavior.
There are a number of reasons that you might want to provide a customized feedback listener. Some of these are:

- To collect request parameters, referral parameters, or cookies. By default, the feedback listener does not collect this data. You must implement a customized feedback listener that enables the collection of this data.
- To prevent private information from being stored. For instance, you could mask the userid or IP address to ensure the privacy of your users.
- To augment the event data. Suppose your event data contains a product id number and you would like to report on products by product name and id. You could perform a custom lookup during event processing and add the product name to the event.

There are a number of reasons that you might want to provide a new custom listener class. Some of these are:

- To store event data in a separate database. Suppose that you want to capture rating data in a preference database. You can do this through your own web application or you could do this as by adding a custom listener and capturing rating events. The custom listener could facilitate real time rating results.
- To generate email when an event occurs. Perhaps you want to send email to customers after they purchase a large order. By processing an action event with the purchase amount included as action data, you could do this with a custom log listener.
- To generate notifications when an event occurs. You can detect the frequency of shopping cart abandons and generate a notification to the site administrator to check site availability and performance.

Custom listener classes:
A custom listener class is a class that implements the LogListener interface. View the steps to implement a custom listener class.

The custom listener class can implement the LogListener interface explicitly and provide implementations for all of the LogListener methods. Alternatively, the custom listener class can extend the LogAdaptor class. Since the LogAdaptor class contains default implementations of all of the LogListener methods, LogAdaptor subclasses only need to implement the LogListener methods of interest. To implement a custom listener class:

1. Implement a class that extends com.ibm.wcp.analysis.event.LogAdaptor. Override the handleEvent methods that accept the event type of interest. You can also provide an implementation for startHandlingEvents and stopHandlingEvents if your listener needs to perform initialization or cleanup, respectively.

2. Install the class file for your custom listener in the classpath of the server where the Personalization run-time is installed.

3. Add the class name to the logListeners property in the FeedbackService.properties file, located in the \wp_profile_root\PortalServer/config/config/services directory.

4. Restart the Personalization run-time server.

Note that custom listener classes (other than customized feedback listeners) are always enabled when the Personalization run-time enterprise application is running.

The following example illustrates the implementation of a custom listener class. The listener in this example will generate an alert whenever the "MegaPurchase" action is logged.

```java
import com.ibm.wcp.analysis.event.*;

public class SimpleCustomListener extends LogAdapter
{
    /**
     * Method to handle action events.
     */
    public void handleEvent( ActionEvent event )
    {
        if (event.getActionName().equals( "MegaPurchase" ))
            generateAlert();
    }

    /**
     * Method to generate an alert.
     */
    private void generateAlert()
    {
        // Your custom code for generating an alert can go here.
        System.out.println( "We have a big purchase!" );
    }
}
```

**Customized feedback listeners:**

A customized feedback listener is a subclassed FeedbackListener object that is registered with the feedback LogManager. View the steps to implement a custom feedback listener.

Perform the following steps to implement a custom feedback listener:
1. Implement a class that extends com.ibm.wcp.analysis.event.FeedbackListener. Override the handleEvent methods that accept the event type of interest.

2. Install the class file for your custom feedback listener in the classpath of the server where the Personalization run-time is installed.

3. Add the class name to the logListeners property in the FeedbackService.properties file, located in the \wp_profile_root\PortalServer/config/config/services directory.

4. Ensure that the LoggingEnabled is set to true.

5. Restart the Personalization run-time server.

Note that whenever logging is enabled, your customized FeedbackListener is enabled. Conversely, your customized FeedbackListener is disabled whenever logging is disabled. Only the default FeedbackListener or one custom feedback listener can be active at a time. The specification of a custom feedback listener will override the registration of the default FeedbackListener with the LogManager.

The following example illustrates the implementation of a customized feedback listener. The listener in this example will collect cookie data, request query data, and referral query data. Note that this data is not collected by the default FeedbackListener, but can be set in the event data with a customized feedback listener.

```java
import com.ibm.wcp.analysis.event.*;

public class ParmCookieListener extends FeedbackListener {
    /**
     * Method to handle action events.
     */
    public void handleEvent( ActionEvent event )
    {
        enableParmCollection( event );
        super.handleEvent( event );
    }

    /**
     * Method to handle category events.
     */
    public void handleEvent( CategoryEvent event )
    {
        enableParmCollection( event );
        super.handleEvent( event );
    }

    /**
     * Method to handle rule events.
     */
    public void handleEvent( RuleEvent event )
    {
        enableParmCollection( event );
        super.handleEvent( event );
    }

    /**
     * Method to set switches to collect cookie and query string data.
     */
    private void enableParmCollection( LogEvent event )
    {
        event.enableLogCookies();
    }
}
```
Classes and APIs for writing custom listeners

In order to write custom listeners, it is necessary to familiarize yourself with the classes representing logging events and info that is contained in those events. A reference of these classes is provided here.

**LogEvent class:**

The LogEvent class is the base class for all run-time logging events. The methods in this class are used to access all basic http request information.

All run-time log data is accessible through the log event sub classes. These log events are constructed by either a rule trigger or a logging bean. They are then routed to the registered log listeners.

The direct subclasses of the LogEvent class are ActionEvent, CategoryEvent, CustomLogEvent, PageViewEvent, RatingEvent, and RuleEvent.

### Table 130. Method summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>protected LogEvent( HttpServletRequest request )</td>
<td>Constructor</td>
</tr>
<tr>
<td>public Cookie[] get Cookies( )</td>
<td>Returns the cookies available with the request for this event.</td>
</tr>
<tr>
<td>public void set Cookies( Cookie[] )</td>
<td>Sets the cookies for this event. Can be used by custom listeners in order to replace the cookie data received with the current JSP request.</td>
</tr>
<tr>
<td>public void enableLogCookies( boolean enable )</td>
<td>If enable is true, enables the collection of cookies for this event instance; otherwise, disables the collection of cookies for this event instance. If you want to collect cookie information with the FeedbackListener or a subclassed FeedbackListener, use this method.</td>
</tr>
<tr>
<td>public void enableLogCookies( )</td>
<td>Same as enableLogCookies( true ).</td>
</tr>
<tr>
<td>public boolean log Cookies( )</td>
<td>Returns true if cookie information should be logged to the Feedback schema for this event; otherwise, returns false.</td>
</tr>
<tr>
<td>public String get IPAddress( )</td>
<td>Returns the Web client IP address.</td>
</tr>
<tr>
<td>public void set IPAddress( String ipAddress )</td>
<td>Sets the IP address for this event. Can be used by custom listeners in order to replace the IP address for the current JSP request.</td>
</tr>
<tr>
<td>public String getMethod( )</td>
<td>Returns the HTTP method used in the current JSP request (GET, POST). Note that the method is not stored in the Feedback schema.</td>
</tr>
<tr>
<td>public void setMethod( String method )</td>
<td>Sets the HTTP method for the current JSP request. Can be used by custom listeners in order to replace the method from the current JSP request. Note that the method is not stored in the Feedback schema.</td>
</tr>
</tbody>
</table>
Table 130. Method summary (continued)

<table>
<thead>
<tr>
<th>Method</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>public String getProtocol( )</td>
<td>Returns the protocol for the current JSP request (http or https).</td>
</tr>
<tr>
<td>public void setProtocol( String protocol )</td>
<td>Sets the protocol for the current JSP request. Can be used by custom listeners in order to replace the protocol from the current JSP request.</td>
</tr>
<tr>
<td>public void enableLogReferralParms( boolean enable )</td>
<td>If enable is true, enables the collection of referral parameters for this event instance; otherwise, disables the collection of referral parameters for this event instance. If you want to collect the referral query string parameters with the FeedbackListener or a subclassed FeedbackListener, use this method.</td>
</tr>
<tr>
<td>public void enableLogReferralParms( )</td>
<td>Same as enableLogReferralParms( true ).</td>
</tr>
<tr>
<td>public boolean logReferralParms( )</td>
<td>Returns true if referral parameter information should be logged to the Feedback schema for this event; otherwise, return false.</td>
</tr>
<tr>
<td>public String getRemoteHost( )</td>
<td>Returns the host name of the client machine that issued this JSP request. Note that the remote host name is not stored in the Feedback schema.</td>
</tr>
<tr>
<td>public void setRemoteHost( String hostName )</td>
<td>Sets the host name of the client machine that issued this JSP request. Can be used by custom listeners that perform DNS resolution. Note that the remote host name is not stored in the Feedback schema.</td>
</tr>
<tr>
<td>public String getQueryString( )</td>
<td>Returns the query string parameters for this event.</td>
</tr>
<tr>
<td>public void setQueryString( String queryString )</td>
<td>Sets the query string parameters for this event. Can be used by custom listeners in order to replace the query string parameters for the current JSP request.</td>
</tr>
<tr>
<td>public void enableLogQueryParms( boolean enable )</td>
<td>If enable is true, enables the collection of query parameters for this event instance; otherwise, disables the collection of referral parameters for this event instance. If you want to collect the query string parameters with the FeedbackListener or a subclassed FeedbackListener, use this method.</td>
</tr>
<tr>
<td>public void enableLogQueryParms( )</td>
<td>Same as enableLogQueryParms( true ).</td>
</tr>
<tr>
<td>public boolean logQueryParms( )</td>
<td>Returns true if query parameter information should be logged to the Feedback schema for this event; otherwise, return false.</td>
</tr>
<tr>
<td>public String getReferrer( )</td>
<td>Returns the URL of the referral page or null if there was no referrer.</td>
</tr>
<tr>
<td>public void setReferrer( String referrer )</td>
<td>Sets the referral page for this event. Can be used by custom listeners in order to replace the referrer for the current JSP request.</td>
</tr>
<tr>
<td>public String getServerName( )</td>
<td>Returns the host name of the server machine that is processing this JSP request. Note that the server host name is not stored in the Feedback schema.</td>
</tr>
<tr>
<td>public void setServerName( String serverName )</td>
<td>Sets the host name of the server machine that is processing this JSP request. Can be used by custom listeners to set the name of the current server. Note that the server host name is not stored in the Feedback schema.</td>
</tr>
<tr>
<td>Method</td>
<td>Explanation</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>public String getSessionId( )</td>
<td>Returns the id of the session for the current JSP request.</td>
</tr>
<tr>
<td>public void setId(String sessionId)</td>
<td>Sets the id of the current user session. Can be used by custom listeners in order to replace the id of the current HttpSession object. This can be useful when an alternative session identification mechanism is used.</td>
</tr>
<tr>
<td>public Date getTimestamp( )</td>
<td>Returns the time that this log event was generated. Note that this timestamp differs slightly from the time the event was received by the IBM WebSphere Application Server. Note that if there are multiple logging beans or content spots in a JSP, the timestamps in the generated log events will be the same.</td>
</tr>
<tr>
<td>public void setTimestamp(Date timestamp)</td>
<td>Sets the time for this log event. Can be used by custom listeners in order to replace the timestamp used for this event.</td>
</tr>
<tr>
<td>public String getUrl( )</td>
<td>Returns the URL of the page request encapsulated by this log event.</td>
</tr>
<tr>
<td>public void setUrl(String url)</td>
<td>Sets the URL of the page request for this event. Can be used by custom listeners in order to replace the URL for the current JSP request.</td>
</tr>
<tr>
<td>public String getUser( )</td>
<td>Returns the HTTP authenticated user for the current session.</td>
</tr>
<tr>
<td>public void setUser(String user)</td>
<td>Sets the user for this event. Can be used by custom listeners in order to replace the user for the current JSP request. This can be useful when an alternative authentication mechanism is used.</td>
</tr>
<tr>
<td>public String getUserAgent( )</td>
<td>Returns the browser engine for the current session.</td>
</tr>
<tr>
<td>public void setUserAgent(String userAgent)</td>
<td>Sets the user agent for this event. Can be used by custom listeners in order to replace the user agent from the current JSP request.</td>
</tr>
<tr>
<td>public String toString( )</td>
<td>Returns a String representation of this event.</td>
</tr>
</tbody>
</table>

**RuleEvent class:**

A RuleEvent class is constructed whenever a rule is executed. It contains information about the rule that was executed and the resulting resources. It is an implicitly constructed event; a logging bean is not necessary.

```java
```

Get an overview of the methods of the RuleEvent class.

**Table 131. Method summary**

<table>
<thead>
<tr>
<th>Method</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>public RuleEvent(HttpServletRequest request, RuleInfo ruleInfo, ResourceInfo[] resourceInfo)</td>
<td>Constructor.</td>
</tr>
<tr>
<td>public RuleInfo getRuleInfo( )</td>
<td>Returns a rule information object containing the campaign and rule name for the rule that was executed in the content spot.</td>
</tr>
</tbody>
</table>
Table 131. Method summary (continued)

<table>
<thead>
<tr>
<th>Method</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>public void setRuleInfo( RuleInfo ruleInfo )</td>
<td>Sets the rule information for this event. Can be used by custom listeners in order to replace the rule execution data.</td>
</tr>
<tr>
<td>public ResourceInfo[] getResourceInfo( )</td>
<td>Returns a resource information array containing the results of the rule for this event. The resource information contains the collection name and resource ids of the results.</td>
</tr>
<tr>
<td>public void setResourceInfo( ResourceInfo[] resourceInfo )</td>
<td>Sets the resource information for this event. Can be used by custom listeners in order to replace the resource information.</td>
</tr>
<tr>
<td>public String getResourceClass( )</td>
<td>Returns the class name of the resources returned by the executed rule.</td>
</tr>
<tr>
<td>public void setResourceClass( String className )</td>
<td>Sets the class name of the resources returned by the executed rule. Can be used by custom listeners in order to replace the class name.</td>
</tr>
<tr>
<td>public String toString( )</td>
<td>Returns a String representation of this event.</td>
</tr>
</tbody>
</table>

CategoryEvent class:

The CategoryEvent class is used to access the data logged with a Category bean.

```java
    implements Serializable
```

Get an overview of the methods of the CategoryEvent class.

Table 132. Method summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>public CategoryEvent( HttpServletRequest request, String[] topics )</td>
<td>Constructor.</td>
</tr>
<tr>
<td>public String[] getTopics( )</td>
<td>Returns the array of topics for this category event.</td>
</tr>
<tr>
<td>public void setTopics( String[] topics )</td>
<td>Sets the topics for this event. Can be used by custom listeners in order to replace the topics for this event.</td>
</tr>
<tr>
<td>String toString( )</td>
<td>Returns a String representation of this event.</td>
</tr>
</tbody>
</table>

ActionEvent class:

The ActionEvent class is used to access the data logged with an Action bean.

```java
    implements Serializable
```

Get an overview of the methods of the ActionEvent class.
**Table 133. Method summary**

<table>
<thead>
<tr>
<th>Method</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>public ActionEvent( HttpServletRequest request, ResourceInfo resourceInfo, String actionName, Hashtable actionData )</td>
<td>Constructor.</td>
</tr>
<tr>
<td>public ResourceInfo getResourceInfo( )</td>
<td>Returns the resource upon which the action in this event was taken.</td>
</tr>
<tr>
<td>public void setResourceInfo( ResourceInfo resourceInfo )</td>
<td>Sets the resource info for this action event. Can be used by custom listeners to replace the resource information in this event.</td>
</tr>
<tr>
<td>public HasTablet getActionData( )</td>
<td>Returns the supplemental data associated with this action. The action data is in key value format. Since a single key can have multiple values, the action data values are stored in the hashtable as String arrays.</td>
</tr>
<tr>
<td>public void setActionData( HasTablet actionData )</td>
<td>Sets the action data for this action event. Can be used by custom listeners to replace the action data in this event.</td>
</tr>
<tr>
<td>public String getActionName( )</td>
<td>Returns the name given to the current action.</td>
</tr>
<tr>
<td>public void setActionName( String actionName )</td>
<td>Sets the action name for this event. Can be used by custom listeners to replace the action name in this event.</td>
</tr>
<tr>
<td>public RuleInfo getRuleInfo( )</td>
<td>Returns the rule associated with this action. The subject rules for this action are determined by all rules in the current session that returned the target resource.</td>
</tr>
<tr>
<td>public void setRuleInfo( RuleInfo ruleInfo )</td>
<td>Sets the rule information for this action event. Can be used by custom listeners to replace the rule information in this event.</td>
</tr>
<tr>
<td>public String toString( )</td>
<td>Returns a String representation of this event.</td>
</tr>
</tbody>
</table>

**CustomLogEvent class:**

Get an overview of the CustomLogEvent class and its methods.

The CustomLogEvent class is used to access the data logged with a CustomLog bean.

```java
```

**Table 134. Method summary**

<table>
<thead>
<tr>
<th>Method</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>public CustomLogEvent( HttpServletRequest request, HasTablet customData )</td>
<td>Constructor.</td>
</tr>
<tr>
<td>public HasTablet getCustomData( )</td>
<td>Returns the custom data as key value information. Since a single custom data key can have multiple values, the custom data values are stored in the hashtable as String arrays.</td>
</tr>
</tbody>
</table>
Table 134. Method summary (continued)

<table>
<thead>
<tr>
<th>Method</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>public void setCustomData( Hashtable customData )</td>
<td>Sets the custom data for this event. Can be used by custom listeners to replace the data in this custom data event.</td>
</tr>
<tr>
<td>public String toString( )</td>
<td>Returns a String representation of this event.</td>
</tr>
</tbody>
</table>

*RatingEvent class:*

The RatingEvent class is used to access the data logged with a Rating bean.

```java
    implements Serializable
```

Get an overview of the methods of the RatingEvent class.

Table 135. Method summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>public RatingEvent( HttpServletRequest request,</td>
<td>Constructor.</td>
</tr>
<tr>
<td>ResourceInfo int resourceInfo, rating )</td>
<td>Returns the resource that is the target of the current rating. The resource object returned contains the collection name and resource id.</td>
</tr>
<tr>
<td>public ResourceInfo getResourceInfo( )</td>
<td>Sets the resource information for this rating event. Can be used by custom listeners to replace the resource information for this rating.</td>
</tr>
<tr>
<td>public void setResourceInfo( ResourceInfo resourceInfo )</td>
<td>Returns the rating as an integer value. The value can be any valid integer as defined by the Web application implementor.</td>
</tr>
<tr>
<td>public int getRating( )</td>
<td>Sets the rating for this event. Can be used by custom listeners to replace the rating in this event.</td>
</tr>
<tr>
<td>public String toString( )</td>
<td>Returns a String representation of this event.</td>
</tr>
</tbody>
</table>

*PageViewEvent class:*

The PageViewEvent class is used to access data logged by a PageView bean.

```java
    implements Serializable
```

Get an overview of the methods of the PageViewEvent class.

Table 136. Method summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>public PageViewEvent( HttpServletRequest request )</td>
<td>Constructor.</td>
</tr>
<tr>
<td>public String toString( )</td>
<td>Returns a String representation of this event.</td>
</tr>
</tbody>
</table>

*ResourceInfo class:*
The ResourceInfo class is a wrapper class for a resource id and collection name tuple.

```java
public class com.ibm.wcp.analysis.event.ResourceInfo extends Object
    implements Serializable
```

Get an overview of the methods of the ResourceInfo class.

**Table 137. Method summary**

<table>
<thead>
<tr>
<th>Method</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>public ResourceInfo( String resourceId,</td>
<td>Constructor.</td>
</tr>
<tr>
<td>String collectionName )</td>
<td></td>
</tr>
<tr>
<td>public ResourceInfo( )</td>
<td>Constructor.</td>
</tr>
<tr>
<td>public String getResourceId( )</td>
<td>Returns the id of the resource.</td>
</tr>
<tr>
<td>public void setResourceId( String resourceId )</td>
<td>Sets the id of the resource.</td>
</tr>
<tr>
<td>public String getCollectionName( )</td>
<td>Returns the name of the collection that contains this resource.</td>
</tr>
<tr>
<td>public void setCollectionName( String collectionName )</td>
<td>Sets the name of the collection that contains this resource.</td>
</tr>
<tr>
<td>public String toString( )</td>
<td>Returns a String representation of this resource tuple.</td>
</tr>
<tr>
<td>public boolean equals( ResourceInfo resourceInfo )</td>
<td>Returns true if and only if (1) the objects are the same or (2) the resource id and collection name information is equal.</td>
</tr>
<tr>
<td>public int hashCode( )</td>
<td>Returns a hash code enabling objects of this type to be used as hash keys.</td>
</tr>
</tbody>
</table>

**RuleInfo class:**

The RuleInfo class is a wrapper class for a rule name and campaign name tuple.

```java
public class com.ibm.wcp.analysis.event.RuleInfo extends Object
    implements Serializable
```

Get an overview of the methods of the RuleInfo class.

**Table 138. Method summary**

<table>
<thead>
<tr>
<th>Method</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>public RuleInfo( String rule,</td>
<td>Constructor.</td>
</tr>
<tr>
<td>String collectionName )</td>
<td></td>
</tr>
<tr>
<td>public RuleInfo( )</td>
<td>Constructor.</td>
</tr>
<tr>
<td>public String getRule( )</td>
<td>Returns the name of the rule.</td>
</tr>
<tr>
<td>public void setRule( String rule )</td>
<td>Sets the name of the rule.</td>
</tr>
<tr>
<td>public String getCampaign( )</td>
<td>Returns the name of the campaign containing this rule.</td>
</tr>
<tr>
<td>public void setCampaign( String campaign )</td>
<td>Sets the name of the campaign containing this rule.</td>
</tr>
<tr>
<td>public String toString( )</td>
<td>Returns a String representation of this rule tuple.</td>
</tr>
</tbody>
</table>
### Table 138. Method summary (continued)

<table>
<thead>
<tr>
<th>Method</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>public boolean equals( RuleInfo ruleInfo )</td>
<td>Returns true if and only if (1) the objects are the same or (2) if the rule names are equal and the campaign names are equal.</td>
</tr>
</tbody>
</table>

### Reports

Once data has been logged to the Feedback database schema, it is possible to use any reporting tool that can connect to a standard SQL database to generate reports based on this data. Reference the Feedback database schema section of the Information Center when writing such reports.

**Related concepts:**

"Feedback database schema"

The Feedback schema, within the Feedback database, stores data about your Web site visitors' actions. This schema is closely aligned with the Tivoli Site Analyzer schema.

**Feedback database schema**

The Feedback schema, within the Feedback database, stores data about your Web site visitors' actions. This schema is closely aligned with the Tivoli Site Analyzer schema.

Any application that can connect to a standard SQL database can make use of the data that has been logged to the Feedback database schema.

For instance, you can use the data in the Feedback database schema to:

- Generate reports using reporting software.
- Integrate OnLine Analytical Processing (OLAP) tools.
- Create your own data integration tools.

**Feedback schema diagram:**

View a diagram that depicts the Feedback schema for the Personalization database.
**Feedback schema tables:**

Learn about the feedback schema tables and the fields or columns they contain.

The following tables are used by the Feedback schema:

**Table Name: Browsers**

*Table description:* This table contains the client Web browser defined internally by a User Agent mapping.

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Column description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Browser_ID</td>
<td>The primary key. A unique number automatically generated by the database.</td>
</tr>
</tbody>
</table>
Table 139. **Table description:** This table contains the client Web browser defined internally by a User Agent mapping. (continued)

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Column description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>The browser name defined internally by a User Agent mapping.</td>
</tr>
</tbody>
</table>

**Table Name: Calendar**

Table 140. **Table description:** Initially, this table is empty. The Feedback component calculates date stamp IDs, assuming a start date of 1/1/1995. The table can be populated with a new configuration task for applications that use custom logging or reports that reference this table.

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Column description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Date_ID</td>
<td>The primary key. A unique identifier automatically generated by the database.</td>
</tr>
<tr>
<td>Month</td>
<td>Month expressed as integer 1-12</td>
</tr>
<tr>
<td>Day</td>
<td>Day expressed as integer 1-31</td>
</tr>
<tr>
<td>Year</td>
<td>Year expressed as integer 1995-2030</td>
</tr>
<tr>
<td>Quarter</td>
<td>Quarter expressed as integer 1-4</td>
</tr>
<tr>
<td>Weekday</td>
<td>Weekday expressed as integer 1-7</td>
</tr>
<tr>
<td>WeekOfYear</td>
<td>The week number expressed as an integer 1-52</td>
</tr>
<tr>
<td>EpochDay</td>
<td>The number of days since the beginning date in this table.</td>
</tr>
<tr>
<td>EpochWeek</td>
<td>The number of weeks since the beginning date in this table.</td>
</tr>
<tr>
<td>EpochMonth</td>
<td>The number of months since the beginning date in this table.</td>
</tr>
<tr>
<td>EpochQuarter</td>
<td>The number of quarters since the beginning date in this table.</td>
</tr>
<tr>
<td>AggrWeekID</td>
<td>The identifier of the first day of the week that this day belongs to.</td>
</tr>
<tr>
<td>AggrMonthID</td>
<td>The identifier of the first day of the month that this day belongs to.</td>
</tr>
<tr>
<td>AggrQuarterID</td>
<td>The identifier of the first day of the quarter that this day belongs to.</td>
</tr>
<tr>
<td>AggrYearID</td>
<td>The identifier of the first day of the year that this day belongs to.</td>
</tr>
</tbody>
</table>

**Table Name: Domain**

Table description: This table is RESERVED for future Feedback component use.

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Column Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Domain_ID</td>
<td>The primary key; a unique identifier automatically generated by the database.</td>
</tr>
<tr>
<td>Name</td>
<td>The domain name as resolved from the TCP/IP address.</td>
</tr>
</tbody>
</table>
Table Name: Entities

Table description: This table is RESERVED for Feedback component use.

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Column Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entity_ID</td>
<td>The primary key; a unique identifier automatically generated by the database.</td>
</tr>
<tr>
<td>Name</td>
<td>The name of the entity.</td>
</tr>
<tr>
<td>Type</td>
<td>Indicates the type of entity: 0=entity 1=category 2=key/value pair</td>
</tr>
<tr>
<td>ParentEntity_ID</td>
<td>For a category entity, indicates the entity the category is based on.</td>
</tr>
</tbody>
</table>

Table Name: EntityTraversal

Table description: This table is RESERVED for IBM Feedback component use.

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Column Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EntityTraversal_ID</td>
<td>The primary key. A unique identifier automatically generated by the database.</td>
</tr>
<tr>
<td>Entity_ID</td>
<td>The identifier of the entity we start from.</td>
</tr>
<tr>
<td>Table1</td>
<td>The table this entity is stored in.</td>
</tr>
<tr>
<td>Column1</td>
<td>ID column for this entity.</td>
</tr>
<tr>
<td>Table2</td>
<td>The table of the parent entity.</td>
</tr>
<tr>
<td>Column2</td>
<td>Join column between entity and parent entity table.</td>
</tr>
<tr>
<td>PickThis</td>
<td>RESERVED for IBM Feedback component use.</td>
</tr>
</tbody>
</table>

Table Name: Hit_Facts

Table description: This table contains the information processed during data collection. Each row represents a hit to a page of your Web site--if that page contains a rule or logging bean. Not all fields are used on each entry.

Note: The first date stamp (date ID=1) corresponds to 1/1/1995. Time stamp IDs are calculated such that each second of the day has a separate ID.

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Column Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JS_ID</td>
<td>A foreign key to the JavaScriptStatus table.</td>
</tr>
<tr>
<td>CookiesStatus_ID</td>
<td>A foreign key to the CookiesStatus table.</td>
</tr>
<tr>
<td>Hits</td>
<td>The actual number of hits.</td>
</tr>
<tr>
<td>Status_ID</td>
<td>A foreign key to the ResetStatus table.</td>
</tr>
<tr>
<td>ReturnCode_ID</td>
<td>A foreign key to the ReturnCodes table.</td>
</tr>
<tr>
<td>HTTPVersion_ID</td>
<td>A foreign key to the HTTPVersion table.</td>
</tr>
<tr>
<td>RecordType</td>
<td>Indicates what type of log file this hit came from. This field is for Feedback component use only.</td>
</tr>
<tr>
<td>Column Name</td>
<td>Column Description</td>
</tr>
<tr>
<td>-----------------------</td>
<td>------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>LastUpdated</td>
<td>Time stamp of when this row was last created / updated.</td>
</tr>
<tr>
<td>PageViews</td>
<td>The actual number of page views. If the resource for this hit is a page, this field is set to 1.</td>
</tr>
<tr>
<td>TimeTaken</td>
<td>Time taken by the Web server to process this hit. This column is not implemented and is RESERVED for future Feedback component use.</td>
</tr>
<tr>
<td>Bytes</td>
<td>The number of bytes sent by the Web server to the client browser for this hit.</td>
</tr>
<tr>
<td>CorrelationKey</td>
<td>The key used to correlate information between the various types of logs that can be processed during data collection.</td>
</tr>
<tr>
<td>Session_ID</td>
<td>The identifier of the session this hit belongs to. A foreign key to the Session_Facts table.</td>
</tr>
<tr>
<td>HitTimestmp</td>
<td>Time stamp of this hit expressed as a Java timeInMillis value.</td>
</tr>
<tr>
<td>LocalDate_ID</td>
<td>The date of this hit. The first date stamp (date ID=1) corresponds to 1/1/1995. See also, Populating CALENDAR and TIMEOFDAY tables.</td>
</tr>
<tr>
<td>ImportHistory_ID</td>
<td>Not implemented. This field is RESERVED for future Feedback component use.</td>
</tr>
<tr>
<td>Hit_ID</td>
<td>The primary key. A unique identifier automatically generated by the database.</td>
</tr>
<tr>
<td>RefProtocol_ID</td>
<td>Protocol that the referrer used to request this hit. A foreign key to the Protocols table.</td>
</tr>
<tr>
<td>Referrer_ID</td>
<td>The referrer for this hit. A foreign key to the Referrer table.</td>
</tr>
<tr>
<td>Protocol_ID</td>
<td>The protocol the Web server used to process this hit. A foreign key to the Protocols table.</td>
</tr>
<tr>
<td>LocalTimeOfDay_ID</td>
<td>The time of this hit. See also, Populating CALENDAR and TIMEOFDAY tables.</td>
</tr>
<tr>
<td>GMTDate_ID</td>
<td>The GMT date of this hit. See also, Populating CALENDAR and TIMEOFDAY tables.</td>
</tr>
<tr>
<td>Resource_ID</td>
<td>The resource requested by the client browser. A foreign key to the Resources table.</td>
</tr>
<tr>
<td>GMTTimeOfDay_ID</td>
<td>The GMT date of this hit. See also, Populating CALENDAR and TIMEOFDAY tables.</td>
</tr>
<tr>
<td>Resource_ID</td>
<td>The resource requested by the client browser. A foreign key to the Resources table.</td>
</tr>
<tr>
<td>GMTTimeOfDay_ID</td>
<td>The GMT time of this hit. See also, Populating CALENDAR and TIMEOFDAY tables.</td>
</tr>
</tbody>
</table>
Table Name: HitParms

**Table description:** This table lists the personalization data (such as rules, campaigns, ratings), as well as query strings associated with a given hit.

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Column Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hit_ID</td>
<td>The identifier of the Hit that contains the query string. This is a foreign key to the ID column of the Hit_facts table.</td>
</tr>
<tr>
<td>Parms_ID</td>
<td>The identifier for a specific query string. This is a foreign key to the ID column of theParms table.</td>
</tr>
<tr>
<td>Ordering</td>
<td>Ordering is used to group a set of keys during the processing of a Web page.</td>
</tr>
<tr>
<td>ParmType</td>
<td>Identifies the type of key value data. For example, personalization rules data, query strings, or referral query status.</td>
</tr>
</tbody>
</table>

Table Name: HTTPVersion

**Table description:** This table lists all known HTTP versions. It is pre-populated by the Feedback component when the table is created.

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Column Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTTPVersion_ID</td>
<td>1 - 'HTTP 1.0' 2 - 'HTTP 1.1' 99 - 'Unknown'</td>
</tr>
<tr>
<td>Name</td>
<td>The name of the HTTP version.</td>
</tr>
</tbody>
</table>

Table Name: ImportHistory

**Table description:** This table is RESERVED for future Feedback component use

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Column Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ImportHistory_ID</td>
<td>The primary key. A unique number automatically generated by the database.</td>
</tr>
<tr>
<td>BeginTimeStamp</td>
<td>Start time of the import, expressed as a JAVA timeInMillis value.</td>
</tr>
<tr>
<td>EndTimeStamp</td>
<td>Time the import finished, expressed as a JAVA timeInMillis value.</td>
</tr>
<tr>
<td>Status</td>
<td>The status of the import.</td>
</tr>
</tbody>
</table>

Table Name: Key

**Table description:** Lists all Keys (part of Key/Value pairs) encountered during data collection.

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Column Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key_ID</td>
<td>The primary key. A unique identifier automatically generated by the database.</td>
</tr>
<tr>
<td>Name</td>
<td>The name of the key.</td>
</tr>
</tbody>
</table>
### Table Name: Key_Value_combo

**Table description:** This table lists all Key/Value pairs that are grouped together.

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Column Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parms_ID</td>
<td>A foreign key to the Parms table.</td>
</tr>
<tr>
<td>KeyValuePair_ID</td>
<td>A foreign key to the Key_Value_Pair table.</td>
</tr>
</tbody>
</table>

### Table Name: Key_Value_Pair

**Table description:** This table lists all Key/Value pairs processed during data collection.

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Column Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>KeyValuePair_ID</td>
<td>The primary key. A unique identifier generated automatically by the database.</td>
</tr>
<tr>
<td>Key_ID</td>
<td>A foreign key to the Key table.</td>
</tr>
<tr>
<td>Value_ID</td>
<td>A foreign key to the Value table.</td>
</tr>
</tbody>
</table>

### Table Name: Networks

**Table description:** This table contains all the TCP/IP addresses processed during data collection.

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Column Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Network_ID</td>
<td>The primary key; a unique number automatically generated by the database.</td>
</tr>
<tr>
<td>IP_Address</td>
<td>TCP/IP address as obtained from the page hit.</td>
</tr>
<tr>
<td>Subdomain_ID</td>
<td>The TCP/IP subdomain this address belongs to. A foreign key to the ID column of the Subdomains table.</td>
</tr>
<tr>
<td>Domain_ID</td>
<td>The TCP/IP domain this address belongs to. A foreign key to the ID column of the Domain table.</td>
</tr>
</tbody>
</table>

### Table Name: Parms

**Table description:** This table lists all the key/value pairs strings processed during data collection.

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Column Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>KVCount</td>
<td>Indicates how many key/value pairs are in this query string.</td>
</tr>
<tr>
<td>ParmsString</td>
<td>This field is not used.</td>
</tr>
<tr>
<td>WebNode_ID</td>
<td>This field is not used.</td>
</tr>
<tr>
<td>Parms_ID</td>
<td>A unique identifier automatically generated by the database.</td>
</tr>
</tbody>
</table>
Table Name: Platforms

**Table description:** This table contains the client operating system defined internally by a User Agent mapping.

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Column Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Platform_ID</td>
<td>The primary key. A unique number automatically generated by the database.</td>
</tr>
<tr>
<td>Name</td>
<td>The Operating System name as defined via the User Agent global settings.</td>
</tr>
</tbody>
</table>

Table Name: Protocols

**Table description:** Lists all network protocols processed during data collection.

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Column Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Protocol_ID</td>
<td>The primary key. A unique identifier generated automatically by the database.</td>
</tr>
<tr>
<td>Name</td>
<td>The name of the network protocol.</td>
</tr>
</tbody>
</table>

Table Name: Referrer

**Table description:** This table contains all the referrers processed during data collection.

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Column Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Referrer_ID</td>
<td>The primary key. A unique number automatically generated by the database.</td>
</tr>
<tr>
<td>RefHost_ID</td>
<td>A foreign key to the ID column of the REFERRERHOST table.</td>
</tr>
<tr>
<td>ReferrerURL_ID</td>
<td>A foreign key to the ID column of the REFERRERURL table.</td>
</tr>
</tbody>
</table>

Table Name: ReferrerHost

**Table description:** This table stores all the referrer site host names processed during data collection.

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Column Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RefHost_ID</td>
<td>The primary key. A unique number automatically generated by the database.</td>
</tr>
<tr>
<td>IsLocal</td>
<td>Indicates whether this referrer is internal or external. 0 - external referrer 1 - internal referrer</td>
</tr>
<tr>
<td>Name</td>
<td>Host name part of referrer as parsed from the Web log record.</td>
</tr>
</tbody>
</table>

Table Name: ReferrerURL

**Table description:** This table contains the referrer pages processed during data collection. Note that the referrer site’s host name is stored in the ReferrerHost table.
<table>
<thead>
<tr>
<th>Column Name</th>
<th>Column Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReferrerURL_ID</td>
<td>The primary key. A unique number automatically generated by the database.</td>
</tr>
<tr>
<td>Name</td>
<td>The name of the referrer page as parsed from the Web log record.</td>
</tr>
</tbody>
</table>

**Table Name: Resources**

**Table description:** This table contains all resources (URLs) processed during data collection.

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Column Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resource_ID</td>
<td>The primary key. A unique number automatically generated by the database.</td>
</tr>
<tr>
<td>IsPage</td>
<td>Indicates whether this resource is a page. 0 = false 1 = true</td>
</tr>
<tr>
<td>IsImage</td>
<td>Indicates whether this resource is an image. 0 = false 1 = true</td>
</tr>
<tr>
<td>IsDirectory</td>
<td>Indicates whether this resource is a file directory. This column is not currently implemented and is RESERVED for future Feedback component use.</td>
</tr>
<tr>
<td>Name</td>
<td>The resource string as parsed from the Web log record.</td>
</tr>
</tbody>
</table>

**Table Name: Session_Facts**

**Table description:** This table contains all the visits calculated from the Web log data. Note: The first date stamp (date ID=1) corresponds to 1/1/1995. Time stamp IDs are calculated such that each second of the day has a separate ID.

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Column Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>firstHitTimestamp</td>
<td>Timestamp of the first hit in session expressed as a JAVA timeInMillis value.</td>
</tr>
<tr>
<td>lastHitTimestamp</td>
<td>Timestamp of the last hit in the session expressed as a JAVA timeInMillis value.</td>
</tr>
<tr>
<td>Result_ID</td>
<td>Foreign key to the ID column of the RESULT table. For your use in classifying this session.</td>
</tr>
<tr>
<td>User_ID</td>
<td>The visitor to your site. A foreign key to the Users table.</td>
</tr>
<tr>
<td>Referrer_ID</td>
<td>The referrer for this session. A foreign key to the Referrer table.</td>
</tr>
<tr>
<td>Sessions</td>
<td>The actual number of sessions. This field will always contain the value. It is used during Report Database processing.</td>
</tr>
<tr>
<td>LastUpdated</td>
<td>Timestamp of when this row was last created/updated.</td>
</tr>
<tr>
<td>PageViews</td>
<td>Number of page views that are part of this session.</td>
</tr>
<tr>
<td>Column Name</td>
<td>Column Description</td>
</tr>
<tr>
<td>--------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Duration</td>
<td>Duration of the session up to the last hit. The Feedback component cannot calculate the full duration of the session because there is no explicit end-of-session signal from the Web log.</td>
</tr>
<tr>
<td>Hits</td>
<td>Number of hits that are part of this session.</td>
</tr>
<tr>
<td>SessionIdentifier</td>
<td>The session identifier string from the Web log. This is sometimes referred to as a session cookie.</td>
</tr>
<tr>
<td>SessionTimestamp</td>
<td>Beginning of this session expressed as a JAVA timeInMillis value.</td>
</tr>
<tr>
<td>LocalDate_ID</td>
<td>This is the starting date of the session in the local time zone of the Web server that handled this session. See also, Populating CALENDAR and TIMEOFDAY tables.</td>
</tr>
<tr>
<td>LocalTimeOfDay_ID</td>
<td>This is the start time of the session in the local time zone of the Web server which handled this session. See also, Populating CALENDAR and TIMEOFDAY tables.</td>
</tr>
<tr>
<td>ImportHistory_ID</td>
<td>This column is not used.</td>
</tr>
<tr>
<td>Session_ID</td>
<td>The primary key; a unique number automatically generated by the database.</td>
</tr>
<tr>
<td>UserAgent_ID</td>
<td>The signature of the Web browser used for this session.</td>
</tr>
<tr>
<td>EntryResource_ID</td>
<td>The first resource viewed during this session.</td>
</tr>
<tr>
<td>Network_ID</td>
<td>Pointer to the Networks table.</td>
</tr>
<tr>
<td>GMTDate_ID</td>
<td>This is the start date of the session translated to the GMT time zone. See also, Populating CALENDAR and TIMEOFDAY tables.</td>
</tr>
<tr>
<td>ExitResource_ID</td>
<td>The last resource viewed during this session.</td>
</tr>
<tr>
<td>GMTTimeOfDay_ID</td>
<td>This is the start time of the session translated to the GMT time zone. See also, Populating CALENDAR and TIMEOFDAY tables.</td>
</tr>
</tbody>
</table>

**Table Name: Result**

**Table description:** For your use in classifying sessions.

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Column Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Result_ID</td>
<td>The primary key. We recommend that you use numbers automatically generated by the database. In DB2/UDB, this column is defined as IDENTITY GENERATED BY DEFAULT. On ORACLE, the sequence RESULT_ID is created for your use with this table.</td>
</tr>
<tr>
<td>Name</td>
<td>For your use.</td>
</tr>
</tbody>
</table>
Table Name: SessionParms

**Table description:** This table lists all the Key/Value pairs associated with the session referrer.

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Column Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parms_ID</td>
<td>Foreign key to the ID column of the PARMS table.</td>
</tr>
<tr>
<td>Session_ID</td>
<td>Foreign key to the ID column of the SESSION_FACT table.</td>
</tr>
<tr>
<td>ParmType</td>
<td>Identifies the type of Key/Value data.</td>
</tr>
</tbody>
</table>

Table Name: Subdomains

**Table description:** This table is RESERVED for future Feedback component use.

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Column Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Domain_ID</td>
<td>Foreign key to the ID column of the DOMAIN table. The TCP/IP domain this subdomain belongs to. Name Subdomain name string from a resolved TCP/IP address.</td>
</tr>
<tr>
<td>Subdomain_ID</td>
<td>The primary key, a unique number automatically generated by the database.</td>
</tr>
</tbody>
</table>

Table Name: TimeOfDay

**Table description:** Initially, this table is empty. Table rows are reserved to list all seconds in a day, when the database is populated. The table can be populated with a new configuration task for applications that use custom logging or reports that reference this table.

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Column Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minute</td>
<td>a number in the range 0..59</td>
</tr>
<tr>
<td>Hour</td>
<td>a number in the range 0..23</td>
</tr>
<tr>
<td>Second</td>
<td>a number in the range 0..59</td>
</tr>
<tr>
<td>AggrHourID</td>
<td>For each database row, indicates which hour it belongs to.</td>
</tr>
<tr>
<td>TimeOfDay_ID</td>
<td>The primary key. A unique number automatically generated by the database.</td>
</tr>
<tr>
<td>SecondsSinceMidnight</td>
<td>The number of seconds elapsed since midnight.</td>
</tr>
<tr>
<td>TimeSpan_ID</td>
<td>Foreign key to the ID column of the TIMESPAN table.</td>
</tr>
</tbody>
</table>
Table Name: TimeSpan

Table description: For your use in classifying time of day values.

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Column Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TimeSpan_ID</td>
<td>The primary key. We recommend that you use numbers automatically generated by the database. In DB2/UDB, this column is defined as IDENTITY GENERATED BY DEFAULT. On ORACLE, the sequence TIMESPAN_ID is created for your use with this table.</td>
</tr>
<tr>
<td>Name</td>
<td>For your use.</td>
</tr>
</tbody>
</table>

Table Name: UserAgents

Table description: This table contains all UserAgents processed during data collection.

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Column Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Browser_ID</td>
<td>Foreign key to the ID column of the BROWSERS table.</td>
</tr>
<tr>
<td>Name</td>
<td>The UserAgent string as available from the Web page hit.</td>
</tr>
<tr>
<td>Platform_ID</td>
<td>Foreign key to the ID column of the PLATFORMS table.</td>
</tr>
<tr>
<td>UserAgent_ID</td>
<td>The primary key. A unique number automatically generated by the database.</td>
</tr>
</tbody>
</table>

Table Name: Users

Table description: This table contains all userids processed from data collection.

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Column Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FirstVisitDate_ID</td>
<td>Date of first visit to the site by this userid. See also, Populating CALENDAR and TIMEOFDAY tables.</td>
</tr>
<tr>
<td>Name</td>
<td>The userid string as parsed from the Web log record.</td>
</tr>
<tr>
<td>User_ID</td>
<td>The primary key. A unique number automatically generated by the database.</td>
</tr>
</tbody>
</table>

Table Name: Value

Table description: Lists all values (from Key/Value pairs) encountered during data collection.

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Column Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value_ID</td>
<td>A unique identifier automatically generated by the database.</td>
</tr>
</tbody>
</table>
The following tables are included in the Feedback schema, however, the tables are not used and are not populated.

- Aggregate_Content
- AggregateKey
- Aggregates
- AggregateStatus
- Categories
- Category_Patterns
- Category_Sets
- CategoryMap
- CookiesStatus (Note: This table may be pre-populated, however, the table is not used.)
- JavaScriptStatus
- Linkage
- Log_file_status
- Logs
- ResetStatus
- ReturnCodes
- ServerNodes
- Web_Nodes

**Key value pairs:**

The additional information passed by rule logging events and certain bean logging events is logged to the Feedback database in the form of key value pairs.

Rule logging events and certain bean logging events pass additional information such as rule names, action names, resource collection names, resource identifiers, etc. when they are fired (refer to the individual rule logging and logging bean sections for more information on what information may be passed). This data is logged to the Feedback database in the form of key value pairs. Key strings representing each type of data are stored in the key table of the Feedback schema, while the values that each key references are recorded in the value table of the Feedback schema. Refer to the Feedback schema diagram for more information on how to join these tables to retrieve a mapping of values onto keys.

The following tables describe what each key represents:

**Rule related keys:**

<table>
<thead>
<tr>
<th>Key</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>wcpRule</td>
<td>The name of the rule that was fired</td>
</tr>
<tr>
<td>wcpCampaign</td>
<td>The campaign associated with the rule</td>
</tr>
</tbody>
</table>
Table 141. Key descriptions (continued)

<table>
<thead>
<tr>
<th>Key</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>wcpCollection</td>
<td>The name of the resource collection associated with the rule</td>
</tr>
<tr>
<td>wcpResourceId</td>
<td>An identifier for each individual resource the associated resource collection</td>
</tr>
</tbody>
</table>

Action bean related keys:

<table>
<thead>
<tr>
<th>Key</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>wcpAction</td>
<td>The name of the action being logged</td>
</tr>
<tr>
<td>wcpActionResId</td>
<td>The resource (if any) that has been acted upon</td>
</tr>
<tr>
<td>wcpActionCollection</td>
<td>The name of the resource collection associated with the resource</td>
</tr>
<tr>
<td>wcpActionRule</td>
<td>The name of the rule (if any) that surfaced the resource being acted upon</td>
</tr>
<tr>
<td>wcpActionCampaign</td>
<td>The campaign (if any) associated with the rule</td>
</tr>
</tbody>
</table>

Category bean related keys:

<table>
<thead>
<tr>
<th>Key</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>wcpCategory</td>
<td>A category string being logged</td>
</tr>
</tbody>
</table>

Rating bean related keys:

<table>
<thead>
<tr>
<th>Key</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>wcpRating</td>
<td>The rating value being logged</td>
</tr>
<tr>
<td>wcpRatingResId</td>
<td>The resource (if any) that is associated with the rating</td>
</tr>
<tr>
<td>wcpRatingCollection</td>
<td>The name of the resource collection associated with the resource</td>
</tr>
</tbody>
</table>

Developing a personalized portlet

This exercise demonstrates how to use Personalization features of WebSphere Portal and Rational Application Developer to build your first personalized portlet. Your final result is a working portlet that uses Personalization rules and content spots to display personal news based on user attributes (or profiles).

The final portlet looks like the following example:
The demonstration Web site is organized into the following three pages:

1. **Front Page**
   - **Internal News** displays internal YourCo news for the user. This portlet displays the news upon clicking upon one of the links.
   - **General News** displays external worldwide news articles for users. This portlet messages to the GeneralNewsDetails upon clicking upon one of the links.
   - **UserInfo** displays the current user's information. This information is used in rules to target content throughout the website to the particular preferences of your users. This user information can be modified using the edit feature.

2. **Partners Page**
   - **Products** displays products for the current user. This portlet messages to the PartnersDetails upon clicking upon one of the links.
   - **Promotions** runs a simple rule targeting the information in the portlet to the user's preferences.
   
   The promotions portlet also demonstrates running a campaign and the splitting of rules within a campaign. In this case, between March 1 and April 16, 2005 the Tax Season campaign will run. Within this campaign, one content spot is filled with 2 distinct rules, Get Tax Season Promotions and Get 3 Promotions about IRAs. Get Tax Season Promotions will be run 70% of the time and Get 3 Promotions about IRAs the remaining 30%.

   A second campaign is also running in the promotions spot between April 14 and April 15, 2005. This campaign has a higher priority than the Tax Season campaign so it will run at this time.

3. **Services Page**
   - **Services** displays services that are offered to the user. Clicking on one of the services will display the details of the service. If logging is enabled, clicking on one of the services will log the fact that this particular type of service was selected by the user.
   - **Offers** displays current offers for the user. This portlet demonstrates calling an analysis bean from a rule.

Personalization can use database content. This exercise uses the sample database that comes with the Personalization sample. The following tables are used:

- **PZN_USER**: contains user information, including profile attributes
- **PZN_OFFERS**: contains text for special offers categorized by customer type

<table>
<thead>
<tr>
<th>FIRST_NAME</th>
<th>LAST_NAME</th>
<th>DEPARTMENT</th>
<th>CUSTOMERTYPE</th>
<th>USERNAME*</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scott</td>
<td>Green</td>
<td>Loans</td>
<td>Gold</td>
<td>scott</td>
</tr>
</tbody>
</table>
Table 142. Sample data from PZN_USER table (primary key = USERNAME) (continued)

<table>
<thead>
<tr>
<th>FIRST_NAME</th>
<th>LAST_NAME</th>
<th>DEPARTMENT</th>
<th>CUSTOMERTYPE</th>
<th>USERNAME*</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tawana</td>
<td>Streble</td>
<td>Human Resources</td>
<td>Platinum</td>
<td>tawana</td>
</tr>
<tr>
<td>Marge</td>
<td>Roorda</td>
<td>Human Resources</td>
<td>Platinum</td>
<td>marge</td>
</tr>
<tr>
<td>Andy</td>
<td>McPherson</td>
<td>IT</td>
<td>Titanium</td>
<td>andy</td>
</tr>
</tbody>
</table>

Table 143. Sample data from PZN_OFFERS table (primary key = OFFER_ID)

<table>
<thead>
<tr>
<th>OFFER_ID*</th>
<th>CUSTOMERTYPE</th>
<th>TITLE</th>
<th>DETAILS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Platinum</td>
<td>2nd House Mortgage</td>
<td>Excellent Mortgage rates on your beach house.</td>
</tr>
<tr>
<td>2</td>
<td>Gold</td>
<td>Add other Cards</td>
<td>Add other members of your household to your account.</td>
</tr>
<tr>
<td>3</td>
<td>Platinum</td>
<td>ARM Mortgage</td>
<td>1/3 5.9% APR Mortgage for 27 years.</td>
</tr>
<tr>
<td>4</td>
<td>Titanium</td>
<td>Free Checking Deluxe</td>
<td>Free Checking with the attainment of a YourCo Financial Credit Card.</td>
</tr>
</tbody>
</table>

Notice the common element in these tables is the column CUSTOMERTYPE. This column is important when you invoke Personalization later in the exercise.

Prerequisites for the Personalization portlet exercise

This exercise is intended for users with significant WebSphere Portal administration experience and strong portlet Java development skills.

Software requirements:

- WebSphere Portal, including IBM Web Content Manager. You can use the default database or another supported database.
- Rational Application Developer Version 7.0.0.6 or 7.5 with the Portal Tools feature. No agent controller is required. No WebSphere Test Environment or integrated Portal environment is used in this example. For additional information about Rational Application Developer, visit the IBM software library.
Use the administration tools provided with the portal to perform a variety of day-to-day administration tasks. There are two methods for editing portal setup: portlets and the XML configuration interface. The administration portlets are a convenient way to make real-time updates to the portal's configuration, while the XML configuration interface is well suited to more advanced administration, including batch processing of updates.

**Install the Personalization sample**

This exercise installs the Portal Personalization sample and configures your database for the Personalization sample. No additional database requirements are needed.

Ensure you have the basic system requirements listed in the Prerequisites for the Personalization portlet exercise topic.

1. Start the IBM WebSphere Portal server.
2. From a command prompt, navigate to `wp_profile_root/ConfigEngine`.
3. Enter the following commands to install the Personalization sample and create the users for this sample:
   - **UNIX:**
     ```
     ./ConfigEngine.sh create-pzndemo-users -DPortalAdminPwd=password
     -DWasPassword=password
     ./ConfigEngine.sh install-pzndemo -DPortalAdminPwd=password
     -DWasPassword=password
     ```
   - **Windows:**
     ```
     ConfigEngine.bat create-pzndemo-users -DPortalAdminPwd=password
     -DWasPassword=password
     ConfigEngine.bat install-pzndemo -DPortalAdminPwd=password
     -DWasPassword=password
     ```
4. Stop the WebSphere Portal server.
5. Restart the WebSphere Portal server.
6. Confirm that the users were created by logging in as the following users. Use `pzndemo` as the password for each user.
   - scott
   - marge
   - tawana
   - andy
7. Navigate to **Personalization > Demo**, then click through the different pages and portlets for each user. Notice the different information that displays for each user.
8. Log out of WebSphere Portal.

9. Stop the WebSphere Portal server. This action is necessary to free the single connection to the Derby database, so you can continue the next steps in this exercise.

The sample demo and database configuration is complete. You can now begin coding a basic personalized portlet.

**Create the JSP file in Rational Application Developer**
Create a basic JSR 168 portlet file for the Personalized Offers project and add the welcome text to the portlet.

Before you begin, complete the following prerequisites:
- Ensure you have properly installed the demo and databases.
- Ensure that the WebSphere Portal Server is stopped so that you can make a connection to the database.
1. Launch Rational Application Developer.
2. Click **File > New > Project** and select **Portlet Project**.

3. Click **Next**. The Confirm Enablement window appears. Click **OK** to enable Portal Development.
4. Complete the New Portlet Project screen with the following information:
   a. In the **Project Name** field, type **Pers_Offers**.
   b. Use the default project contents.
   c. Select **WebSphere Portal v6.1 stub** from the **Target Runtime** list.
   d. Select **Add project to an EAR** and in the field **EAR Project Name** select **Pers_OffersEAR** from the list.
   e. In the **Portlet API** field, select **JSR 168 Portlet** from the list.
   f. Select **Create a portlet** and specify the following properties:
      • In the **Portlet Name** field, type **Pers_Offers**.
      • In the **Portlet Type** field, select **Basic Portlet** from the list.

5. Click **Next**.

6. Accept the default Portlet Settings and click **Next**.
7. No Actions are necessary. Clear any selected actions and click Next.
8. No Advanced Settings are necessary. Click Finish.
9. The JSP file opens automatically in Design mode. Edit the text of the portlet to display Welcome to Personalized Offers!
10. Save and close the JSP.

You have created a JSP file using Rational Application Developer.

You can now create the Personalization content resource classes and content spot.

**Create the Personalization content resource classes and content spot**

View the steps to create the Personalization content resource classes and content spot using IBM Rational Application Developer.

1. In the Project Explorer tab in Rational Application Developer, right-click on the Pers_Offers project and choose **New > Other**.
2. From the New window, select **Portal > Personalization > Content or User Resource**.
3. Click Next.
4. Select the following options by clicking the appropriate radio button:
   a. SQL
   b. Web Content
5. Select **Create a new connection** and click **Next**.
6. Enter the following values:
   a. From the JDBC driver drop-down list, select Derby Embedded JDBC Driver.
   b. From the Database location field, click Browse. Navigate to and select \wp_profile_root\PortalServer/derby/pzndemo_db/
   c. From the Class location field, click Browse. Navigate to and select \AppServer_root\derby/lib/derby.jar
7. Click **Next**. The personalization resource wizard opens.
8. Expand **PZNDEMO**.
9. On the **Tables** tab, select **PZN_OFFERS**, and click the arrow to add the table to the list of selected tables. Click **Primary Table** to mark it as the primary table.
10. Select the **Columns** tab.

11. On the **Columns** tab, move all columns to the selected columns list by clicking the double arrow button. Notice the primary key is the column **Offer_ID**.
12. Click the **Mappings** tab.
13. On the **Mappings** tab, select **Customertype** and click **Populate**.
14. Click the Select buttons and expand PZNDEMO > PZN_OFFERS to select CUSTOMERTYPE for the Description and Value fields. Click OK.

15. Click the Deployment tab.

16. On the Deployment tab, change the datasource to jdbc/pzndemo. This datasource is defined in WAS by the Personalization demo program installation.
17. Click Next.
18. Set the package name as pers_offers. Select Generate a Content Spot for this resource. Select Include schema names in the generated Resource Runtime Manager.

19. Click Finish.

You can now see the new JAVA classes in your project:
You can now see the new JAVA classes in your project, and create the Personalization user resource classes and content spot.

**Create the Personalization user resource classes and content spot**

Use the Project Explorer of IBM Rational Application Developer to create the user resource classes and content spot for the Personalization demo that creates the Personalized Offers portlet for different customer profiles.

1. In the Project Explorer tab of Rational Application Developer, right click the Pers_Offers project and choose **New > Other**.
2. From the New window, select **Portal > Personalization > Content or User Resource**.

![Project Explorer][1]
3. Click Next
4. Select the following options by clicking the appropriate radio button:
   a. SQL
   b. Web users
5. Select **Use an existing connection** and select `pzdemon.db` from the list of existing connections.
6. Click Next. The personalization resource wizard opens.
7. Expand PZNDEMO.
8. On the Tables tab, highlight PZN_USER by single clicking on it. Click the appropriate arrow button to select the table.
9. Right click on **PZN_USER** and select **Edit Table**.

10. Change the display name of **PZN_USER** to **Per_Offers_User** to avoid naming conflicts with the previously installed demo code. Click the Primary Table button to mark it as the primary table.
11. Select the **Columns** tab
12. On the **Columns** tab, move all of the columns to the selected columns area by clicking the double arrow button. Notice the primary key is the column **USERNAME**.
13. Click the Mappings tab.

15. Click the Select buttons and expand PZNDEMO > PZN_USER to select CUSTOMERTYPE for the Description and Value fields. Click OK.

16. Click the Deployment tab.

17. On the Deployment tab, change the datasource to jdbc/pzndemo. This datasource is defined in WAS by the Personalization demo program installation.
18. Click Next.
19. Set the package name as pers_offers. Select **Generate a Content Spot for this resource**. Select **Include schema names in the generated Resource Runtime Manager**.

20. Click Finish.
You can now see the new JAVA classes in your project:

![Project Explorer](image)

You can now finish coding the portlet JSP.

**Finish coding the portlet JSP**

Use the Rational Application Developer Project Explorer to work with the `pers_offers.Pzn_offersSpot.class` and the `pers_offers.Per_Offers_UserSpot.class` to define the context parameters `offersSpot` and the `userSpot`, respectively. Code the `setRequest` calls to pass the user context to personalization.

1. From the RAD Project Explorer, open `PersOffers/WebContent/pers_offers/jsp/html/Pers_OffersPortletView.jsp`
2. Drag and drop the `pers_offers.Pzn_offersSpot.class` file from `PersOffers/WebContent/WEB-INF/classes/pers_offers` onto the JSP file. This is one of the classes you generated in the content resource wizard.
3. In the Properties window, select offersSpot from the Id list.

4. Drag and drop the pers_offers.Per_Offers_UserSpot.class file from PersOffers/WebContent/WEB-INF/classes/pers_offers onto the JSP file.

5. In the Id field, type userSpot.
6. Switch to source mode in the JSP file editor.
7. Replace the existing text with the following sample. The setRequest calls pass the user context to personalization:

```jsp
<%@ page session="false" contentType="text/html" import="java.util.*, pers_offers.*" %>
<%@ taglib uri="/WEB-INF/tld/portlet.tld" prefix="portletAPI" %>
<portletAPI:init/>
<jsp:useBean class="pers_offers.Pzn_offersSpot" id="offersSpot"></jsp:useBean>
<jsp:useBean class="pers_offers.Per_Offers_UserSpot" id="userSpot"></jsp:useBean>
<%
offersSpot.setRequest(request);
userSpot.setRequest(request);
%>
```
8. Save the file and check for syntax errors.
9. Insert the following code at the end of the JSP file:

```jsp
<DIV style="margin: 6px">
<H3 style="margin-bottom: 3px">Welcome to Personalized Offers!</H3>
Here is a personalized offer:<BR>
<%=offersSpot.getRuleContent()[0].getTitle() %>
<br>
<%=offersSpot.getRuleContent()[0].getDetails() %>
```
10. Save and close the JSP file.
You have completed the basic coding of the personalized portlet. You can now export the WAR file from Rational Application Developer and install the portlet in WebSphere Portal.

**Export the WAR file and install the portlet**

Export the IBM Rational Application Developer project to a war file and install the portlet on a portal page.

Ensure you have properly followed the steps in **Finish coding the portlet JSP**

1. Export the Rational Application Developer project to a war file called PersOffers.war.
   b. In the Destination field, type filepath/PersOffers.war.
   c. Select Export source files.
   d. Click Finish.
2. Start WebSphere Portal.
3. Log in as the Portal administrator (wpsadmin).
5. Click Install.

6. Complete the installation of Pers0ffers.war. Verify installation was successful.

8. Search for the Pers_Offers portlet and grant Privileged User to All Authenticated Portal Users.
9. Create a new page called Pers Offers by completing the following steps:
   a. Select Portal User Interface > Manage Pages.
   b. Select Content Root > Personalization.
   c. Select New Page.
   d. Label the Page Pers Offer.
10. Add the Pers_Offers portlet to the page.

11. Click Done.

   **Note:** The portlet is not ready to execute yet. If you try, you will get an error.

You can now import the Personalization workspace resource collections.

**Import Personalization Workspace resource collections**

Use the Personalization Navigator to create Workspace folders for the resource collections. Then import the resource collections into the Workspace of the Personalization Navigator.

Before you can use the content and user resources in the Personalization Navigator, you must place these class files into a directory accessible by that portlet. To do this, export the pers_offers folder in RAD under Pers_Offers/Java Resources/JavaSource as a JAR file. Make the target location `/PortalServer_root/pzn/prereq.pzn/collections/pers_offers.jar`. Accept the defaults and click Finish, then restart the server.

Ensure you have properly followed the steps in "Export the WAR file and install the portlet" on page 485.

1. Click the Personalization tab.
2. In the Personalization Navigator portlet, click New > Folder
3. Enter the name Pers Offers and click **Create**.

4. Change to the **Pers Offers** folder.

5. Click **Import**.


7. Click **Import**.

8. See the resource collection in the Workspace.
9. Do the same to import the Pers_offers_User.hrf file.

You can now create a simple content rule.

**Create a simple content rule**
Use the Personalization Editor to create a business rule for a resource collection.

Before you begin this procedure, ensure you have imported the resource collections into the Personalization Workspace.

1. While still in the Pers_Offers folder, click **New > Rule**.
2. Type **Show Gold Offers** in the New Rule field.

3. Select **Select Action** from the **Rule Type** drop-down list.
4. Click **content** next to **Select**.
5. Select **Pzn_offers** from the drop-down list.
6. Click **attribute** and select **Customertype** in the drop-down list.

7. Hover over **value** and click the greater than symbol, >, then select **Gold** in the drop-down list.
8. The completed rule will look like the following example. Click **Save**.

You can now create a content spot for your rule.

**Create a content spot**

Use the Personalization Editor to create a placeholder for the rule that renders the selected content on a Web page. This placeholder is the **content spot**. Specify which rule to place in the content spot; this is called mapping the rule to the content spot or creating a **rule mapping**. In this example, the content is a resource collection.
Before you begin this procedure, ensure you have created the simple content rule
*Show Gold Offers* for the *Personalized Offers* resource collection.

**Note:** You must use the same name as the original content spot’s display name. Do not place the new content spots into a folder, unless your display name already includes slashes. Place them directly into the root Workspace.

1. Return to the root directory in the Personalization Navigator.
2. Click New > Content Spot to display the Personalization Editor.
3. Type `Pzn_offersSpot` in the title field of the content spot.
4. Select `Pzn_offers` from the Output Type drop-down list.

5. Click the button in the Default Mapping section.

6. Expand the Pers Offers folder in the tree, select Show Gold Offers and click OK.
7. Click **Save**. The content spot should look similar to the following screen.

![Personalization Picker](image)

8. Test the content spot with the Personalization Editor portlet by selecting it in the Personalization Navigator and clicking the **Preview** tab in the Personalization Editor.

   **Note:** If the previous values appear on the Preview tab, the rule and content spot work.
9. View the page with your Pers_Offers portlet.

You can now edit the JSP file to contain dynamic code.

**Enhance the Personalized Portlet**
Introduce enhancements in the Personalized Portlet.

Ensure you have properly followed the steps in “Create a content spot” on page 490.

We need a few more things in this portlet to make it interesting:
- User profiling
- Special userid handling, so we can build the user profiles
- A more complex binding rule
- A dynamic html table, to display any number of special offers

**Insert dynamic table html/jsp code**
Use Rational Application Developer to code the dynamic table in the Personalized Offers portlet JSP. Export the project as a Web archive (WAR) file. Open the WebSphere Portal Administration page for Portlet Management and update the Web module containing the Personalized Offers portlet.

Before you begin this procedure, ensure that you have created a content spot.
1. Open Rational Application Developer and open Pers_OffersPortalView.jsp in edit mode.
2. Insert the following code at the end of the JSP page:

   `<HR>
   Here are all your personalized offers:`
3. Save and close the file. Verify no errors other than exception handling errors.
4. Export the project as a war file. Close Rational Application Developer.


6. Click the Update button to update the Web module.

7. Complete the installation process using the newly created war file.

You can now modify the resource collection properties for Pzn_Offers_User.

**Modify resource collection properties**

Use the Personalization Navigator to specify the Translator Class for the resource collection.

Before you begin this procedure, ensure that you have inserted the dynamic table code into the Personalized Offers portlet.

1. Open the Personalization Navigator.
2. Navigate to the Pers_Offers folder.
3. Select the Per_Offers_User resource collection.

4. In the Personalization Editor, click **Edit**.
5. Set the Translator Class to `com.ibm.websphere.personalization.security.RegularExpressionSecurityTranslator` and click Save.

You can now create a user profiler rule.

**Create the user profiler rule**

Use the Personalization Editor to create a profiler rule for users who qualify as customers for Gold Offers in the Personalized Offers resource collection.

Before you begin this procedure, ensure that you have modified the resource collection properties to specify the Translator Class.
In the Personalization Editor, follow these steps:

1. Within the Pers Offers folder, click New > Rule.
2. Enter the following values:
   a. In the New Rule field, type Pers Offers User Profiler.
   b. Select Profiler in the Rule Type menu.
3. Click Profile, type Gold in the Profile field, and click Submit.
4. Click attribute and select Per_Offers_User in the drop down, then Customertype in the expanded drop down.
5. Click the greater than symbol, >, next to value and select Gold in the drop down list.
6. Click **add Profile** to add another conditional expression to the profiler rule.

7. Complete the profiler rule by adding profiles that define distinct sets of users. Each profile expresses the conditions of the user type as illustrated in the following example.
8. Click Save.

You can now create some additional advanced rules for your personalized portlet.

**Create additional advanced rules**

For each customer type specified by a profile, add to the business rule (the profiler) by selecting actions to build the syntax of the rule.

Ensure you have properly followed the steps in "Create the user profiler rule" on page 496.

1. Within the Pers Offers folder, select New > Rule
2. Type Show Platinum Offer in the New Rule field.
3. Select Select Action from the Rule Type drop-down list.
4. Edit the rule so it has the following values and click **Save**.

5. In the Pers Offers folder, create another rule based on the following:

6. Within the same folder, select **New > Rule**
7. Type **Show Offers by Customer Type** in the **New Rule** field.
8. Select **Binding** from the **Rule Type** drop-down list.
9. Click **Profiler > Select a Profiler**.
10. Expand the Pers Offers folder, select **Pers Offers User Profiler** and click **OK**.
11. Click Profile and select Gold from the drop-down list.

12. Click do Action > Select Actions....
13. Select **Show Gold Offers** and click **OK**.

**Tip:** To find **Show Gold Offers**, you may reorder the columns to make navigation easier or page through the list of rules.

14. Click **Profile** and choose **Platinum**. Complete the fields as follows:
You can now change your content spot mapping.

**Change content spot rule mapping**

Change the default rule mapping to the new binding rule that you created for different customer types who will use the Personalized Offers portlet. Test the portlet for use by all customer types. Verify that the portlet displays the personalized content that is specified by the business rule for each user profile.

Before you begin this procedure, ensure you have properly created additional advanced rules.

1. In the Personalization Navigator, select the content spot **Pzn_offersSpot**, located in the **Workspace Root**.
2. In the Personalization Editor portlet, click **Edit**.
3. Change the default rule mapping to the new binding rule and click **Save**.
4. Log in to the Portal as Scott, with password pzndemo, a Gold customer.
5. Run the enhanced Personalized portlet.

6. Log out, then log in as Marge, with password pzndemo, a Platinum customer.
7. Again, run the portlet. See the differences?

If you run the portlet as wpsadmin, it fails because wpsadmin does not exist in the user resource database, and the programmer who coded the JSP did not code the proper error handling.

Congratulations! You have finished building a Personalized portlet.

The next topic, Personalized list portlet, shows how to use this portlet instead of coding the Rational Application Developer portlet.
**Personalized List portlet**
The Personalized List portlet provides a ready-to-use portlet for displaying personalized content from rules, content spots, or resources. In many cases it eliminates the need to code new portlets and JSP files yourself. You can use this portlet instead of coding the IBM Rational Application Developer portlet.

**Restriction:** The Personalized List portlet is not intended to be used with the Web Content resource collection or with rules involving the Web Content resource collection. To display Web Content rules, use a Portal Personalization Component in IBM Web Content Manager. When the Web Content resource collection is used with the Personalized List portlet, certain attributes like authoring template will show raw values from the repository that cannot be translated to a readable form. The details page of the Personalized List portlet will not show the results of the rendered content. The details page will show some attribute from the content, such as the creator or last modified date, which is not suitable for production use of content from Web Content Manager.

The intended use of the Personalized List portlet is to display personalized lists of documents. The Personalized List can also be used with some generated or custom resource collections.

1. Copy the pers_offers folder from the /Pers_Offers.war/WEB-INF/classes directory under your Pers_Offers portlet in the installedApps location into /wp_profile_root/PortalServer/pzn/collections. You may need to create this classes folder first.
2. Restart IBM WebSphere Portal.
3. Log in as the admin user.
5. Search for the Personalized List portlet.
6. Make a copy of the Personalized List portlet, named **Personalized List Special Offers**.

7. Set access control for the new portlet so **All Authenticated Portal Users** are Privileged Users.
8. Add the Personalized List Special Offers portlet to a new page under Personalization called **Pers List Portlet - Rules**.
9. Display the new page with the new portlet. Click the portlet menu on the Personalized List Special Offers portlet, and select **Configure** from the drop-down list.

10. Click the menu icon under **Which Personalization resources are retrieved** and select **Select a Rule, Content Spot or Resource Collection**.

11. Expand the **Pers Offers** folder and select **Pzn_offers** and click **OK**.
12. Click Display Options.

13. Select the following options and click OK.
14. Click **OK** again to see the portlet.

15. Click on one of the title links to see the details of that resource.
16. Click Back.
17. Configure the portlet again to show more personalized offers.
18. From the Personalization Picker, select the content spot **Pzn_offersSpot** under the Workspace Root and click **OK**.

19. Click **Display Options**, set the following values, and click **OK**.
20. Set the Title Attribute and Detail Attribute values to Fixed. Click OK.
21. Complete the configuration and see that the Personalized List Special Offers portlet is empty. This is because the administrator has not entered customer details in the pzndemo database for the user resource being used.

22. Log in as Scott and view the Personalized List Special Offers portlet. The portlet displays the personalized offers based on the rule, Show Offers By Customer Type, which is mapped to the content spot Pzn_offersSpot.

23. Test this portlet, logging in as each pzndemo user having a different profile. You should get the same content results that you saw previously using your own custom-built portlet, Pers_Offers, except you do not have to code a portlet or a JSP file.

Uninstall Personalization sample and database
View the steps to uninstall Personalization sample demo, database, and users.

Note: On z/OS, this function is currently not supported.
1. From a command prompt, navigate to wp_profile_root/ConfigEngine.
2. From this command prompt, run the following commands:
   - UNIX:
     ```
     ./ConfigEngine.sh remove-pzndemo -DPortalAdminPwd=password
     -DWasPassword=password
     ./ConfigEngine.sh remove-pzndemo-users -DPortalAdminPwd=password
     -DWasPassword=password
     ```
   - Windows:
     ```
     ConfigEngine.bat remove-pzndemo -DPortalAdminPwd=password
     -DWasPassword=password
     ConfigEngine.bat remove-pzndemo-users -DPortalAdminPwd=password
     -DWasPassword=password
     ```

The sample demo, database, and users will be removed.

Note: The database connection to the included Apache Derby database may stay open and prevent removal of the database directory. Restart IBM WebSphere Portal and run ConfigEngine.bat/sh remove-pzndemo-database
-DPortalAdminPwd=password -DWasPassword=password if this is a problem. The directory PznDemo in the root of the Portal Personalization Workspace may need to be removed manually.

Personalization programming reference
IBM WebSphere Portal provides the programming model, processes, and APIs for the Personalization rules and resource engines.
Preparing your personalized application

Before deploying applications that take advantage of the features of Portal Personalization, certain features must be configured in order to work properly. The Feedback and LikeMinds components of Personalization both communicate with their databases using Java data sources. Before using either of these components, you must create resource references to the data sources in your project.

If you are using IBM Rational Application Developer to add resource references to the Feedback and LikeMinds data sources, complete the following steps:

1. From the WAR application WEB-INF/ directory, open the web.xml file. Click the References tab. Below the list of defined references, click Add.
2. In the Add Reference panel, select Resource Reference and click Next.
3. Complete the following fields in the Add Resource Reference panel and click Finish.
   - Name: jdbc/feedback
   - Type: javax.sql.DataSource
   - Authentication: Container
   - Sharing scope: Sharable
   - Description: This field is optional.
4. In the section marked WebSphere Bindings, enter the JNDI name of your Feedback Datasource.
5. Save the file.

Repeat this process for the LikeMinds data source. Change the name of the reference to jdbc/lmdbDS. All other settings remain the same.

Programming model

Personalization builds on the same programming model used by WebSphere Application Server. Two components of that model are Java Server Pages (JSP) and business logic (JavaBeans and enterprise beans). JSP files enable you to effectively separate HTML coding (presentation) from business logic. You isolate your business logic in beans that the Web page author embeds in the JSP.

The term developer could refer to Web page authors or programmers. The developer does not need programming skills to author JSPs that access databases and reusable Java components. The author only needs to know the type of content that the bean, servlet, or other Java component adds to the page. The programmers create the reusable Java components and provide the Web page authors with the component names and properties. The database administrators provide the Web page programmers with the database access and table information.

The model expands three Web team roles:

- The business users
The business users understand the business goals that the Web site or Web application must achieve. They understand the business conditions and business rules that must be factored into the Personalization solution. For example, the business users know the characteristics by which to categorize users for effective implementation of the business rules.

- The developer
The developer understands Web site development and can use Rational Application Developer (or similar tools) to create and publish Web pages and sites. The developer works with the business user to determine where personalized content should appear and creates content spots of the appropriate type at the proper locations on the Web pages.

- The administrator
This team member is the Web content expert and/or server administrator. The administrator knows what content already exists, where it is stored, the people who provide the content, how often the content is updated and other details. This person understands how to categorize content and helps the business users organize and target content for categories of users.

**User and content models**
The first step in developing a Personalization solution is to analyze your business conditions and events to determine the users and content to be targeted. The business users and administrators are primarily responsible for this task.

After this analysis is completed, focus on the user, content and other data for your Personalization solution. The subsequent tasks include:

- Developing a user model and a content model
The user model consists of the properties (attributes) of the user (that is, the typical Web site visitor), such as name and address. Similarly, the content model consists of properties of the Web content, such as title and author. Another term for model is schema.

  If you have already stored user data, some level of user model already exists. In such cases, the decision becomes whether to create a new model, or adapt the existing model to meet the requirements for your Personalization solution. You will also need to ensure that the user model and content model facilitate mapping users to content. For example, if you want to display news headlines that are of interest to the site visitor, the user model might include a property called user_interests that is a list of the topics in which the user is interested. The content model, for article content, could include a property topics that is a list of the subjects covered in the article. To support mapping a user interest to an article topic, you would ensure that there is an understandable relationship between possible values of user_interests and values of topics.

- Implementing the user and content models
You must implement the user and content data models. If database tables matching the data models do not exist, implementing the data model involves creating and populating the data store for the user and content data. If the database tables exist, you might need to combine data from multiple tables, or add columns to create a user or content resource that matches the corresponding data model. Reusing existing tables might involve combining data from multiple tables to create a user or content resource.

- Implementing the Personalization APIs for accessing user, content and other resources
After you implement the user and content data stores, you must enable the Personalization runtime environment to access the data stores. This task involves creating implementations of the Personalization APIs for accessing resources in the customer data store. You can either implement the `ResourceManager3` and `ResourceDomain3` interfaces manually or you can use the wizards provided with Rational Application Developer to generate resources for LDAP repositories. There are also implementations of the Portal user as a Personalization resource and a resource collection implementation for accessing the DB2 Content Manager.

**How the rules engine works**

The rules engine processes and delivers the results of a rule execution to a content spot contained in a Web page. The content spot is marked by a content spot bean which is placed in either a JSP file or a servlet. The JSP file or servlet is then linked to the Web page.

At run-time, the content spot bean searches for the best rule to fill the spot with personalized data. The best rule to use depends on the campaigns that have been set up to execute. The rule mapped to the content spot in the campaign with the highest priority will be executed to personalize your Web page.

Rule processing results in the return of a set of resources or a profiler. The returned resources or profiler can be used for generating a partial or entire Web page. The figure illustrates how a rule is processed.

**Step A:** The process begins when a user requests a Java Server Page (JSP) or servlet in which a content spot bean has been embedded. The content spot bean contains the code to find and execute the rule. When the Web server receives the client request for the JSP or servlet, the Web server passes the JSP or servlet request to the IBM WebSphere Application Server, which then invokes its JSP or servlet processor.
The content spot bean can be embedded in the JSP using any JSP editor. The following example code demonstrates embedding and using a content spot bean in a JSP.

**Note:** IBM Rational Application Developer provides a visual JSP editor (Page Designer) that simplifies the development task and generates the JSP scriptlet coding for you.

The bean is embedded using the JSP useBean tag. The HttpServletRequest object is passed to the bean within the body of the useBean tag. The content spot bean properties are retrieved in the same manner as retrieving properties for any JavaBean. The getRuleContent method of the contactsByLocation content spot bean determines the appropriate rule to execute based on campaigns having a rule mapped to the spot, executes the rule, and returns the results. The bean returns an array of Personnel objects.

**Table 144. Example code that demonstrates embedding and using a content spot bean in a JSP**

```jsp
<jsp:useBean id="contactsByLocation" class="GetContactsByLocation">
<% contactsByLocation.setRequest(request); %>
</jsp:useBean>
<%
try{
Personnel[] contentArray = contactsByLocation.getRuleContent();
Personnel theContent = contentArray[0]; // throws an exception if empty
%
<TABLE border="1">
<TBODY><TR>
<TD>Last Name</TD><TD>First Name</TD><TD>Role</TD><TD>Site</TD></TR>
<% for (int _i0 = 0;;){%>
<TR>
<TD><%= theContent.getLastName() %></TD>
<TD><%= theContent.getFirstName() %></TD>
<TD><%= theContent.getRole() %></TD>
<TD><%= theContent.getSite() %></TD>
</TR>
<% _i0++;
try{
theContent = contentArray[_i0];
} catch (java.lang.ArrayIndexOutOfBoundsException _e0) {
break;
} %>
</%>
</TBODY>
</TABLE>
<%
} %>
catch (java.lang.ArrayIndexOutOfBoundsException _e0) {
}%>
```

**Note:** When deciding to add personalized content to a Web page, all that is needed to develop the JSP is the content spot bean. The content spot can display personalized data of a single data type (for example, Personnel in this example).
The Web developer does not need to know where or how the content is retrieved, only that personalized content of type Personnel will be returned and it will have a set of properties to be displayed.

**Step B:** When the embedded content spot bean is invoked, the processor passes the HTTP servlet request object to the content spot bean. The client request is used to initialize the RequestContext. The RequestContext provides access to the resources needed for rule processing. Those resources include collections, application objects, requests, and sessions. The RequestContext is applicable for the life of the HTTP request.

**Step C:** The Personalization rules engine find the appropriate rules in the IBM Java Content Repository and executes them.

**Step D:** The rules engine processes the rule to obtain the results of the rules execution.

**Step E:** The rules engine returns the result of the rule execution to the content spot bean. The result can be a list of valid content from which a user can make a selection, a string, a boolean value, or no results. The JSP scriptlet or servlet uses the rule result to derive specific Web content for use in the generated Web page. The Web server forwards the page to the client.

**Workload management**

Only one Portal Personalization engine can be installed on an application server. Consequently, there is one Personalization engine for each servlet engine. Personalization supports application server clustering, because each Personalization instance in the cluster shares the same IBM Java Content Repository. Therefore, each Personalization instance accesses the same customer data stores.

IBM WebSphere Application Server dynamic caching is used to cache resource instance and the results of queries used in rules. The dynamic cache shares expiration notification for the cache across clones in a cluster. Although Personalization uses the dynamic cache internally to cache the results of rules, it is also possible to use the WebSphere Application Server dynamic cache to cache the entire response from a JSP or servlet.

**Note:** Care must be taken when using Personalization and the dynamic cache of JSPs, servlets, or portlets. When using the dynamic cache to cache JSPs or servlets, the cache key must take into account all the inputs into any rules on that page. If rules on the page use an employee department attribute of a user resource, the cache key must be configured to contain this employee department attribute.

**Using the Personalization APIs**

Personalization provides open APIs that enable the Personalization run-time environment and Rational Application Developer to access user and content data in customer data stores.

The Javadoc for Personalization APIs is available from the PortalServer_root/PortalServer/doc/Javadoc directory.

**Resource interface:**
The interface `com.ibm.websphere.personalization.resources.Resource` enables mapping your user model, content model, or other resource model to data in your customer data store. Get an overview of the methods defined by this interface that you must implement.

**Table 145. Explanation of methods used by `com.ibm.websphere.personalization.resources.Resource`**

<table>
<thead>
<tr>
<th>Method</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>getId()</code></td>
<td>Returns the primary key or identifier for this resource.</td>
</tr>
<tr>
<td></td>
<td>The primary key must be a string and unique within the resource collection.</td>
</tr>
<tr>
<td></td>
<td>This method behaves in coordination with the <code>findById</code> method of the</td>
</tr>
<tr>
<td></td>
<td>associated resource manager class such that the following method returns</td>
</tr>
<tr>
<td></td>
<td>true:</td>
</tr>
<tr>
<td></td>
<td><code>manager.findById(resource.getId(), context).getId().equals(resource.getId())</code></td>
</tr>
<tr>
<td><code>get(String name)</code></td>
<td>Returns the value of the specified dynamic property for this resource</td>
</tr>
<tr>
<td><code>keys()</code></td>
<td>Returns all (an Enumeration) of the dynamic property keys associated with</td>
</tr>
<tr>
<td></td>
<td>this resource.</td>
</tr>
<tr>
<td><code>put(String name, Object value)</code></td>
<td>Sets the specified dynamic property for this resource</td>
</tr>
<tr>
<td><code>remove(String name)</code></td>
<td>Removes the specified dynamic property</td>
</tr>
</tbody>
</table>

In addition to the methods listed in this table, your implementation must contain methods for setting and getting each fixed property in the data model. For example, if your user model includes a fixed property `userName`, you would define the methods `getUserName()` and `setUserName()`.

Given an implementation of fixed properties, dynamic properties are optional. The get, keys, put, and remove methods may be implemented to perform no operation. If the content schema or resource attributes are known when the Java classes are developed, fixed properties are preferred. If the attributes of a resource are not determined until the resource is instantiated in the application server, dynamic properties are preferred. Dynamic and fixed properties may be used together in a single resource.

Rules support nested method calls. For example, a resource interface implementation could define a `user` object with a fixed property `employer` for which there is a fixed property `name`.

**APIs for multivalue properties:**

Use the `com.ibm.websphere.personalization.resources.IMVResource` interface to enable mapping multi-value properties. Use the `com.ibm.websphere.personalization.resources.IMultiValueUtils` interface for retrieving multi-value properties when those resources are in a database.

The `com.ibm.websphere.personalization.resources.IMVResource` interface extends the `Resource` interface and enables mapping multi-value properties. Implementing this interface is only required when an `IMultiValueUtils` implementation is used by the `ResourceManager3`. This is the default case for resources generated with the Portal tools, but may not be necessary for custom resources.
Table 146. Explanation of methods for mapping multi-value properties

<table>
<thead>
<tr>
<th>Method</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>setMultiValueUtils(IMultiValueUtils instance)</td>
<td>Saves the reference to the instance of MultiValueUtils. That reference is used when invoking the fillinMultiValueProperties method of the MultiValueUtils class. This method does not return output.</td>
</tr>
<tr>
<td>addMultiValuePropertyValue(String propertyName, Object propertyvalue)</td>
<td>Enables the IMVResource instance to add values for multi-value properties. This method does not return output.</td>
</tr>
</tbody>
</table>

The interface com.ibm.websphere.personalization.resources.IMultiValueUtils is a set of utilities for retrieving multi-value properties when those resources are in a database. This class supports mapping multi-value properties to the corresponding database tables.

Table 147. Explanation of methods for retrieving multi-value properties

<table>
<thead>
<tr>
<th>Method</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>convertSQLtoMultiValue(String query)</td>
<td>Converts the SQL query string for ResourceManager3 classes that need to search on single value and multi-value properties. Returns an enumeration that contains the converted where clause followed by one or more elements that contain the table names that are involved in the query.</td>
</tr>
<tr>
<td>populateJoinedProperties(IMVResource theResource, RequestContext context)</td>
<td>Retrieves the value for all multi-value properties and calls the addMultiValuePropertyValue method of IMVResource. This method does not return output.</td>
</tr>
<tr>
<td>populateJoinedProperty(IMVResource theResource, String propertyName, RequestContext context)</td>
<td>Retrieves the value for all single value properties and calls the addMultiValuePropertyValue method of IMVResource. This method does not return output.</td>
</tr>
<tr>
<td>syncJoinedProperty(IMVResource theResource, String propertyName, List values, RequestContext context)</td>
<td>Populates the multi-value property into the resource repository.</td>
</tr>
</tbody>
</table>

General tips:

View some general tips related to resource classloading when using the Personalization API.

Resource classloading

- The recommended location for generated content spot classes and resource collection classes is in a WebSphere Application Server shared library. When a rule is executed, Personalization uses the class loader of the content spot to load any resource collections or application objects required for that rule. If you are using the default com.ibm.websphere.personalization.ContentSpot class to execute rules, then that class loader is the Personalization shared library class loader. In this scenario, your resource collections and application objects must be visible to the class loader of the com.ibm.websphere.personalization.ContentSpot class. With a generated
content spot class, you have the flexibility to place this class and any associated resource collections directly in a web module or portlet application, but they must also be available to the Personalization portlets. Whichever mechanism you use for content spots, the best way to ensure that your resource collection classes are available is to put them in a shared library on the application server. By default, Personalization is shipped with a shared library that may be used for resource collections. Place your resource collection classes in the \texttt{wp_profile_root}/PortalServer/pzn/collections directory and reload the server.

- In previous versions of Personalization, the user interface classes were loaded dynamically out of other web applications, out of databases, and out of other paths entered by the user. In the current version of Personalization, the content spot and resource collections must be on the classpath for the Personalization portlets. The best way to achieve this is through use of a shared library.

- If you use the Personalization resource wizard in Rational Application Developer to create the resource classes in a Web or portlet project, those generated classes will be deployed in the Web module and not be available to the Personalization portlets. You will have to put these classes on a shared library and make that shared library available to the Personalization portlets.

- Classes in a shared library are only reloaded when the application server is reloaded. Classes in a Web module are reloaded when the Web module is reloaded.

- Ensure the global uniqueness of the class names by using package names.

- In a cluster of application servers, be sure to copy your content spot and resource classes to each application server in the cluster.

**Personalization jar files that use public API:**

Learn about the \texttt{.jar} files that contain public APIs documented in Personalization JavaDoc information. Unless they are included in the JavaDoc documentation, classes are not public API even if they are included in these \texttt{.jar} files. When you use IBM Rational Application Developer and the Portal Personalization tools, these classes are added to your project classpath automatically and no further action is necessary.

If you are compiling your application outside of Rational Application Developer and you are using Personalization APIs, you may need to add the following classes to your classpath when compiling your application. These \texttt{.jar} files are located in \texttt{PortalServer_root}/pzn/prereq.pzn/lib:

- \texttt{pznquery.jar}
- \texttt{pznruntime.jar}
- \texttt{pznwpsruntime.jar}

**Generic query framework**

The generic query framework enables resource collection developers to convert a property-based query object into a language specific executable query string. It contains query component classes, and builder and callback interfaces.

- Query framework classes, such as the Query class, Predicate class, and Condition class provide an object representation of a query.
- Builder and callback interfaces together facilitate a query string generation mechanism that delegates operations to domain specific callbacks for property-to-attribute resolution and query string syntax conversion.
Using the Generic Query Framework:

The resource engine constructs a generic query object and passes it to domain developers through the ResourceDomain3 interface method findResourceByQuery(). Get an overview of how this query object can be converted into a meaningful domain query string.

The developer can take one of the following approaches to convert this query object into a meaningful domain query string.

1. Walk through the query object.

   **Note:** The com.ibm.websphere.query.base.Query class contains query components that the developer can cover to generate domain specific query string. For detailed information on query hierarchy and components, see the Portal Personalization API documentation.

2. Use a system provided builder callback.

   There are nine builder callbacks:
   a. Microsoft SQL Server Enterprise Edition:
      (com.ibm.websphere.query.callbacks.SqlSelectQueryCallback)
      **Note:** Although the generic SQL callback can be used for most SQL Server databases, there are minor differences in SQL syntax and availability of functions which require specific subclasses for some databases.
   b. IBM DB2 Universal Database Enterprise Server Edition:
      (com.ibm.websphere.query.callbacks.DB2SqlSelectQueryCallback)
   c. IBM DB2 Universal Database for z/OS:
      (com.ibm.websphere.query.callbacks.DB2390SqlSelectQueryCallback)
   d. IBM DB2 Universal Database for i:
      (com.ibm.websphere.query.callbacks.DB2400SqlSelectQueryCallback)
   e. Apache Derby:
      (com.ibm.websphere.query.callbacks.DerbySqlSelectQueryCallback)
   f. Oracle Enterprise Edition:
      (com.ibm.websphere.query.callbacks.OracleSelectQueryCallback)
   g. LDAP: (com.ibm.websphere.query.callbacks.LdapSelectQueryCallback)

   **Note:** The LDAP callback supports a set of function common to many LDAP repositories. Users may subclass this callback to support more advanced vendor specific functions.

   Property resolution and query syntax conversion are handled in the callbacks. The developer can prepare a property mapping hash table and use it with one of the previous callbacks to build the executable query string. Here is the sample code for SQL query string generation:

   ```java
   String s=q.buildString(new SqlSelectQueryCallback(h));
   ```

   where `q` is a query object, and `h` is a property mapping hash table.

   For detailed information on these builder callbacks, see the Personalization API documentation.

3. Develop and use a domain specific builder callback.

   If the system provided builder callbacks do not satisfy resource domain requirements, a domain specific builder callback can be created and used as long as it implements ISelectQueryCallback. The developer can decide the
mechanisms to interpret properties and derive the proper query syntax in their own callbacks. The code would look like:

```java
String s=q.buildString(new MySelectQueryCallback(myParameter));
```

where q is a query object, and MySelectQueryCallback is the custom builder callback that takes myParameter as parameter.

**Request context interface**

This is the interface used to access various attributes for rules. For HTTP contexts, it provides access to the HttpServletRequest and HttpSession attributes. For non-HTTP contexts, it provides the same interface to a surrogate for the request and session.

The request context, and any request values accessed via the request context, are only valid for the life of the request.

The Request Context string is used in caching lookups. The lookup is created by using a user-specified string combined with query values as the lookup key. The user-specified string should uniquely represent the current Request Context. This key is stored under ibm.wcp.cache.user.key as a request attribute or as a session attribute with the request attribute taking precedence.

**Accessing the Request Context**

The Request Context provides the Personalization rules engine with the data and environmental information needed for rules processing. In other words, the Request Context contains all the input to execute Personalization content spots and rules. This includes simple inputs like request and session data, and more advanced input like the user object.

You can access the Request Context from a content spot by first using the setRequest method of the content spot to back the content spot with a request, and then by calling getContext to retrieve the context. You can also use the Request Context to call directly into the ResourceDomain3 and ResourceManager3 APIs.

The Request Context allows you to retrieve session, request, portlet attribute, date, cookie, and other data and environmental information from the resource layer.

The Request Context includes:

- **Session**
  The session information identifies the HttpSession object that is associated with the current user.

- **Request date**
  This request date is the date the HTTP request was received. This information supports rules that have date-dependent actions.

Since Personalization uses the Request Context to contain all the rule input, the Request Context must be set onto the content spot prior to rule execution. The code with the content spot's useBean tag must be similar to:

```java
<jsp:useBean id="gold_promo_bean" class="yourco.goldpromo.BannerSpot" />
<% gold_promo_bean.setRequest (request); %>
```

In the previous section, the jsp:useBean tag constructs the yourco.goldpromo.BannerSpot class and stores an instance of that class into the local variable gold_promo_bean. The next line calls setRequest to put the
HttpServletRequest or PortletRequest onto the newly constructed content spot bean. The content spot then implicitly constructs a Request Context which is backed by the given HttpServletRequest or PortletRequest. This Request Context then provides access to that request’s parameters and attributes and attributes of the session through the com.ibm.websphere.personalization RequestContext interface.

In some cases, it may be useful to call into a content spot without having access to an HttpServletRequest or PortletRequest. The interface com.ibm.websphere.personalization.PznRequestObjectInterface can be used in these situations. A implementation of this class called com.ibm.websphere.personalization.PznRequestObjectImplementor is provided for convenience.

Sample Personalization resources XML file

Use the sample file ExportFromServlet.xml as a reference for coding other Personalization actions.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ibm-websphere-personalization>
<ResourceCollection action="create" name="News">
  <ResourceType>Content</ResourceType>
  <ResourceManagerClass>com.mycompany.personalization.ContentResourceManager</ResourceManagerClass>
  <ResourceClass>com.mycompany.personalization.Content</ResourceClass>
</ResourceCollection>
<SecurityMappings>
  <collection EnableSecurity="false" name="News"></collection>
</SecurityMappings>
</ibm-websphere-personalization>
```

Content spot exits

Content spot exits provide the ability to alter the default flow of processing of content spots, making it possible in the runtime environment to override the rule to process, the current user, and the results of rule processing. Instances of the same exit class are instantiated for all content spots. The exit class interface name is public interface RuleExit.

The content spot exit class can provide the following actions:

1. Access request and session information
2. Set information in the personalization context, including changing the user
3. Get the campaign name
4. Get the rule name to be executed for campaign
5. Override and specify the rule to be executed
6. Bypass rule processing; no rule is processed.

After rule processing, the content spot exit class can provide the following actions:

1. Add result items
2. Remove result items
3. Completely replace result items

RuleExit methods:

```java
void aboutToExecuteRule(RuleTrigger contentSpot, RequestContext requestContext)
```

Get the results of the rule execution.
Object[] getFilteredResults(RuleTrigger contentSpot, RequestContext requestContext, Object[] originalResults)

Exit to allow for changing the results of the rule execution. originalResults: the original result array is supplied to the exit. filteredResults: the exit should return the original array if no changes are needed

RuleTrigger methods:

String getCampaignName()
Get the campaign name for the content spot.

String getRuleName()
Get the name of the rule to be executed in this spot. Can return null if the name is not yet established.

void setRuleName(String ruleName)
Set the name of the rule to be executed in this spot. If the name is set to null, no rule will be executed.

void setRuleExit(RuleExit ruleExit instance)
Set the rule exit for this particular content spot instance.

The RuleTrigger also supports a static method that contains an instance of RuleExit that is set up at initialization time. You can implement setRuleExit(RuleExit) on a per-content spot basis to override only the content spots that you specify.

The class name of the default RuleExit implementor is read from the PersonalizationService.properties file.

Example Usage Scenario

1. Specify a RuleExit class in PersonalizationService.properties, as shown:
   rulesEngine.defaultRuleExit=com.ibm.websphere.personalization.RuleExitSample
   ...

2. At startup time, the Personalization run-time environment creates an instance of that class (aRuleExit in this scenario) and caches it in a private RuleTrigger static method

3. As each content spot is triggered, the Personalization rule engine determines the rule name to be used

4. aRuleExit.aboutToExecuteRule() is invoked, passing the spot and the request context

5. aRuleExit has several options:
   - Access request context information (includes http request and session)
   - Get the campaign and rule names
   - Specify a request user ID
   - Change the rule name to be executed
   - Bypass the rule by setting the rule name to null

6. The rule is executed

7. aRuleExit.getFilteredResults() is invoked; the rule exit modifies results as required, then returns the updated set

8. The updated results are stored

Resource cache
Personalization uses the WebSphere Application Server Dynamic Cache service to cache the results of select rules and to cache the rules themselves in a
DistributedMap. When publishing, importing or saving rules, the cache is flushed automatically to ensure that the site is current.

**Configuration settings available for caching rules**

The PersonalizationService.properties file contains cache control settings. The following table summarizes the configuration settings available for caching.

Table 148. PersonalizationService.properties properties and descriptions. Each PersonalizationService property is listed by name and described.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rulesEngine.cache.enabled</td>
<td>Global setting to disable all caching. Setting to false will override any collections that are individually set to be cached.</td>
</tr>
<tr>
<td>rulesEngine.cache.jndiName</td>
<td>The Dynamic Cache DistributedMap to use, as specified by its JNDI name.</td>
</tr>
<tr>
<td>rulesEngine.cache.maxEnumSize</td>
<td>Although the DistributedMap may be configured in the WebSphere Integrated Solutions Console to limit the maximum number of entries in the cache, the collection of resources that is returned for each rule is counted as one entry in the cache. When a rule returns a very large number of results, caching the results may result in extra memory overhead. Therefore, when the result set of a rule exceeds the number specified in this property, it will not be cached.</td>
</tr>
<tr>
<td>rulesEngine.cache.timeout</td>
<td>The number of seconds before an entry in the cache expires.</td>
</tr>
<tr>
<td>rulesEngine.cache.priority</td>
<td>The priority of a cache entry, relative to other entries in the same DistributedMap.</td>
</tr>
<tr>
<td>rulesEngine.cache.enabled.collectionName</td>
<td>The same as rulesEngine.cache.enabled, except that this property applies to a specific resource collection. The variable collectionName is the fully qualified path to the resource collection in Personalization. For instance, the GeneralNews resource collection that is included in the sample folder called PznDemo has the collectionName of /PznDemo/GeneralNews.</td>
</tr>
<tr>
<td>rulesEngine.cache.jndiName.collectionName</td>
<td>The same as rulesEngine.cache.jndiName, except that this property applies to a specific resource collection.</td>
</tr>
<tr>
<td>rulesEngine.cache.maxEnumSize.collectionName</td>
<td>The same as rulesEngine.cache.maxEnumSize, except that this property applies to a specific resource collection.</td>
</tr>
<tr>
<td>rulesEngine.cache.timeout.collectionName</td>
<td>The same as rulesEngine.cache.timeout, except that this property applies to a specific resource collection.</td>
</tr>
<tr>
<td>rulesEngine.cache.priority.collectionName</td>
<td>The same as rulesEngine.cache.priority, except that this property applies to a specific resource collection.</td>
</tr>
</tbody>
</table>
Caching business rules for resources

You might need to programmatically flush the Personalization cache, for example when a resource is updated outside of Personalization rules through some other application. A programming interface is provided to flush the cache. Since the timeout interval for the cache can be specified in the properties file, in many cases it may be adequate to wait for the cache timeout before updates are seen. The class com.ibm.websphere.personalization.resources.cache.CacheManager can be used to invalidate the cache for a particular resource, a particular resource collection, or the entire cache. Personalization uses this class internally to flush the cache when updates occur.

Flushing the cache for a particular resource may require that all cached queries be flushed. Flushing the cache on a collection may flush the cache for all collections using the same dynamic cache map. When the application frequently flushes the cache for a particular collection, isolating that collection in its own cache map through the use of the ruleEngine.cache.jndiName.resourceCollectionName property will result in better cache utilization.

Caching occurs before any rule exits are called.

For more information about using the DistributedMap and DistributedObjectCache interfaces for the dynamic cache, refer to the IBM WebSphere Application Server Information Center.

Related tasks:

Using the DistributedMap and DistributedObjectCache interfaces for the dynamic cache

Programmatically invoking rules

All types of rules can be accessed programmatically within a Java application. For example, a profiler can be used to determine the behavior an application should exhibit depending on the current user, or an action can return content to your application for further processing before the content is displayed. Rules are mapped to content spots, and because a content spot is an implementation of a JavaBean, it can be programmatically declared and implemented.

To programmatically invoke a rule, follow these steps:

1. Instantiate the bean. If the class name of your content spot was ProfilerSpot, you would instantiate the bean using one of the following options:

   ```java
   com.ibm.websphere.personalization.ContentSpot contentSpot =
   new com.ibm.websphere.personalization.ContentSpot("ProfilerSpot");
   ```

   or

   ```java
   ProfilerSpot contentSpot = new ProfilerSpot();
   ```

2. Call the method setRequest() and pass the HttpServletRequest object or an object that implements PznRequestObjectInterface. This makes current information stored in the request object known to the Personalization Runtime Engine and the Resource Engine.

   ```java
   contentSpot.setRequest(request);
   ```

3. Trigger the rule and, if applicable, get the content from the rule. Use one of the following methods, depending on the type of rule:

   - Select content actions, bindings and recommendation actions are mapped to content spots and return content from a resource.
     - getRuleContext() - returns an array of results
- getRuleContent(int which) - returns the resource at the given index

- String[] getProfiles() - returns an array of profiles.
  - For example, if a user is in profiles "young," "hip," and "sporty," this method will return an array of the profiles.

- boolean isProfiledAsAll(String[] profiles) - returns True or False depending on whether all the profiles match all the profiles in the list of profiles passed to the method

- boolean isProfiledAsAny(String[] profiles) - returns True or False depending on whether the profile is in the list of profiles passed to the method

- boolean isProfiledAs(string value) - returns true or false depending on whether the string passed to the method matches a profile given to the user

- String getProfile(integer value) - returns the profile in the location specified by the integer passed
  - A user may have more than one profile. For example, a user may fit profiles named "young," "hip," and "sporty"; when looking for the profile at location 0, "young" is returned.

- boolean isProfiledAsEmpty(int which) - returns true or false depending on whether the string passed to the method matches a profile given to the user.

- String[] getProfiles() - returns an array of profiles.
  - For example, if a user is in profiles "young," "hip," and "sporty," this method will return an array of the profiles.

- boolean isProfiledAsAll(String[] profiles) - returns True or False depending on whether all the profiles match all the profiles in the list of profiles passed to the method

- boolean isProfiledAsAny(String[] profiles) - returns True or False depending on whether the profile is in the list of profiles passed to the method

- Profilers are mapped to empty content spots which do not declare a return type.

- Update actions and email actions are mapped to empty content spots.
  - a. trigger() - runs the rule

**Rule Exception Handling in the run-time environment**

The default method of error or exception handling within the Personalization run-time environment is for the engine to print out a trace error to the application server's stdout log. Using an exception handling utility, it is possible to specify the type of output (error message or stacktrace), the output log file to use (stdout or stderr), and whether the exception should be rethrown to the JSP. The additional exception handling capabilities can be set on a per-request basis or globally.

**Per-request**

To set the scheme on a per-request basis, include the following code within a content spot on a JSP:

```java
request.setAttribute(RuntimeUtils.PZN_RUNTIME_EXCEPTION_HANDLING_KEY, RuntimeUtils.HANDLING_SCHEME);
```

Where HANDLING_SCHEME is one of the options specified for the setRuntimeExceptionHandlingScheme method described in the following section.

This method will only change the exception handling scheme for rules running on that JSP.

**Global**

To set the same scheme for every rule running on a server, execute the following code:

```java
RuntimeUtils.setRuntimeExceptionHandlingScheme(RuntimeUtils.HANDLING_SCHEME);
```

To reset the global exception handling scheme, call:

```java
RuntimeUtils.resetRuntimeExceptionHandlingScheme();
```
Note: Restarting the application server will automatically reset the exception handling scheme.

You can also change the global behavior of the server by changing the following value in PersonalizationService.properties:

rulesEngine.exceptionHandling=logMessage_stdout

Note: To make any changes effective, you must restart the server.

RuntimeUtils methods

public static void setRuntimeExceptionHandlingScheme(String value)
Sets the exceptions handling scheme to the specified value. Options for value include:
• RuntimeUtils.IGNORE
• RuntimeUtils.LOG_MESSAGE_STDOUT
• RuntimeUtils.LOG_MESSAGE_STDERR
• RuntimeUtils.LOG_MESSAGE_STDOUT_AND_RETHROW
• RuntimeUtils.LOG_MESSAGE_STDERR_AND_RETHROW
• RuntimeUtils.LOG_STACKTRACE_STDOUT
• RuntimeUtils.LOG_STACKTRACE_STDERR
• RuntimeUtils.LOG_STACKTRACE_STDOUT_AND_RETHROW
• RuntimeUtils.LOG_STACKTRACE_STDERR_AND_RETHROW
• RuntimeUtils.LOG_MESSAGE_AND_STACKTRACE_STDOUT
• RuntimeUtils.LOG_MESSAGE_AND_STACKTRACE_STDERR
• RuntimeUtils.LOG_MESSAGE_AND_STACKTRACE_STDOUT_AND_RETHROW
• RuntimeUtils.LOG_MESSAGE_AND_STACKTRACE_STDERR_AND_RETHROW
• RuntimeUtils.RETHROW_EXCEPTION

Note: Settings with RETHROW will pass the exception to the screen and are recommended for use within a testing environment only.

public static void resetRuntimeExceptionHandlingScheme()
Resets the current exception handling scheme to LOG_MESSAGE_STDOUT.

public static String getRuntimeExceptionHandlingScheme()
Returns the current exception handling scheme.

Exception Handling Process

When an exception occurs, if the code can continue, it will do so while logging the exception. If not, the exception is wrapped in a PersonalizationException which is passed to RuleTrigger. RuleTrigger looks for a request override for the exception handling scheme, processes the scheme, and returns to the JSP. Any subclasses of Throwable might be wrapped in a PersonalizationException.

Tracing

To trace the runtime classes, enable trace for com.ibm.websphere.personalization.**=all. To trace the authoring portlet classes, enable trace for com.ibm.wps.caf.**=all.
Chapter 6. Delivering web content

The type of delivery method you use to deliver web content to your viewers will depend on the type of content being delivered, and the type of viewers your website is intended for.

Accessing web content via a servlet

Users can access content displayed via the Web Content Manager servlet by connecting to a URL. A servlet delivered website should be used when you don't need to use any WebSphere Portal based features such as authoring tools.

Accessing a web page using a servlet

The following URL structure is used to connect to a web page:

http://[HOST]:[PORT]/wps/wcm/connect/[PATH]?srv=

Non-ascii characters:

Non-ascii characters can not be used in the query string section of URLs. For this reason, it is best not to name Web Content Manager items using Non-ascii characters if you plan to use URLs to call Web Content Manager items.

- [PATH] can be the path to a site area or content item. This must be entered for all types of content including components. In the case of components, this is the path to the site area or content item that the component is displayed with.
- srv= is either cmpnt or page.

<table>
<thead>
<tr>
<th>Service option</th>
<th>Details</th>
</tr>
</thead>
</table>
| srv=cmpnt     | This will retrieve a component either from the component library or from a site area or content item. You must also specify the following:  
  source=  
  This determines where the component is being sourced from. This is either:  
  - library  
  - sitearea  
  - content  
  cmpntname=[componentname]  
  This is the name of the component being retrieved. |
Table 149. Service options (continued)

<table>
<thead>
<tr>
<th>Service option</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>srv=page</td>
<td>This will retrieve a content item. As \texttt{srv=page} is returned as default, this can be omitted from the URL. The presentation template to use when displaying this content is specified by adding: \texttt{presentationtemplate=library/presentationtemplatename}</td>
</tr>
</tbody>
</table>

Examples:

URL to content:

\texttt{http://[HOST]:[PORT]/wps/wcm/connect/[PATH]}

Example: \texttt{http://host:10039/wps/wcm/connect/sitearea/content}

URL to content with a presentation template defined:

\texttt{http://[HOST]:[PORT]/wps/wcm/connect/[PATH]}

\texttt{?presentationtemplate=[libraryname/presentationtemplatename]}

Example: \texttt{http://host:10039/wps/wcm/connect/sitearea/content?presentationtemplate=library/presentationtemplatename}

URL to a library component:

\texttt{http://[HOST]:[PORT]/wps/wcm/connect/[PATH]}

\texttt{?srv=cmpnt&source=library&cmpntname=[componentname]}

Example: \texttt{http://host:10039/wps/wcm/connect/sitearea/content?srv=cmpnt&source=library&cmpntname=component}

URL to a content component:

\texttt{http://[HOST]:[PORT]/wps/wcm/connect/[PATH]}

\texttt{?srv=cmpnt&source=content&cmpntname=[componentname]}

Example: \texttt{http://host:10039/wps/wcm/connect/sitearea/content?srv=cmpnt&source=content&cmpntname=component}

Applying Custom Caching and Expiring Parameters.

Like any other URL request made to a Web Content Manager Server, Custom Caching and Expiring parameters can be added to a request. See the topic, “Using Custom Caching” for further information.

Example:

\texttt{http://[HOST]:[PORT]/wps/wcm/connect/[PATH]?CACHE=SITE&EXPIRES=REL+9000s}

In this example, the content being retrieved via URL will be saved in the Basic Site Cache, and expired after 9000 seconds (two and half hours).

Delivering web content on a portal page

Using tools like web content viewer portlets, content associations, and web content page templates, you can build portal pages and display web content. You can also combine web content with other portlet-based content. Content associations tie viewers and portal pages to the site structure of your web content libraries.

You can customize your portal delivery as well. Examples include:

- Creating your own web content page templates
• Deploying preconfigured web content viewers
• Providing convenience features like customized error messages and friendly URLs

Getting started with delivering web content on a portal page

The building blocks for delivering web content in a portal are web content viewers, web content page templates, and content mappings. These pieces provide a flexible framework that you can use to quickly assemble pages.

To help you get started, sample web content is also included in preinstalled libraries. The sample content demonstrates how the pieces work, and you can also adapt the sample content for your own use.

Web content viewers

Web content viewers are portlets that render content from a web content library as part of a portal page. If your presentation is simple, a single viewer can be sufficient. To provide a richer experience for your users, use multiple viewers to aggregate content from different libraries.

How viewers locate content

When you add a web content viewer to a web content page, the viewer locates the content to be rendered by evaluating several pieces of information:

• The portlet configuration settings for the viewer can identify the default content to be rendered when a user navigates to a page containing the viewer.
• If the viewer does not specify default content, it determines whether a default content association is defined for the page. If a content association exists, the viewer renders the default content of the referenced site area in IBM Web Content Manager.
• If the current request contains a public render parameter (path-info or context), the viewer renders the content identified by the render parameter. This setting overrides the content setting from the portlet configuration or from the content association on the page. An example of a case where a render parameter might be involved is when users click a link to a content item.

When determining which content to render, a web content viewer checks first for a render parameter on the request. As shown in Table 1, if no parameter exists, the viewer evaluates its own portlet configuration and any content association on the page containing the viewer.

Table 150. How web content viewers determine which content to render

<table>
<thead>
<tr>
<th>Content reference in portlet configuration?</th>
<th>Content association on web content page?</th>
<th>Content rendered by viewer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes</td>
<td>No</td>
<td>Content identified by portlet configuration of the viewer.</td>
</tr>
<tr>
<td>No</td>
<td>Yes</td>
<td>Content identified by association on page.</td>
</tr>
</tbody>
</table>
Table 150. How web content viewers determine which content to render (continued)

<table>
<thead>
<tr>
<th>Content reference in portlet configuration?</th>
<th>Content association on web content page?</th>
<th>Content rendered by viewer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes</td>
<td>Yes</td>
<td>Content identified by portlet configuration of the viewer. The content specified in the portlet configuration can use a relative path. In this case, the complete path to the target content is derived by combining the associations on the page and the viewer.</td>
</tr>
</tbody>
</table>

References to content can be direct or relative:

**Direct path to the target content**

A web content viewer can reference either a site area or a specific item in the library. This example is a reference to the item Article in the site area Articles of the library Web Content:

\[
\text{Web Content/Articles/Article}
\]

When you configure the viewer with a direct path to content, the content association on the page that contains the viewer is ignored.

To configure a direct path to content, use the **Select content and path** setting in the **Content behavior** settings of the web content viewer.

**Relative path to the target content**

When deriving a content path based on a relative path, the viewer appends the content path in its configuration to the content association on the page. For example:

- Content association on web content page: Web Content/Articles
- Content referenced in portlet configuration: Article
- Resolved content path: Web Content/Articles/Article

To configure a relative path to content, use the **Select content and use the content association of current page** setting in the **Content behavior** settings of the web content viewer.

Depending on how you reference content, you can create web content viewers and web content pages that are as specific or generic as you require.

**Reference no content**

You can define a content association on a web content page and then add a generic viewer that references no content. The viewer detects the site area defined by the content association on the page and renders content from the site area. No additional configuration of the web content viewer is required.

**Reference specific content**

You can configure a viewer to point to a specific piece of content. You can then add this viewer to any web content page to render the mapped content, regardless of any content association on the page.

**Reference content with a relative path**

If you are using a consistent site structure with your web content libraries,
you can take advantage of the relative path capability for referencing content. For example, you can create a reusable viewer that can render different content depending on the web content page where the viewer is deployed. By defining a viewer with a content association that uses a relative path, you can add instances of that viewer to different pages. The viewers then render different content, according to the content associations on the pages.

You can use one content association on a page and then add multiple web content viewers on the page. In this case, the web content viewers are configured to use a relative path into different site areas in the library. By changing only the content association on the page, you can then redirect the viewers to another library or other set of site area content.

Create content with web content viewers

When added to a page, a viewer can create a copy of the content that is identified in the portlet configuration. This feature provides several advantages:

- You can create content items quickly and easily, within the scope of the page where they are used.
- You can modify the individual copies of the content independently of each other.
- Typically, there is no need to further adjust the configuration of the viewer after you add it to the page.

Web content viewers configured to create content can be used multiple times, either on the same page or on different pages. Each instance of the viewer references a separate copy of the content item referenced in the portlet configuration.

Copied content is stored in the site area identified by the default content association of the current page. In addition, the portlet configuration of the newly added viewer is automatically updated to specify a relative path to the copied content.

Link web content viewers

Many web content viewers can be added to a single portal page or a series of pages. Sometimes it is necessary for different web content viewers to interact with each other. For example, a menu component might be placed in one viewer and a content item in another viewer. If you want the content item to change when a different link is clicked in the menu, you must link the two viewers.

Web content viewers can broadcast or receive links:

Broadcasting links

The state or context of a web content viewer is not sent directly from one portlet to another. You can configure viewers to broadcast their current state or context to other viewers on the same page or to viewers on other pages. Any information broadcast by a web content viewer is received only by viewers that are configured to receive this information.

Receiving links

A web content viewer can receive the following information:

- Information about the state or context of the current content item or component being rendered by the viewer.
- Information from content items or components rendered by other viewers that are broadcasting links.
For examples of the different ways that you can use linking with web content viewers, see Link examples for web content viewers.

**Web content viewers and remote servers**

To display web content on a portal that does not include Web Content Manager, you can use the web content viewer and the WSRP support in the portal. The web content viewer can then retrieve and display content from a web content system on a different server.

**Related concepts:**
- Working with web content pages
  
  A web content page is a portal page that displays web content from a site or site area by mapping the page to the web content site structure in your Web Content Manager system. When you add the web content viewer to a web content page, the portlet automatically renders the default content of the attached site or site area. In addition to displaying the default content, the web content viewer provides a dynamic broadcasting option that selects the best matching web content page when selecting links to other content items.

**Related tasks:**
- “Performing remote rendering with WSRP and the web content viewer” on page 565
  
  To display web content on a portal that does not include IBM Web Content Manager, you can use the web content viewer and the WSRP support in the portal. The web content viewer can then retrieve and display content from a web content system on a different server.

**Related reference:**
- “Link examples for web content viewers” on page 545
  
  Web content viewers can broadcast and receive links to communicate with other viewers. Depending on the link settings that you use with the viewers, the behavior of the viewers can be different. These examples demonstrate how different broadcast and receive settings can affect what a viewer renders.

**Web content pages and templates**

Web content pages are portal pages that are associated with content that is managed in IBM Web Content Manager. Similar to web content pages, web content templates are page templates that are associated with content in Web Content Manager.

Pages and page templates are tied to web content by content associations. These associations are defined in the page properties and can be specified with the Associations window when editing page properties. You can associate a page with one or more site areas in one or more web content libraries.

If managed pages are enabled, all managed pages are automatically associated with a corresponding page site area in the Portal Site library. This type of association is called a system content association and enables changes to the page to be managed by Web Content Manager. System content associations are managed by the portal and cannot be deleted or changed manually.

When there are multiple content associations for a page, one of the associations is designated as the default association. Typically, the default association is the system content association for the page. However, you can use the Associations window to specify a different content association as the default association.
Web content pages

With web content pages, you can take advantage of the following benefits when rendering web content:

- If you add an unconfigured web content viewer to the page, the viewer automatically renders the default content of the site area specified by the default content association.
- Dynamic page selection determines the best web content page to use to render a content item when clicking a link to the content item. For example, if you click a link to a content item in a search result, the portal evaluates the following set of content associations:
  - Associations that exist for the site area that directly contains the target content item.
  - Associations for any site areas that are ancestors of the site area that contains the target content item.

The portal identifies the page that is mapped to the site area that is closest to the target content item. The viewer then renders the content item on that page.
- You can extend friendly URLs to reference content items rendered on a web content page. Friendly URLs for web content are composed by combining the friendly URL of the current page and the content path of the rendered content item.

You can create a web content page in two ways:
- Create the page from a web content page template.
- Add a content association to an existing portal page.

Web content page templates

To create a web content page template, create a page under the Page Templates label. This label is the root label for all page templates in the portal. You can access the Page Templates label in the administration interface in either of the following locations:

- Administration > Portal User Interface > Page Templates
- Administration > Portal User Interface > Manage Pages > Content Root > Hidden Pages

When you create a web content page template, you define the layout, style, and contents of any web content page that is created from the template. Web content page templates have all the same flexibility and customization features as a standard portal page or portal page template. You can perform common tasks like adding content with portlets, changing the style of the page, or changing the layout of objects on the page. By using viewers with other portlets in a web content page template, you can create pages that support a wide range of user goals. Likewise, you can rely on only viewers and create a website that is primarily composed of information in your web content system.

As with standard page templates, you can create web content pages from a web content page template:
- Manage Pages administration portlet
- Create Page window in the site toolbar
When users create a web content page from a template, content that is associated with the page template is copied with the page itself. If you create multiple pages from the same page template, each page results in a separate copy of that content.

When you create a page from a template, page titles in any language are not copied. The following elements are copied to the new page:

- Portlet entities, including portlet preferences
- Page layout and style
- Theme and skin settings
- Portlet wires for communication with other portlets
- Page parameters
- Page description (all languages)

In addition, the following changes take place automatically, depending on the individual web content associations that exist on the page.

- If the page template is a managed page, with a system content association that references the Portal site library, the following changes apply:
  - A portal page site area is created in the Portal Site library, with the title of the site area being derived from the title of the new page. The hierarchy structure of the portal page site area is automatically synchronized with the page hierarchy in the portal.
  - All authoring template mappings and all nested content are copied over into the new portal page site area. However, any nested portal page site areas are not copied.

- If there is a default content association that references a library other than the Portal Site library, the following changes apply. These changes apply regardless of whether the page template is a managed page.
  - A site area is created, with the title of the site area being derived from the following elements:
    - The title of the new page.
    - The name of the site area that is being derived from the friendly URL name of the page.
  - The site area is created as a child of the site area that is defined as the default content association of the parent page of the new page. This support requires that the parent page is associated with a site area outside the Portal Site library.
  - All site area properties and all nested content are copied over to the new site area.
  - The default content association on the new page is modified to reference the newly created site area.

Web content viewers on the page template can be configured to reference content that is copied when a page is created from the template. When the page is created, the viewer configuration is automatically adjusted to point to the new content that is created during page instantiation.

**Note:** Managed pages must be enabled to support page templates that store their associated web content in the Portal Site library. If you disable managed pages, the content that is associated with a template is no longer copied during page instantiation. In addition, the corresponding preferences of any web content viewers that are on the page are not adjusted.
Hierarchical page templates

You create hierarchical page templates by creating child pages under your page templates.

When creating a page from a child page template, the following conditions apply:
- Like the top-level page templates, these child pages are displayed in the list of available page templates.
- Page template instantiation always creates a single page, regardless of other child pages that might exist under the page template.
- Child pages of page templates can be instantiated individually under the page that was instantiated with the parent page template.
- If a child page has a default content association to web content that it is outside the Portal Site library, that content is not copied during instantiation. The content is not copied because the content was already copied during instantiation of the parent page template.
- If the default content association on the child page references a site area that is contained in the site area that is associated with the parent page template, the new page is updated to reference the corresponding nested site area that is created during instantiation of the parent page.
- You can configure web content viewers on a child page to reference content that is copied when a page is created from the parent page template. When the page is created from the child page of a page template, the viewer configuration is automatically adjusted to reference the new content that is created during instantiation of the parent page.

Related tasks:
“Creating a web content page” on page 548

A web content page is a page that is associated to one or more site areas in IBM Web Content Manager. You can create a web content page from a web content template page, or you can convert an existing portal page into a web content page.

Web content associations
Web content associations map portal pages to the site structure in the IBM Web Content Manager system. You can define a default content association and multiple other associations, which are used for dynamic page resolution. For each page, there is also a system content association that maps the page to its corresponding portal page site area in the Portal Site library.

Each web content association consists of a reference to a portal page and a reference to a site area in a web content library. When a page contains a web content association, web content viewers added to the page can automatically render the content provided by the associated site area. In addition, a web content page template containing an association can create copies of associated content when you create a page by using the template.

When multiple associations are defined for the same web content page, one of those associations is identified as the default content association. When you create a page, a system content association is automatically created to the corresponding portal page site area in the Portal Site library. This system content association is designated as the default association, but you can change that setting later as needed.

The default content association has several uses:
When you add a web content viewer to the page without configuring the viewer to reference content, it renders the content indicated by the default association. When creating a page from a web content page template, the default content association of the page template indicates the site area to be copied during page instantiation. If you are building a friendly URL to content on the page, the default association indicates the path to the rendered content. This content path fragment is appended to the friendly URL of the current page to generate the complete friendly URL.

All content associations for a page are used for dynamic page resolution. The portal uses dynamic page resolution to determine the best matching page for rendering a specific content item.

You can also configure each web content association to enable or disable page-based access control when rendering content from the mapped site area. With this feature, users who are authorized to view the page are also assumed to have view access for content under the associated site area.

Web content associations are managed by the content mapping service of the portal. You can manipulate web content associations with the following methods:

- The Page Associations window in the site toolbar.
- The page properties available from the Manage Pages portlet.
- XML configuration interface, using the xmlaccess command
- Portal Scripting Interface
- REST API for content associations
- Public Java API for the content mapping service

**Page context and user context**

The content associations of a page can define the initial web content context of the page. This context is used for rendering when users first access a page. The context of the page can change when users interact with the content on the page. Each web content viewer on the page can be configured with an explicit context that overrides the rendering page context. The context of the portlet can also change if it is configured to receive links. When users click a link within the viewer that is configured to broadcast its links, the page context is updated. This new context is maintained until users click another link on this page or until users start a new session. When users start a new session, the original page context is used.
Related tasks:

“Enabling page-based access control for web content pages” on page 558

Typically, when you render content items in a web content viewer, access control enforcement on those content items is handled by IBM Web Content Manager. However, you can use page-based access control to delegate access control enforcement to the web content page that is used to display the content.

Related reference:

“Content associations reference” on page 567

Content associations are used to associate web content pages with your web content site structure. When you select a folder to associate with a web content page, a content association is created and maintained within the portal.

“XML configuration interface and content associations” on page 568

With the XML configuration interface (xmlaccess command), you can perform batch updates of content associations or export associations to import into another portal. Content association information is represented in the XML configuration schema by content-mapping-info elements.

“Portal Scripting Interface and content associations” on page 569

With the Portal Scripting Interface, you can create scripts to automate the management of content associations. Using the ContentMapping bean with the Portal Scripting Interface, you can add, modify, and remove content associations.

“REST API and content associations” on page 572

If you are creating or extending an application and want to manage content associations with that application, you can use portal remote APIs. These APIs retrieve a list of content associations and then create, update, or delete associations.

Related information:

Java API for Content Mapping Service

Creating content with sample web content template items

To illustrate how page templates, web content viewers, and content associations work together, IBM Web Content Manager provides sample web content. The sample content includes examples of web content template pages and predefined portlets that you can add to pages to render content.

In addition to examining how these rendering pieces are defined, you can also explore the web content libraries containing the sample content.

Note: For a typical installation using the Full option, the sample web content template items are deployed by default. However, if you installed the portal using the Base options, you can add the items after installation, as described in Adding templating sample content to a base installation.

Related information:

Adding sample web content template items to a base installation

Adding sample content with the site toolbar:

The sample content provided with the portal includes four preconfigured web content viewer portlets that you can add to a page from the site toolbar. The viewers are configured with the Create content (based on selection) setting so that adding the viewers to a page automatically creates content items on the page. The viewers are available in the site toolbar in the Web Content category of the Content tab.
The Article viewer presents a simple article composed of a title, a short description, and a rich-text body. The default presentation template that is used to render individual articles includes the following elements:
- Inline widgets to support tagging and rating
- Additional site analytics data that is added to the generated markup

The List of Articles viewer presents a list of all articles that exist in the context of the current page. The viewer also enables users to create articles.

The Rich Text viewer presents a simple rich text element.

The Image viewer presents a simple image.

To use these sample web content viewers, complete the following steps:
1. Navigate to the page where you want to add the viewers.
2. Edit the page, and click Content in the site toolbar.
3. Click Web Content in the list of categories. This category lists all content viewer portlets that are configured with the Create content (based on selection) setting to copy associated content into the page.
4. Add the web content viewers to your page by clicking the plus sign (+) or dragging the viewers onto the page. You can also adjust the arrangement of the portlets.
5. Click Save & Exit.

When you add one of these web content viewers to a page, new content is created. The content items that are referenced in the portlet configuration are copied into the site area that is specified by the default content association. The portlet configuration of each viewer instance is automatically adjusted to reference the copied content items, so that the viewers render the new content.

To help you better understand what these components are doing, here is a more detailed look.

**Article**
The Article viewer is a copy of the web content viewer that is customized to render specific content. This approach is common for building web content viewers.

The Article viewer has several notable features:

- The viewer uses a customized portlet title (Article) instead of the default title to provide contextual information about the content rendered by the viewer. When you are assembling pages and adding web content viewers, taking advantage of details like customized titles is important for orienting users.

- The viewer is configured to render this content: Template Page Content/Articles/Article. The Template Page Content library is preinstalled with the portal and includes sample content for content templating. In this case, the viewer is referencing the Article content item in the Articles site area. You can see the content reference in the content settings of the Configure mode of the portlet.

- The viewer is configured to create new content when an instance of the viewer is added to a page. This configuration is specified with the Create content (based on selection) setting in the portlet configuration. Because of this setting, whenever the Article viewer is added to a page, the referenced content item is copied to the site area associated with the page. This site area is identified by the default content association. The viewer can be added multiple times, either on the same page or on different pages. Each instance of the viewer references the dedicated copy of the base content item that was created when the viewer was added to the page.

- When an instance of the viewer is created, the Content field in the shared settings of the viewer is rewritten to point to the newly created Article item.

- To ensure that the content item to be rendered can be passed in by other viewers, the viewer is configured to receive links with the setting Other portlet and this portlet. For example, if you click a link in an instance of the List of Articles viewer on the same page, this viewer renders the linked content.

- The Article viewer also defines preferences that enable the generation of additional metadata for the page, based on the currently rendered article instance. This metadata includes description, keywords, and author.

- The default presentation template that is used to render articles (Web Content Templates/Article) includes an authoring tools component. The component enables users to modify the content of individual articles with inline editing. The authoring tools component is hidden automatically when the page is not in edit mode.
List of Articles

The List of Articles viewer provides a content item that has the following features:

- The content item references a menu in Web Content Manager that renders a list of all articles that are associated with the current page.
- The List of Articles viewer does not contribute page metadata or a page title. This behavior occurs because the viewer renders a list of items instead of specific content that would determine the semantic content of the page.
- Because the viewer is rendering a list of items, the viewer is not listening to links that are broadcast from other portlets.
- The default presentation template that is used to render lists of articles (Web Content Templates/List of Article) includes an authoring tools component. The component enables users to create an article with inline editing. The authoring tools component is hidden automatically when the page is not in edit mode.

The List of Articles viewer has several similarities to the New Article viewer:

- The viewer references base content from the Template Page Content library.
- The viewer is configured to create copies of its referenced content when an instance of the viewer is added to a page.
- The viewer is configured to broadcast links with the Dynamically select a web content page setting. This configuration results in the List of Articles viewer determining the content that is rendered by other viewers on the same page, such as the Articles viewer.

Rich Text

The Rich Text viewer provides a content item that consists of the name, which is not rendered by default, and a rich text body. The default presentation template that is used to render rich text (Web Content Templates/Rich Text) includes an authoring tools component. The component enables users to modify the content of the rich text item with inline editing. The authoring tools component is hidden automatically when the page is not in edit mode.
The Image viewer provides a content item that consists of the name, which is not rendered by default, and a reference to an image. The default presentation template that is used to render images (Web Content Templates/Image) includes an authoring tools component. The component enables users to modify the image reference with inline editing. The authoring tools component is hidden automatically when the page is not in edit mode.

Related tasks:

"Creating a web content page” on page 548

A web content page is a page that is associated to one or more site areas in IBM Web Content Manager. You can create a web content page from a web content template page, or you can convert an existing portal page into a web content page.

Creating content with the Articles template page:

The sample content provided with IBM Web Content Manager includes a web content template page called Articles that demonstrates how you can build pages with predefined content.

The Articles template page is a managed page that contains two web content viewers. These viewers are configured to render content items that are stored in the site area for the page in the Portal Site library.

To create a page based on the Articles template, complete the following steps:
1. Navigate to the page where you want to add the new page.
2. Edit the page, click More, and then select either New Child Page or New Sibling Page.
3. Use the Create Page window to create the page. Select the Articles template for the new page.

The new page contains the List of Articles web content viewer and the Article viewer:

The Articles template demonstrates several key features of web content page templates:
• The Articles template page contains a system content association to the site area for the template page in the Portal Site library (Portal Site/Content/Content Root/Hidden Pages/Page Templates/Articles). This site area contains three content items that are used as the initial content for pages created from the page template. To see this web content association, edit the page, and then in the site toolbar, click edit in the Web Content field of the Overview tab.
• There are two web content viewers included with the Articles template:
  – The List of Articles viewer renders links to all articles stored in page site area.

Chapter 6. Delivering web content 543
- The Article viewer renders a specific article that is selected in the List of Articles viewer. The viewer also contributes page metadata to the HTML head section of the rendered portal page.

- The page uses a dedicated theme module (wp_oob_sample_styles) to include the CSS styles that are required by the rendered content items. To ensure that this theme module is available to pages that are created from the page template, the template explicitly references the full theme profile. This profile contains the wp_oob_sample_styles theme module. The reference to the full theme profile is specified in the page metadata by the resourceaggregation.profile parameter, with a value of profiles/profile_full.json. When a page is created from the template, this parameter is copied, with all page parameters, to the new page instance.

For more information about theme modules and theme profiles, see Modules.

When you create a page based on the Articles template, several things happen:

- A page is created based on the page template. The new page has the same layout and style as the template and contains copies of the two web content viewers.
- The new page has a system content association to its corresponding portal page site area.
- The content items contained in the site area for the page template are copied into the site area associated with the new page.
- The Articles viewer and List of Articles viewer on the new page automatically render the newly created content that is associated with the new page. The rendering occurs because of the viewer configuration:
  - The configuration of the List of Articles viewer is updated during instantiation to point to the newly generated List of Articles content item.
  - The Article viewer is configured to directly retrieve its context from the content association of the containing page.

Note: The Articles page template is a managed page that stores its associated web content in the Portal Site library. If you disable managed pages, the content that is associated with this template is no longer copied during page instantiation. In addition, the corresponding preferences of the Articles viewer and List of Articles viewer are not adjusted.

Related information:
- The module framework

CSS styles used by the sample web content template items:

The markup that is generated by the sample web content template items is primarily controlled by presentation templates that are stored in the Web Content Templates library. These templates rely on the availability of several CSS class definitions in the wp_oob_sample_styles theme module.

Any pages that render the sample items must use a theme profile that includes the wp_oob_sample_styles theme module or another module that contains the same CSS class definitions. An example of such a theme profile is the full theme profile that is installed by default.

The wp_oob_sample_styles theme module includes a single CSS file that is named oob_samples.css. This stylesheet is loaded from the path common_resources_root/ibm/css/samples. The common_resources_root node is specified in the WebSphere Integrated Solutions Console by the resources.commonResourcesRootURI property.
of the WP GlobalThemeConfig resource environment provider. The default value for this property is dav:fs-type1/common-resources. This value causes the CSS styles to be loaded from the portal file store with the following path: /common-resources/ibm/css/samples/oob_samples.css. For more information about setting the common_resources_root node, see Adapt the list of required runtime configuration changes for your theme.

**Stylesheet note:** The Rich Text presentation template that is used as the default presentation for Rich Text content items uses CSS styles that render differently depending on the layout container that contains the Rich Text web content viewer. Different classification CSS classes are assigned to these layout containers. As a result, the font sizes of headings and body elements are reduced when a rich text item is moved from the center column into a more narrow side column.

**Related information:**
Adapt the list of required runtime configuration changes for your theme

**Adding the sample web content libraries in the authoring portlet:**

The templating sample content provided with IBM WebSphere Portal is delivered in two web content libraries: Template Page Content and Web Content Templates. You can use these libraries and their content as a starting point for working with web content page templates and developing your own templates.

**Web Content Templates library**
This library contains the shared components, authoring templates, and presentation templates that are used by the content items contained in the Template Page Content library. The library also contains the sample content items that are used with the "Articles" page template.

**Template Page Content library**
This library contains the content items that are used by web content viewer clones that create content when added to a page. These content items represent the base content that is copied when you add content from the Web Content category of the site toolbar to a page. Initially, this library contains two content items. The Article content item represents a simple article created from the "Articles" authoring template in the Web Content Templates library. The List of Articles content item is an instance of the "List of items" authoring template in the Web Content Templates library. The item represents a list of all articles that exist in the context of the currently rendered page.

**Virtual portal note:** If you want to use the sample content with a specific virtual portal, you must syndicate these web content libraries to the virtual portal. If you fail to syndicate these libraries, an error is displayed when you add the sample content to a page.

To work with the sample web content libraries, complete the following steps:
1. Open the authoring portlet for Web Content Manager by clicking Applications > Content > Web Content Management.
2. In the authoring portlet, click Preferences > Configure.
3. In the Library Selection section, add the Template Page Content and Web Content Templates libraries to the list of selected libraries.

**Link examples for web content viewers**
Web content viewers can broadcast and receive links to communicate with other viewers. Depending on the link settings that you use with the viewers, the
behavior of the viewers can be different. These examples demonstrate how
different broadcast and receive settings can affect what a viewer renders.

**Example 1: Configuring links for a single portlet**

In this scenario, a single web content viewer is configured to show a dynamic
component, such as a menu. The menu contains links to content in a site area.
When you click a link, the viewer continues to render the component, but the
component uses the target content to set the web content context. Based on the
new context, the component generates appropriate markup. In this example of a
menu, the component shows links to content of the site area where the target
content resides.

<table>
<thead>
<tr>
<th>Web content viewer content</th>
<th>Broadcast links to</th>
<th>Receive links from</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component</td>
<td>None</td>
<td>This web content viewer</td>
</tr>
</tbody>
</table>

When adding a component to a web content viewer, you can also select an
alternative presentation template in the content view of the content section. When
a link is clicked in the component, the content that is then rendered uses the
alternate presentation template.

**Example 2: Configuring links for a menu and content**

In this scenario, one web content viewer contains a Web Content Manager menu
and another web content viewer contains Web Content Manager content. Links
must be created between the two viewers to enable the rendered content to change
when different links in the menu are selected.

<table>
<thead>
<tr>
<th>Web content viewer content</th>
<th>Broadcast links to</th>
<th>Receive links from</th>
</tr>
</thead>
<tbody>
<tr>
<td>Menu</td>
<td>This page</td>
<td>None</td>
</tr>
<tr>
<td>Content</td>
<td>None</td>
<td>Other web content viewers and this viewer</td>
</tr>
</tbody>
</table>

**Example 3: Configuring links for a navigator and content**

In this scenario, one web content viewer contains an item views navigator and
another viewer contains some Web Content Manager content. Links must be
created between the two viewers to enable the rendered content to change when
different links in the navigator are selected. The navigator also changes to reflect
the current state of the content being rendered.

<table>
<thead>
<tr>
<th>Web content viewer content</th>
<th>Broadcast links to</th>
<th>Receive links from</th>
</tr>
</thead>
<tbody>
<tr>
<td>Navigator</td>
<td>This page</td>
<td>Other web content viewers and this viewer</td>
</tr>
<tr>
<td>Content</td>
<td>This page</td>
<td>Other web content viewers and this viewer</td>
</tr>
</tbody>
</table>
Example 4: Configuring dynamic links for a navigator and web content pages

In this scenario, one web content viewer contains a site views navigator, and several web content pages that are associated with different site areas. Each web content page contains a viewer and is configured to show the default content item of the site area that is associated with the page.

Instead of manually creating links between the different viewers, you can use dynamic link broadcasting with the viewer. Dynamic link broadcasting automatically determines which web content page is used as the target for the links to the site area in the navigator.

Table 154. Example 4: Configuring dynamic links for a navigator and web content pages

<table>
<thead>
<tr>
<th>Web content viewer content</th>
<th>Broadcast links to</th>
<th>Receive links from</th>
</tr>
</thead>
<tbody>
<tr>
<td>Navigator</td>
<td>Dynamically select a web content page</td>
<td>None</td>
</tr>
<tr>
<td>Content on web content page</td>
<td>This page</td>
<td>Other web content viewers and this viewer</td>
</tr>
</tbody>
</table>

Related concepts:
[“Web content viewers” on page 531](#)

Web content viewers are portlets that render content from a web content library as part of a portal page. If your presentation is simple, a single viewer can be sufficient. To provide a richer experience for your users, use multiple viewers to aggregate content from different libraries.

Displaying content with web content viewers

Display content from your web content system by adding a web content viewer to the server where you want the content to show.

If your presentation is simple, a single web content viewer can be sufficient. You can also use multiple web content viewers to provide a richer experience for your users.

Depending on how you decide to deploy the servers in your environment, there are different ways to render content:
- You can install a web content viewer locally on the same portal server where Web Content Manager is installed.
- You can install the viewer remotely on a different portal server.

Adding a web content viewer portlet

Add a web content viewer portlet to a page with the site toolbar.

1. Navigate to the page where you want to add the web content viewer.
2. Edit the page, and click Content in the site toolbar.
3. Click All in the list of categories, and search for the portlet Web Content Viewer (JSR 286).
4. Add the web content viewer to your page by clicking the plus sign (+) or dragging the viewer onto the page. You can also adjust the style and layout of the portlet on the page.
5. After saving your changes, you can modify settings for the web content viewer, such as a content association or portlet title.
Creating a web content page

A web content page is a page that is associated to one or more site areas in IBM Web Content Manager. You can create a web content page from a web content template page, or you can convert an existing portal page into a web content page.

A web content page always restricts the sharing scope for public rendering parameters to the page itself. The corresponding page parameters are automatically added to the page when a page is associated to a site area.

Caching note: When using web content pages, you cannot use advanced web content caching but instead can use only the portlet fragment cache.

1. There are two ways to create a web content page.
   - Follow these steps to create a web content page from a web content template page:
     a. Navigate to the page where you want to add the new page, and edit the page.
     b. Click More > New Sibling Page and use the Create Page window to create the page. Select a web content page template for the new page.

       Note: If managed pages are enabled, all page templates result in web content pages.

     c. Click Create Page to create a web content page from the selected template.

       If the page template has a default content association, the new page is automatically associated with a new copy of the referenced site area. Any other content associations that are not designated as the default association are copied without changes.

   - Follow these steps to convert a portal page to a web content page:
     a. Navigate to the page where you want to add the new page, and edit the page.
     b. In the Overview tab, click Edit in the Web Content section of the page properties.
     c. In the Page Associations window, click Add web content.
     d. Select one or more site areas that you want to associate with the page, and click OK.
     e. Specify the default association by selecting the association in the Default column.

2. After creating the web content page, you can add web content viewers to the page from the Content tab of the site toolbar.
   You can add a standard Web Content Viewer (JSR 286) portlet, or you can add any predefined web content viewers that you create.

   - If you add a standard web content viewer, the viewer renders any content from the site area indicated by the default content association of the page.
   - If you add a predefined viewer that is configured to create content with the Create content (based on selection) setting, the following things happen:
     -- The base content referenced by the viewer is copied into the site area that is referenced by the default content association of the page.
     -- The new instance of the viewer renders the copied content in the site area that is referenced by the default content association of the page.

   An example of this type of predefined viewer is the Rich Text viewer that is available from the Web Content category of the Content tab.
Related concepts:

“Web content pages and templates” on page 534

Web content pages are portal pages that are associated with content that is managed in IBM Web Content Manager. Similar to web content pages, web content templates are page templates that are associated with content in Web Content Manager.

Related tasks:

“Enabling page-based access control for web content pages” on page 558

Typically, when you render content items in a web content viewer, access control enforcement on those content items is handled by IBM Web Content Manager. However, you can use page-based access control to delegate access control enforcement to the web content page that is used to display the content.

Customizing web content delivery

Although web content viewers and page templates provide the basis for delivering web content, you can customize the environment to provide users with a better experience.

For example, you can create custom objects like page templates and web content viewers. You can also provide users with conveniences like customized error messages and friendly URLs that reference web content.

Creating web content page templates

Create web content page templates to quickly deploy new pages containing web content. With a template, you can define the layout and presentation of the page, including adding web content viewers configured to render web content.

When a page is created from the template, the site structure required in the web content library is also created automatically. For example, if managed pages are enabled, a site area that corresponds to the page hierarchy is created in the Portal Site library. If managed pages are disabled, a site area is created as a child of the site area that is associated with the parent page of the new page. You can access the resulting site structure in the Web Content Manager authoring portlet.

1. Using the administration interface, navigate to the page templates location. You can access this location from either of the following paths:
   - Click Portal User Interface > Page Templates.
   - Click Portal User Interface > Manage Pages > Content Root > Hidden Pages > Page Templates.

2. Create a page. When you create the page, use the Basic template and then define a content association for the page. The content contained in the mapped site area is copied to any page that is created from this page template.

3. Edit the layout of the page and add any web content viewers or other portlets that you want to include in the template.

   If you add a viewer that is intended to render content that is copied when the page is created, then configure the viewer:
   - Select Edit Shared Settings.
   - Select Select content and use the content association of current page from the Content behavior section. This setting causes the viewer to reference content from the site area defined in the content association on the page containing the viewer.
Adding predefined web content to the customization shelf

To make web content easier to find, the customization shelf provides categories to organize components that you can add, such as portlets, iWidgets, and web content. By default, the Web Content category includes four sample content items, but you can also add your own content to the Web Content category.

To include a web content item in the Web Content category of the customization shelf, complete the following steps.

1. Using the administration interface, make a copy of the portlet called Web Content Viewer (JSR 286). Give the copy a name that indicates the content item that the viewer represents.
2. Add the new web content viewer to a page. This page is only required temporarily to provide a way of configuring the web content viewer.
3. Open the Configure mode for the portlet.
4. Select the content item that you want to add in the Content Type section.
5. Select Select content and path in the Content Behavior section.
6. Select Create content (based on selection) in the section called When this Portlet is added to the page.

Note: Portlets that use this setting cannot be added to a page that does not have a default content association.
7. In the Content section, specify the content item or site area that this viewer represents.

After you complete these steps, the copy of the web content viewer representing your web content is displayed in the Web Content category of the customization shelf. Whenever you add this portlet copy to a page, the associated content item or site area is copied into the site area that is associated to the page by the default content association.

XML configuration interface note: You can also use the XML configuration interface (xmlaccess command) to create the copies of the web content viewer. To specify the behavior of the Create content (based on selection) setting with the xmlaccess command, specify this portlet preference in the XML import file:

- Preference: com.ibm.portal.wcm.copy.contents
- Value: true

Customizing error messages for web content viewers

If an error occurs during rendering, the web content viewer shows an error screen. You can customize the default error screen, and you can create your own custom JSP file that is used to display error messages.

The default error screen provides a standard error message, which is shown on every type of error, and a more detailed error message. The detailed error message provides information about the cause of the error and is displayed when you click the View details link.

1. Create a customized error JSP file.
a. Copy the original error.jsp file from `wp_profile_root/installedApps/node_name/PA_WCMRingPortJSR286.ear/ilwwcm-localrende.war/jsp/html` directory to create your custom error JSP file.

Almost everything in the original JSP file can be changed according to your requirements. If you want to show the cause of the error, you must retain this part of the original file in your error JSP file:

```jsp
<%@-- use errorbean from request --%>
<jsp:useBean id="errorbean" scope="request"
    type="com.ibm.portal.portletui.messages.StatusMessageBean" />
<% String msg = errorbean.getMessage(); %>
```

The variable `msg` contains the message text of the error. In the original error.jsp file, this message is only shown in a separate window if a user selects the View details link.

2. Configure the web content viewer to use the customized error JSP file.
   a. Log in to the portal as an administrator.
   b. Navigate to Administration > Portlet Management > Portlets.
   c. Locate the Web Content Viewer (JSR 286) portlet.
   d. Click Configure Portlet.
   e. Edit the value of the ERROR_JSP parameter, and set the path to your customized error JSP file as the parameter value.

Storing JSP files: You can store JSP files in one of two locations:

- Within the `wp_profile_root/installedApps/node_name/PA_WCMRingPortJSR286.ear/ilwwcm-localrende.war` directory of your server. When storing JSP files in this directory, enter the path to your custom error JSP relative to the directory.
- Within any other web application running on the portal. When referencing JSP files in another web application, use the following path: `contextPath;jspPath`. For example: `/wps/customapplication;/jsp/error.jsp`.

Friendly URLs and web content viewers

Friendly URLs provide a way for you to define a custom address for a portal page that is easy to remember and share. The web content viewer expands on friendly URL support by enabling you to specify additional path information in the friendly URL.

By defining friendly URLs for your portal pages, users can navigate the portal using more concise URLs that better reflect the page structure. The additional path information in a friendly URL for web content points to a content item to be displayed in the web content viewer.

Related tasks:

“Creating a content URL generation filter class” on page 659
A content URL generation filter is used to customize the URLs that are generated by a web content viewer. By creating a plug-in that implements a content URL generation filter, you can tailor the URLs to content items.

About friendly URLs for web content:

With friendly URLs for web content, you can construct URLs to content items that are clear and concise. Although you can construct friendly URLs that reference web content items, IBM Web Content Manager itself does not generate friendly URLs.
by default. However, to cause the web content viewer to generate friendly URLs, you can create a plug-in that implements a content URL generation filter.

These URLs are easier for users to remember and share and are a convenient way for users to create bookmarks to content items. External applications can also use friendly URLs to provide links directly to content items in the portal. To create effective friendly URLs for web content, you must understand how friendly URLs for portal pages are constructed and how friendly URLs for web content extend those URLs.

**How friendly URLs for pages are constructed**

For a page to be referenced as part of a friendly URL, you must assign a friendly URL name for the page. You can assign a friendly URL when you create the page, or you can edit the page properties after the page is created.

Friendly URLs take the following general form:

http://host_name:port_number/context_root/portal/page_id/[!ut/p/encoded_suffix]

The `page_id` portion of the friendly URL is made up of the friendly URL names of each page in the page structure. The page structure begins at the content root and ends with the currently selected page.

For example, you might have a portal page called Products with a friendly URL name of `products`. Under the Products page is another page called Appliances with a friendly URL name of `appliances`. When referenced as a complete friendly URL, you would enter the following URL to access the Appliances page:

http://www.example.com:10039/wps/portal/products/appliances

For friendly URLs to work for a specific page, you must define a friendly URL name for each page or label in the page structure. If you want to suppress a friendly URL name from showing in the friendly URL, you can specify a friendly URL name of `com.ibm.portal.friendly wildcard` for the page. For example, if the Products page has a friendly URL name of `com.ibm.portal.friendly wildcard`, the friendly URL in the previous example for the Appliances page is abbreviated:

http://www.example.com:10039/wps/portal/appliances

**Note:** When the portal displays a page using a friendly URL, the URL can include an encoded suffix at the end of the URL with the form `!ut/p/base_codec/rich_state`. This suffix contains information about the portal’s state that the portal might use when displaying the page. However, when bookmarking or sharing friendly URLs, it is not necessary to include the suffix.

**How friendly URLs for web content are constructed**

Friendly URLs for web content are constructed just as friendly URLs for pages but include additional information that identifies the path to a content item. When the portal decodes a friendly URL, it decodes the URL from beginning to end. Each path segment of the URL is matched with the friendly URL names of portal pages until no more matches can be located. The remainder of the URL is then considered to be path information to a content item.

This path information is mapped to a shared public render parameter that is scoped to the portal page identified by the URL. The fully qualified name of this `path-info` parameter is http://www.ibm.com/xmlns/prod/websphere/portal/
The **path-info** parameter can contain multiple values, with the individual values representing segments of a content path. The segments are concatenated using a forward slash (/) as a path separator.

Friendly URLs for web content take the following general form:

```
http://host_name:port_number/context_root/portal/page_id/path_to_content/!/ut/p/encoded_suffix
```

When you add a web content viewer to a portal page, the web content viewer reads the **path-info** parameter. The viewer assembles the path to the content to be rendered by appending the path information to the content mapping defined for the current page. For example, you might have the following friendly URL for web content:

```
http://www.example.com:10039/wps/portal/products/appliances/welcome
```

Several conditions contribute to this URL:

- The portal page **Products** has a friendly URL name of **products**, and underneath the **Products** page is another page called **Appliances** with a friendly URL name of **appliances**.
- A web content library contains a site area called **Appliances**, which contains a content item called **welcome**. For this example, the web content library is called **Web Content**.
- The portal page **Appliances** contains a content mapping to the **Web Content/Appliances** site area.

When a web content viewer is added to the **Appliances** page, the web content viewer interprets the **path-info** information from the friendly URL. The viewer identifies **welcome** as path information that represents content in a web content library. By examining the content mapping on the page, the web content viewer locates the **Web Content/Appliances** site area and then displays the **welcome** content item.

The **page_id** portion of the friendly URL is always evaluated first. Because of this priority, ensure that your naming schemes do not overlap when setting up your portal page hierarchy and your web content hierarchy. In particular, the **path_to_content** information cannot begin with segments that could be part of the **page_id** portion of the friendly URL. If the first segment of the **path_to_content** information matches the friendly URL name of a portal page at that point in the page hierarchy, the friendly URL could reference the wrong page.

**Considerations for the path-info parameter:**

- For a web content viewer to process the **path-info** parameter, the web content viewer must be configured to receive links. If it is configured to receive links, the web content viewer gives precedence to the **path-info** parameter over the **context** public render parameter. When you click links displayed by the web content viewer, the link automatically incorporates the path information for the linked item.
- Clicking **Clear page context** when editing the settings of a web content viewer also clears the **path-info** parameter.
- If a friendly URL includes an encoded suffix, it takes this form: `!/ut/p/base_codec/rich_state`. Because this information is encoded, it is not intended to be read by people. However, the portal itself might act on the information, which can sometimes cause the wrong page to be displayed.

If the **path-info** public shared render parameter is encoded in the **rich_state** portion of the suffix, the **path-info** contents overwrites the **path_to_content**
portion of the friendly URL. It is also possible that there could be a mismatch between the `path-info` contents and the path information encoded in the `rich_state` section. If such a mismatch occurs, the portal replaces the `path_to_content` portion of the friendly URL with the `rich_state` information and directs the user to that page.

The following tables demonstrate how the presence of `rich_state` information affects the page that is shown:

**Table 155. Example of rich_state information affecting displayed page**

<table>
<thead>
<tr>
<th>Description</th>
<th>URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>The user navigates to URL in the portal.</td>
<td><a href="http://www.example.com:10039/wps/portal/home/content_item_1/!ut/p/b1/dY07Do">http://www.example.com:10039/wps/portal/home/content_item_1/!ut/p/b1/dY07Do</a>...</td>
</tr>
<tr>
<td>The user modifies the URL in the browser's address bar to go to <code>content_item_2</code>.</td>
<td><a href="http://www.example.com:10039/wps/portal/home/content_item_2/!ut/p/b1/dY07Do">http://www.example.com:10039/wps/portal/home/content_item_2/!ut/p/b1/dY07Do</a>...</td>
</tr>
<tr>
<td>Resulting URL.</td>
<td><a href="http://www.example.com:10039/wps/portal/home/content_item_1/!ut/p/b1/dY07Do">http://www.example.com:10039/wps/portal/home/content_item_1/!ut/p/b1/dY07Do</a>...</td>
</tr>
</tbody>
</table>

Because the `rich_state` portion of the URL still contains path information pointing to `content_item_1`, the portal overwrites the `path_to_content` portion of the URL. The user remains on the same page instead of being directed to the page where `content_item_2` is displayed.

**Table 156. Example of friendly URL for web content without rich_state information**

<table>
<thead>
<tr>
<th>Description</th>
<th>URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>The user navigates to URL in the portal.</td>
<td><a href="http://www.example.com:10039/wps/portal/home/content_item_1/!ut/p/b1/dY07Do">http://www.example.com:10039/wps/portal/home/content_item_1/!ut/p/b1/dY07Do</a>...</td>
</tr>
<tr>
<td>The user modifies the URL in the browser's address bar to go to <code>content_item_2</code>.</td>
<td><a href="http://www.example.com:10039/wps/portal/home/content_item_2">http://www.example.com:10039/wps/portal/home/content_item_2</a></td>
</tr>
<tr>
<td>Resulting URL.</td>
<td><a href="http://www.example.com:10039/wps/portal/home/content_item_2">http://www.example.com:10039/wps/portal/home/content_item_2</a></td>
</tr>
</tbody>
</table>

Because the user removed the `rich_state` portion of the URL when modifying the URL, the `path_to_content` portion of the URL is evaluated. The user is directed to the page where `content_item_2` is displayed.

**Content URL generation filters and friendly URLs**

A content URL generation filter is used to customize the URLs that are generated by a web content viewer. The web content viewer generates a content URL whenever there is a URL to web content within content that the viewer is displaying. By creating a plug-in that implements a content URL generation filter, you can tailor the URLs to content items. For details, see Creating a content URL generation filter class.

**Troubleshooting friendly URLs for web content**

If you are seeing unexpected behavior when using friendly URLs for web content, review these issues to help identify why the friendly URL is not working. Friendly URLs for web content take the following general form:
Support for friendly URLs for web content is only enabled when the configuration properties friendly.enabled and friendly.pathinfo.enabled both have a value of true in the portal Configuration Service.

The web content viewer displays a warning message in the following situations:
- The friendly URL for web content references a content item that cannot be located.
- The user does not have sufficient access rights to view the referenced content item.

The portal page specified in the friendly URL for web content must contain a content mapping to an existing web content site area. If there is no content mapping on the page, any web content viewers on the page display a warning message about the missing page context.

If the target page does not contain a web content viewer that is configured to receive links, the content item specified in the friendly URL for web content is not displayed.

If a web content viewer is not configured to broadcast links, links rendered by the viewer do not affect the friendly URL for web content.

The default portal page selection does not show the path of the default content item in the friendly URL. The path_to_content portion of the URL includes the content path information only after users browse web content using links displayed by the viewer.

Friendly URLs for web content are URL-encoded. When using friendly URLs for web content, special characters that show in any segment of the URL must be URL-encoded. For example, a space character in the URL would be replaced by its URL-encoded equivalent: %20. Some web browsers perform automatic decoding of the URL. In this case, you might see unencoded characters in the URL, but the portal always works with an encoded version of the URL.

The segments of a friendly URL for web content are not localized for multiple languages. The path_to_content portion of a friendly URL for web content is composed of the unlocalized names of web content folders, site areas, and content items. For example, if you name these items with English terms, the friendly URL for web content is constructed of these English terms, even if the portal language is not English.

Friendly URL for web content example:

This example demonstrates how friendly URLs for web content work with multiple web content viewers on a single portal page. The example describes the portal page structure referenced by the friendly URLs and explains the underlying structure of the content in an IBM Web Content Manager site framework.

The example also describes the configuration of the web content viewers.

Elements in the example

The example is composed of the following elements:

Portal page hierarchy

The portal page hierarchy in this example is:

```
Content Root
  > Home
    > Human Resources
```
The page Home has a friendly URL name of home, and the page Human Resources has a friendly URL name of hr. The pages can be accessed directly using the following friendly URLs for pages:

- http://www.example.com:10039/wps/portal/home
- http://www.example.com:10039/wps/portal/home/hr

**Web Content Manager site framework**

The Web Content Manager site framework resembles the portal page hierarchy:

- Web Content (Web content library)
  - Home (site area)
    - Human Resources (site area)
      - HR Welcome (content item)
    - Health (site area)
      - Workplace Safety (content item)
      - Personal Wellness (content item)
    - HR Menu (menu component)

These content items can be referenced by the following content paths:

- Web Content/home/human resources/hr welcome
- Web Content/home/human resources/health/workplace safety
- Web Content/home/human resources/health/personal wellness

The menu component HR Menu is defined to display content from the human resources site area and the health site area.

**Content association**

The portal page Human Resources contains a content association to the Web Content/home/human resources site area.

**Web content viewers**

The page Human Resources contains two instances of the web content viewer, Web Content Viewer A and Web Content Viewer B.

- The Web Content Viewer A viewer renders the menu component HR Menu and is configured to broadcast links to this portal page.
- The Web Content Viewer B viewer inherits the content to display from the content association defined for the page Human Resources. The viewer is configured to receive links from other portlets and from itself.

**Browsing the example content**

With the portal page and web content site framework defined, browsing the content demonstrates how the different elements interact:

1. Navigate to Human Resources page for the first time.
   - The URL that is displayed in the address bar of the browser is http://www.example.com:10039/wps/portal/home/hr/!
   - The URL reflects the friendly URL names of the portal pages Home and Human Resources.
   - The Web Content Viewer A viewer renders the menu component and displays links to the content items HR Welcome, Workplace Safety, and Personal Wellness.
   - The Web Content Viewer B viewer shows the default content item HR Welcome from the site area Human Resources, because of the content association defined on the portal page.
Note: When the portal page is first displayed, the path of the default content item is not included in the friendly URL.

2. Click Workplace Safety from the list of content items.
   - The URL that is displayed in the address bar in the browser is http://www.example.com:10039/wps/portal/home/hr/health/workplace%20safety/!ut/p/b1/...
   - The Web Content Viewer B viewer displays content item Workplace Safety.
   - The URL is adjusted so that the path to the content item (health/workplace%20safety) becomes part of the URL.

3. Click HR Welcome from the list of content items.
   - The URL that is displayed in the address bar of the browser is http://www.example.com:10039/wps/portal/home/hr/hr%20welcome/!ut/p/b1/...
   - The Web Content Viewer B viewer displays the content item HR Welcome again, giving the same result as when the portal page was viewed for the first time.
   - Because the Web Content Viewer A viewer is broadcasting the link to the content item, the URL that is displayed in the browser is updated to reference the path to the content item (hr%20welcome).

Content item references with friendly URLs for web content

The URL displayed in the web browser can sometimes include the content item path when you browse pages and content with web content viewers. However, you can also reference content items directly in friendly URLs for web content.

For example, to reference the content items HR Welcome, Workplace Safety, and Personal Wellness in the context of the Human Resources page, you would use the following friendly URLs for web content:
   - http://www.example.com:10039/wps/portal/home/hr/hr%20welcome
   - http://www.example.com:10039/wps/portal/home/hr/health/workplace%20safety
   - http://www.example.com:10039/wps/portal/home/hr/health/personal%20wellness

Note: These friendly URLs for web content include URL-encoded space characters (%20) instead of unencoded space characters. Your web browser might accept unencoded space characters when specifying content item names in friendly URLs for web content. However, to ensure consistent behavior from the portal, use the URL-encoded value.

Setting up a web content fallback page

Set up a web content fallback page to be used when a web content viewer cannot determine which page to use to display a content item. The fallback page can also be used when users do not have sufficient privileges to view the page originally associated with the content item.

Note: Although a fallback page is one way of handling failed page resolution, you can also use a content page resolution filter to perform more advanced resolution. With these filters, you can tailor how the viewer behaves if no page can be found for a content item. For more information, see Creating a content page resolution filter class.

1. Create the portal page to be used as the web content fallback page.
   - Specify a unique name for the page so that you can reference the page later.
Assign any access rights required for users. For example, if the fallback page is available in the public part of the portal, ensure that anonymous users have view access to the page.

2. Add a web content viewer to the fallback page.
   a. Click the Edit Page Layout icon (small pencil) for the new page.
   b. Click Add portlets and select Web Content Viewer (JSR 286) from the list of portlets.

3. Update the WebSphere Portal configuration service to enable the fallback page.
   a. Log in to the WebSphere Integrated Solutions Console.
   b. Click Resources > Resource Environment > Resource Environment Providers.
   c. Click WP ConfigService.
   d. Under Additional Properties, click Custom Properties.
   e. Click New, and enter the property name wcm.fallback.page. Set the string value to the unique name or object ID of the portal page that you created as the fallback page.
   f. Save the changes to the master configuration.

4. Restart the portal.

The web content fallback page is displayed when a viewer is configured with a link broadcast setting of Dynamically select a web content page and one of the following conditions occurs:

- The web content viewer cannot determine which page to use to display the linked content item.
- The web content viewer identifies the page associated with the content item, but the user does not have sufficient privileges to view that page.

Related tasks:
- “Creating a page from the site toolbar” on page 128
- “Creating a content page resolution filter class” on page 652

A page displays content, such as portlets and other pages. Pages organize your site information. As you create pages, you also create new navigational elements to the site. You can create a page under an existing page or you can create a page that is a peer to an existing page. When you create a page you can also reference an existing page, apply a layout, and select supported markups.

Enabling page-based access control for web content pages

Typically, when you render content items in a web content viewer, access control enforcement on those content items is handled by IBM Web Content Manager. However, you can use page-based access control to delegate access control enforcement to the web content page that is used to display the content.

Page-based access control can be specified for individual content associations of a page. When page-based access control is enabled for a specific association, Web Content Manager does not perform additional access control checks when the associated site area contains the rendered content. This content includes not only direct children of the associated site area but also any nested site areas and their related content. Instead, the access control enforcement layer assumes that the same level of access that is granted on the page also applies to the rendered content.
content. The affected content also includes any content items or components that are loaded during the rendering of the target content item. As a result, rendering performance is improved. If the viewer renders content that is not contained in the associated site area, normal access control enforcement is performed.

Page-based access control affects only the access control enforcement that is triggered by the web content viewer. Other access paths, such as the access control that is used by the search crawler, are not affected. In addition, Web Content Manager resources like images that are loaded directly by the browser are also not affected.

**Context processor plug-ins note:** Context processor plug-ins used with the web content viewer run in the access control context of the initial page request. Changes made to the rendering context that might affect whether page-based access control is enabled are evaluated after all other context processor plug-ins complete their configuration.

1. To enable page-based access control, ensure that the web content page is associated with a site area or folder in the web content system. You can manage content associations using the site toolbar or the Page Properties portlet.

2. For each content association for which you want to enable page-based access control, select **Use Portal Page Security** for the association.

**Note:** If the user creating the page does not have sufficient permissions to enable page-based access control, the check box is disabled. The following access rights must be defined:

- Administrator @ `wcm_library`, where `wcm_library` represents the library containing the content that is associated to the web content page.
- Administrator @ `CONTENT_MAPPINGS`
- Editor @ `wcm_page`, where `wcm_page` represents the web content page for which you want to enable page-based access.

**Related tasks:**

- “Creating a web content page” on page 548

A web content page is a page that is associated to one or more site areas in IBM Web Content Manager. You can create a web content page from a web content template page, or you can convert an existing portal page into a web content page.

**Previewing content on web content pages**

When working with content in the authoring portlet, you can preview content items in a portal environment as part of a web content page. To preview content items, there must be a web content viewer on the associated web content page, and the viewer must be configured to receive links.

This preview shows the content item in a more accurate context, as it would be displayed to users. There are two ways to preview content items in a portal:

- You can use dynamic page selection. In this case, the portal determines the best match for the web content page to use for previewing the content item. The page selection is based on the content associations between the site area that contains the content item and the available web content pages.
- You can specify individual pages for previewing content.

1. Enable the preview function in the authoring portlet. This step must be done only once.
   a. Select either **Edit Shared Settings** or **Configure** from the menu in the title bar of the authoring portlet.
b. In the **Previewing Options** section, indicate the previewing method that you want to use.
   - To use dynamic page selection, select **Allow authors to preview content dynamically in a Web Content portal page**.
   - To preview content items on specific pages, select the pages to use for previewing from the list with the following heading: **Allow authors to preview content in the local portal pages selected below**.

2. Preview the content as a web content page.
   a. Select a content item in the authoring portlet.
   b. Click **Preview**, and then select **Preview as a Web Content portal page**. A separate browser window opens to show the content item on its associated web content page.

**Adding HTML meta tags for Search Engine Optimization**

Search engine optimization (SEO) focuses on improving the visibility of a page or website in search engine results. A basic technique of SEO is adding HTML title and meta tags to your page source. These meta tags are used to define keywords and other metadata that search engines and crawlers can use when creating search indexes and collections. When including content in a page with a web content viewer, you can improve the SEO of the page by adding title and meta tags with values derived from the web content itself.

**Note:** This support is available with cumulative fix 12 for IBM Web Content Manager Version 7.

By default, the HTML title for a page is defined by the page title in the portal. However, when you add a web content viewer to a page to render web content, you can override the value used for the HTML title. Web content viewers that are configured to override the HTML title can also add HTML meta tags as portlet preferences.

With the **Page Display Title** field in the portlet settings for the viewer, you can define an HTML title that better reflects the content on the page. You can even have the viewer pull the title directly from the rendered content.

**Note:** Although multiple web content viewers on the same page can set meta tag values, this practice does not necessarily result in improved SEO. This issue can be further complicated when multiple viewers set different values for the same meta tag name. When you have multiple viewers on the same page, select the viewer whose content best represents what the page is about. You can then use that viewer to define a new HTML title and any meta tags.

To override the HTML title for a page and set meta tags, complete the following steps.

1. Select one web content viewer to be the primary viewer on the page. Click **Edit Shared Settings**, and select a value for the **Page Display Title** field in the portlet settings for the viewer.
   
   To override the HTML title, you must select a value other than **Use default title**. If you want the title value to come directly from the web content being rendered by the viewer, select **Select from content**. This setting uses the value of the **Display title** field for the content item in Web Content Manager.

   After you save the changes, the page header is updated with the new title value. For example:
2. Create portlet preferences for each meta tag that you want to add to the page header. Each meta tag is defined by a pair of portlet preferences:

- meta.tag.name.suffix identifies the name of the meta tag (for example, keywords).
- meta.tag.content.suffix identifies the value of the meta tag.

You can also define a specific attribute for the meta tag with the following portal preference: meta.tag.attribute.suffix. The suffix portion of each preference is used to associate a name preference with its related value preference. The suffix can be any value as long as it is unique across the preferences.

There are two ways you can add portlet preferences:

- The Manage Pages portlet of the administration interface. Locate the instance of the web content viewer you want to modify, and select the Configure portlet icon.

- The XML configuration interface. Export the page containing the instance of the web content viewer you want to modify. Edit the exported XML file with the meta tags you want to add, and update the page by using the XML file along with the xmlaccess command.

If you do not set a portlet preference for the attribute name, the attribute name "name" is used by default.

a. Specify the portal preference for the name of the meta tag. The meta tag name takes the following format:

```
meta.tag.name.suffix=name
```

If you want to specify an attribute other than the name attribute, you can define the attribute name with the following format:

```
meta.tag.attribute.suffix=attribute_name
```

For example, to add the following meta tag with the name keywords:

```
<meta name="keywords" content=""/>
```

Specify the following preference:

```
meta.tag.name.1=keywords
```

To add the following meta tag with the http-equiv attribute:

```
<meta http-equiv="content-language" content="en-US"/>
```

Specify the following preference:

```
meta.tag.attribute.1=http-equiv
```

b. Specify the portal preference for the value of the meta tag. The value of the meta tag can be specified in three ways:

- You can explicitly enter text for the meta tag value.
- The meta tag value can be derived from the value of a text element in the rendered web content.
- The meta tag value can be derived from properties that contain information about the rendered web content.

Depending on how you want to specify the meta tag value, different portlet preferences are required. Only one value can be specified per suffix.
Use preset text
The meta tag value takes the following format:
\[\text{meta.tag.content.text.suffix=text}\]

The suffix portion must match the suffix value of the associated \text{meta.tag.name.suffix} preference. The text portion indicates the text to use for the meta tag value.

Use the value of an element
The meta tag value takes the following format:
\[\text{meta.tag.content.element.suffix=name_of_element}\]

The suffix portion must match the suffix value of the associated \text{meta.tag.name.suffix} preference. The name of element portion indicates the name of the element from the web content being rendered.

Table 157. Elements for populating meta tag values

<table>
<thead>
<tr>
<th>Element</th>
<th>Meta tag value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Text component</td>
<td>Text of the element</td>
</tr>
<tr>
<td>Date component</td>
<td>Date of the element</td>
</tr>
<tr>
<td>Image component</td>
<td>URL of the image</td>
</tr>
<tr>
<td>File component</td>
<td>URL of the file</td>
</tr>
</tbody>
</table>

Use a property
The meta tag value takes the following format:
\[\text{meta.tag.content.property.suffix=property}\]

The suffix portion must match the suffix value of the associated \text{meta.tag.name.suffix} preference. The property portion indicates the property containing information about the web content being rendered. The properties are associated with fields on the rendered content.

Table 158. Properties for populating meta tag values

<table>
<thead>
<tr>
<th>Property</th>
<th>Meta tag value</th>
</tr>
</thead>
<tbody>
<tr>
<td>AdditionalViewers</td>
<td>Name of additional viewers</td>
</tr>
<tr>
<td>Authors</td>
<td>Display names of the authors of the rendered content</td>
</tr>
<tr>
<td>authtemplatename</td>
<td>Name of the authoring template of the rendered content</td>
</tr>
<tr>
<td>authtemplatetitle</td>
<td>Display title of the authoring template of the rendered content</td>
</tr>
<tr>
<td>Categories</td>
<td>Titles of any categories associated with the rendered content</td>
</tr>
<tr>
<td>CreationDate</td>
<td>Creation date of the rendered content</td>
</tr>
<tr>
<td>Creator</td>
<td>Display name of the user who created the rendered content</td>
</tr>
<tr>
<td>CurrentStage</td>
<td>Name of the current workflow stage of the rendered content</td>
</tr>
<tr>
<td>Description</td>
<td>Localized description of the rendered content</td>
</tr>
</tbody>
</table>
Table 158. Properties for populating meta tag values (continued)

<table>
<thead>
<tr>
<th>Property</th>
<th>Meta tag value</th>
</tr>
</thead>
<tbody>
<tr>
<td>ExpiryDate</td>
<td>Expiration date of the rendered content</td>
</tr>
<tr>
<td>ID</td>
<td>ID of the rendered content</td>
</tr>
<tr>
<td>GeneralDateOne</td>
<td>Date from the general date one field</td>
</tr>
<tr>
<td>GeneralDateTwo</td>
<td>Date from the general date two field</td>
</tr>
<tr>
<td>Keywords</td>
<td>Keywords associated with the rendered content</td>
</tr>
<tr>
<td>LastModifiedDate</td>
<td>Date that the rendered content was last modified</td>
</tr>
<tr>
<td>LastModifier</td>
<td>Display name of the user who made the last change to the rendered content</td>
</tr>
<tr>
<td>Name</td>
<td>Name of the rendered content</td>
</tr>
<tr>
<td>Owners</td>
<td>Display names of the owners of the rendered content</td>
</tr>
<tr>
<td>PublishDate</td>
<td>Date the rendered content was published</td>
</tr>
<tr>
<td>SitePath</td>
<td>Site path of the rendered content</td>
</tr>
<tr>
<td>Status</td>
<td>Workflow status of the rendered content</td>
</tr>
<tr>
<td>Title</td>
<td>Localized title of the rendered content</td>
</tr>
<tr>
<td>Workflow</td>
<td>Name of the workflow of the rendered content</td>
</tr>
</tbody>
</table>

For several of the most common meta tags, default values are predefined. For these meta tags, you can create the portlet preference for only the meta tag name. The meta tag value is provided automatically, without the need for a corresponding name preference. The following meta tags have default values:

**Author**

The default value is a list of the authors of the rendered content.

**Keywords**

The default value is list of any keywords associated with the rendered content.

**Description**

The default value is the localized description of the rendered content.

If you do not want to use the default value, you can set the value using one of the methods previously described.

c. Optional: If the value of the meta tag requires a scheme attribute, specify the scheme attribute with the `meta.tag.scheme.suffix` preference. The meta tag scheme attribute takes the following format:

```
meta.tag.scheme.suffix=attribute_value
```

For example, to add the following scheme attribute with the value W3CDTF:

```
<meta name="DC.date" content="2000-01-01T12:00+00:00" scheme="W3CDTF"/>
```

Specify the following preference:

```
meta.tag.scheme.1=W3CDTF
```
The format and scheme that are used to write date elements and content properties related to date and time information, such as the LastModified property, depends on the meta tag attribute name. By default, all date and time information is formatted according to the date format defined by the HTTP specification. The format used to write date and time information in other meta tags is the data and time format recommended by the World Wide Web Consortium (W3C) under the scheme named W3CDTF.

Examples

The following examples demonstrate the different ways of specifying portlet preferences and the resulting meta tags in the output.

- Setting the meta tag value with the user who created the rendered content:
  meta.tag.name.1=DC.creator
  meta.tag.content.property.1=Creator

  Result:
  <meta name="DC.creator" content="content admin"/>

- Setting the meta tag value with preset text:
  meta.tag.name.1=DC.publisher
  meta.tag.content.text.1=IBM

  Result:
  <meta name="DC.publisher" content="IBM"/>

- Setting multiple meta tag values with the default value for the author and the value of the text element descelement in the rendered content:
  meta.tag.name.1=author
  meta.tag.name.2=description
  meta.tag.content.element.2=descelement

  Result:
  <meta name="author" content="content author"/>
  <meta name="description" content="Information about IBM"/>
  <meta name="DC.publisher" content="IBM"/>

- Setting the meta tag with an http-equiv attribute and a value of the date that the rendered content was last modified.
  meta.tag.name.1=last-modified
  meta.tag.attribute.1=http-equiv
  meta.tag.content.property.1=LastModifiedDate

  Result:
  <meta http-equiv="last-modified" content="Mon, 01 Aug 2011 13:45:57 GMT"/>

- Setting the meta tag and with a scheme attribute and a value of the date that the rendered content was published.
  meta.tag.name.1=DC.date
  meta.tag.scheme.1=W3CDTF
  meta.tag.content.property.1=PublishDate

  Result:
  <meta name="DC.date" content="2011-08-01T08:15:30+02:00" scheme="W3CDTF"/>
Performing remote rendering with WSRP and the web content viewer

To display web content on a portal that does not include IBM Web Content Manager, you can use the web content viewer and the WSRP support in the portal. The web content viewer can then retrieve and display content from a web content system on a different server.

**When to use remote rendering:** The preferred way to render content on one server from another server is to syndicate the content to the delivery server. On the delivery server, you can then locally render the content with a web content viewer. However, remote rendering with WSRP is appropriate for service-oriented architecture (SOA) scenarios where you want to incorporate specific pieces of content into your website.

When using the web content viewer for remote rendering with WSRP, the following conditions apply:

- The remote web content server acts as the WSRP Producer.
- The portal with the web content viewer acts as the WSRP Consumer.

**Note:** Remote rendering with WSRP is supported only when rendering content from the default virtual portal.

1. Set up the WSRP environment between the Producer portal and the Consumer portal, as described in *Using WSRP Services*. If you plan to use the **Edit Shared Settings** mode or the **Configure** mode in the portlet with WSRP, configure web service security between the Producer and the Consumer portals.
2. Provide the web content viewer portlet as a WSRP service hosted on the remote web content server acting as the WSRP Producer.
3. Consume the remote web content viewer provided as a WSRP service on the portal acting as the WSRP Consumer.
4. Configure the web content viewer to display content, just as you would configure a local web content viewer.

   When using the viewer with WSRP, settings for selecting content from a web content library show content from the remote web content system.

**Note:** Depending on the configuration of the web content viewer, resources like resource bundle files or content processor plug-ins might be required. In such cases, the resources must be available on the remote web content server acting as the WSRP Producer.

**Limitations when using WSRP with the web content viewer:**

- Because the concept of pages and web content pages does not exist in WSRP, you cannot use the dynamic link broadcasting feature with web content pages. When specifying how to broadcast links, do not select **Dynamically select a web content page** in the **Broadcast links to** field. Selecting this option has the same effect as broadcasting links to the current page.
- The use of authoring tools components or remote authoring action URLs in your web content is not supported with WSRP.
- All tagging and rating features for web content are not supported with WSRP.
- Personalization elements using selection rules based on the federated documents resource collection cannot be used with WSRP.
Advanced administrative examples

You can perform advanced administration for web content artifacts, such as web content pages and content associations.

Creating a web content page with the XML configuration interface

As with other portal pages, you can create a web content page with the XML configuration interface (**xmlaccess** command). Page definition is similar to a standard portal page. However, there is an additional page parameter that specifies the site area that is associated with the web content page.

1. When creating your **xmlaccess** command, specify your page parameters as you would for a standard portal page.

   Here is an example:
   ```xml
   <content-node action="update" content-parentref="parentOID" domain="rel" objectid="someOID" preserve-old-layout="true" type="page">
     <content-mapping-info>
       <content-mapping content-id="/mylibrary2/sitearea2" default="true" delegated-access-level="User"/>
       <content-mapping content-id="ddccb7ed-8485-48c8-b875-31d17d9da65b" default="false"/>
     </content-mapping-info>
   </content-node>
   ```

   **Note:** The value of the content-id attribute can be either the ID or the path to the web content item. If you are using the content path, the value must begin with the forward slash character (/) followed by the library name. When creating a web content page using the content path, you cannot build the path from the **Display title** fields of the items in the path. Instead you must use the **Name** fields of the items when specifying the path.

2. Because the web content viewer uses public render parameters to identify the content to render, include the page parameter **param.sharing.scope** when creating your **xmlaccess** command. Set the value for the parameter to **ibm.portal.sharing.scope.page**.

   Here is an example parameter definition for web content pages when using the XML configuration interface:
   ```xml
   <parameter name="com.ibm.portal.wcm.contentroot" type="string" update="set">
     <![CDATA[/mylib/mysite/mysitearea]]>
   </parameter>
   <parameter name="param.sharing.scope" type="string" update="set">
     <![CDATA[ibm.portal.sharing.scope.page]]>
   </parameter>
   ```

3. When creating your **xmlaccess** command, add at least one web content viewer that is configured to listen to other portlets and make dynamic broadcasts. Adding the viewer ensures that content selected for this page is rendered correctly and that links between pages work properly.
Here is an example of how to add the web content viewer using the XML configuration interface:

```xml
<component action="update" active="true" deletable="undefined" domain="rel"
modifiable="undefined" objectid="7_U796B1A0052501057F1BP3081" ordinal="100"
orientation="H" skinref="undefined" type="container" width="undefined">
  <component action="update" active="true" deletable="undefined" domain="rel"
modifiable="undefined" objectid="7_U796B1A0052501057F1BP3085" ordinal="100"
orientation="V" skinref="undefined" type="container" width="undefined">
    <component action="update" active="true" deletable="undefined" domain="rel"
modifiable="undefined" objectid="7_U796B1A0052501057F1BP3087" ordinal="100"
skinref="undefined" type="control" width="undefined">
      <portletinstance action="update" domain="rel" objectid="5_U796B1A0052501057F1BP3083"
portletref="3_U796B1A0052501057F1BP3084">
        <preferences name="WCM_BROADCASTS_TO" update="set">
          <value><![CDATA[WCM_LINKING_DYNAMIC]]></value>
        </preferences>
        <preferences name="WCM_LISTENS_TO" update="set">
          <value><![CDATA[WCM_LINKING_OTHER]]></value>
        </preferences>
      </portletinstance>
    </component>
  </component>
</component>
</component>
```

**Migration note:** After Version 6.1.5, the format used by the XML configuration interface to represent content associations for a web content page has changed. Typically, the migration process automatically converts all existing web content pages to the updated format. However, if you create web content pages on the older portal after migration and then import the pages to the Version 8.0 portal, the page format is incompatible. In this case, you must manually run the action-migrate-content-mappings configuration task on the Version 8.0 portal to convert the new web content pages to the Version 8.0 format. To perform the conversion, run the following task from the `wp_profile_root/ConfigEngine` directory:

- **Windows:** `ConfigEngine.bat action-migrate-content-mappings -DWasPassword=password -DPortalAdminPwd=password`
- **UNIX:** `./ConfigEngine.sh action-migrate-content-mappings -DWasPassword=password -DPortalAdminPwd=password`
- **IBM i:** `ConfigEngine.sh action-migrate-content-mappings -DWasPassword=password -DPortalAdminPwd=password`
- **z/OS:** `./ConfigEngine.sh action-migrate-content-mappings -DWasPassword=password -DPortalAdminPwd=password`

**Related concepts:**
- [The XML configuration interface](#)
- [Public render parameters](#)

**Content associations reference**

Content associations are used to associate web content pages with your web content site structure. When you select a folder to associate with a web content page, a content association is created and maintained within the portal.

Typically, the management of content associations is automatically handled by the portal. However, for situations where you want to work directly with content associations, several mechanisms are available: the XML configuration interface, the Portal Scripting Interface, and the REST API for content associations.
Related concepts:
“Web content associations” on page 537

Web content associations map portal pages to the site structure in the IBM Web Content Manager system. You can define a default content association and multiple other associations, which are used for dynamic page resolution. For each page, there is also a system content association that maps the page to its corresponding portal page site area in the Portal Site library.

XML configuration interface and content associations:

With the XML configuration interface (xmlaccess command), you can perform batch updates of content associations or export associations to import into another portal. Content association information is represented in the XML configuration schema by content-mapping-info elements.

Export content associations

The content associations for a web content page are represented in an XML export file as nested content-mapping-info elements.

The following example represents a web content page with two content associations:

```xml
<content-node action="update" content-parentref="6_000000000000000000000000A0" domain="rel"
    objectid="someOID" preserve-old-layout="true" type="label">
    <content-mapping-info>
        <content-mapping content-id="74-11" default="false"/>
        <content-mapping content-id="007" default="true" delegated-access-level="User"/>
    </content-mapping-info>
</content-node>
```

Note: If no content-mapping-info elements are present in an XML export document, there are currently no content associations defined for the web content page.

Import content associations

When importing content associations with an XML import file, associations for a web content page are represented in the content-mapping-info element for the web content page. Any content associations that are already defined for the web content page are removed when you perform the import process and replaced with the new associations.

The following example updates a web content page to have two specific content associations:

```xml
<content-node action="update" content-parentref="6_000000000000000000000000A0" domain="rel"
    objectid="someOID" preserve-old-layout="true" type="label">
    <content-mapping-info>
        <content-mapping content-id="74-11" default="false"/>
        <content-mapping content-id="007" default="true" delegated-access-level="User"/>
    </content-mapping-info>
</content-node>
```

Note: If no content-mapping-info element is present in an XML import document, no changes are made to the content associations currently defined for the web content page.
Delete content associations

You can delete content associations by specifying an empty content-mapping-info element in the XML import file.

The following example updates a web content page to delete any defined content associations:

```xml
<content-node action="update" content-parentref="6_000000000000000000000000A0" domain="rel"
        objectid="someOID" preserve-old-layout="true" type="label">
    <content-mapping-info/>
</content-node>
```

Related concepts: [The XML configuration interface](#)

Portal Scripting Interface and content associations:

With the Portal Scripting Interface, you can create scripts to automate the management of content associations. Using the ContentMapping bean with the Portal Scripting Interface, you can add, modify, and remove content associations.

**Note:** Before a script can work with the ContentMapping bean, you must establish a user session with the portal using the `login` command of the Portal bean. The user identity must have sufficient permissions to administer the web content pages and web content library folders referenced by the script.

Retrieve content associations

To retrieve content association information, use the select method to specify the object ID of the web content page. You can often derive the object ID for a resource from another bean and use that as input for the select method. For example, you might have a web content page with the unique name `my.test.page`. Using the find method of the Content bean, you can determine the object ID of the `my.test.page` page.

**Jacl example:**

```
$set the_page [$Content find page uniquename "my.test.page"]
$ContentMapping select $the_page
```

**Jython example:**

```
the_page = Content.find('page','uniquename','my.test.page')
ContentMapping.select(the_page)
```

After you have the object ID of the web content page, you can use the list method and the get methods to access the content associations. The list method returns a list of content association IDs. The IDs can identify either the resource ID of a folder or the content path of the folder, depending on how the page is mapped. You can use the content association IDs returned by the list method as arguments for the get method.

**Jacl example:**

```
$set the_page [$Content find page uniquename "my.test.page"]
$ContentMapping select $the_page
foreach mid [$ContentMapping list mappings] {
    puts " Mapping $mid info:"
    puts " content id: [$ContentMapping get $mid content-id]"
    puts " default? [$ContentMapping get $mid isdefault]"
    puts " scope: [$ContentMapping get $mid scope]"
}
```
Jython example:

```python
var the_page = Content.find('page','uniquename','my.test.page')
ContentMapping.select(the_page)
for mid in ContentMapping.list('mappings').split():
    print " Mapping " + mid + " info:"
    print "  content id: " + ContentMapping.get(mid,'content-id')
    print "  default? " + ContentMapping.get(mid,'isdefault')
    print "  scope: " + ContentMapping.get(mid,'scope')
```

The get method can return the default association for the selected web content page. The list method can retrieve a list of scopes that are defined for the associations of the web content page.

Jacl example:

```clike
$set the_page [$Content find page uniquename "my.test.page"]
$ContentMapping select $the_page
puts "available scopes: [$ContentMapping list scopes]"
puts "default mapping: [$ContentMapping get defaultmapping]"
puts "portal resource OID: [$ContentMapping get oid]"
```

Jython example:

```python
var the_page = Content.find('page','uniquename','my.test.page')
ContentMapping.select(the_page)
print "available scopes: " + ContentMapping.list('scopes')
print "default mapping: " + ContentMapping.get('defaultmapping')
print "portal resource OID: " + ContentMapping.get('oid')
```

Add content associations

Use the add method to add new content associations to a web content page. You can assign a content association by specifying the content path of the folder or the ID of folder. If you identify the folder by content path, the association is internally transformed to actually point to the ID of the folder. As a result, if you rename the folder later, the association still points to the same folder.

Jacl example:

```clike
$ContentMapping select [$ContentNode find page uniquename "my.sample.page"]
$ContentMapping add content-path "/test1/mapping"
set the_content_id .... ## obtain ID of content to be mapped
$ContentMapping add id $the_content_id
```

Jython example:

```python
ContentMapping.select(Content.find('all','un','my.sample.page'))
ContentMapping.add('content-path','/test1/mapping')
var the_content_id = ... ## obtain ID of content to be mapped
ContentMapping.add('id',the_content_id)
```

Remove content associations

The ContentMapping bean provides two methods you can use to remove content associations from a web content page:

```
remove
Removes an individual content association, as specified either by the resource ID of the folder or the content path of the folder.
```

```
delete
Removes all content associations for the web content page.
```

The following examples demonstrate how to remove the content associations for two web content pages. The content associations of the first page are removed individually with the remove method, and the content associations of the second page are removed with the delete method.
Modify content associations

To modify content associations, use the set method of the ContentMapping bean. You can change the following attributes:

- Default flag
- Delegation mode
- Mapping scope

When calling the set method, pass in the ID of the content association that you want to update.

The following example updates two content associations for the web content page identified by the unique name `my.test.page`. Several settings are specified for the first content association:

- The default flag is set to make this content association the default content association for the web content page.
- The association scope is specified as `_scp_`.
- Page-based access control is turned off by setting the delegation mode to `false`.

For the second content association, the association scope is removed by specifying an empty string.

Jacl example:

```jacl
$ContentMapping select [ContentNode find page uniquename "my.sample.page"]
set first_m_id [lindex [ContentMapping list mappings] 0]
$ContentMapping set scope $first_m_id "_scp_"
$ContentMapping set default $first_m_id true
$ContentMapping set delegation $first_m_id false
set second_m_id [lindex [ContentMapping list mappings] 1]
$ContentMapping set scope $second_m_id ""
```

Jython example:

```python
ContentMapping.select(Content.find('all','un','my.sample.page'))
var first_m_id = ContentMapping.list('mappings').split()[0]
ContentMapping.set('scope',first_m_id,'_scp_')
ContentMapping.set('default',first_m_id,'true')
ContentMapping.set('delegation',first_m_id,'false')
var second_m_id = ContentMapping.list('mappings').split()[1]
ContentMapping.set('scope',second_m_id,'')
```
Related reference:
Portal Scripting Interface

REST API and content associations:

If you are creating or extending an application and want to manage content associations with that application, you can use portal remote APIs. These APIs retrieve a list of content associations and then create, update, or delete associations.

Based on the Representational State Transfer (REST) architecture, the APIs represent information about content associations as Atom feeds. To perform actions, you send HTTP requests to specific URLs.

Retrieval of all content associations

This request returns a feed containing all content associations available in the system.

**URL**  
http://hostname:port/context_root/  
mypoc?uri=contentmapping:objecttype:CONTENT_NODE&mode=download

**HTTP method**  
GET

**Links**  
Link information is provided for each entry in the Atom feed, as identified by the rel attribute.

<table>
<thead>
<tr>
<th>Link</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rel=&quot;self&quot;</td>
<td>Link to this Atom entry.</td>
</tr>
<tr>
<td>rel=&quot;edit&quot;</td>
<td>Link to this item that can be used for POST, PUT, and DELETE operations.</td>
</tr>
<tr>
<td>rel=&quot;related&quot;</td>
<td>Link that can be used to view the web content page with which the content association is associated.</td>
</tr>
<tr>
<td>rel=&quot;first&quot;</td>
<td>Link to the first feed fragment. This link is only served if a feed fragment was served.</td>
</tr>
<tr>
<td>rel=&quot;last&quot;</td>
<td>Link to the last feed fragment. This link is only served if a feed fragment was served.</td>
</tr>
<tr>
<td>rel=&quot;previous&quot;</td>
<td>Link to the feed fragment preceding the current feed fragment. This link is only served if a feed fragment was served that does not start at the beginning of the feed.</td>
</tr>
<tr>
<td>rel=&quot;next&quot;</td>
<td>Link to the next feed fragment. This link is only served if a feed fragment was served that does not contain the last entry of the feed.</td>
</tr>
</tbody>
</table>

Supported URL parameters

**Table 160. Supported parameters for requests to retrieve content associations**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>start-index</td>
<td>Identifies the start index of the feed fragment to be served. The default value is 0.</td>
</tr>
</tbody>
</table>
Table 160. Supported parameters for requests to retrieve content associations (continued)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>max-results</td>
<td>Identifies the maximum number of entries to be served by this request, as determined by the configuration of the server.</td>
</tr>
</tbody>
</table>

Retrieve content associations for a specific web content page

This request returns a feed containing the content associations defined for a specific web content page.

**URL**  
The web content page is identified either by its object ID or by its unique name.


**HTTP method**  
GET

**Links**  
Link information is provided for each entry in the Atom feed, as identified by the rel attribute.

Table 161. Link information in the Atom feed for retrieving content associations for a specific web content page

<table>
<thead>
<tr>
<th>Link</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rel=&quot;self&quot;</td>
<td>Link to this Atom entry.</td>
</tr>
<tr>
<td>rel=&quot;edit&quot;</td>
<td>Link to this item that can be used for POST, PUT, and DELETE operations.</td>
</tr>
<tr>
<td>rel=&quot;related&quot;</td>
<td>Link that can be used to view the web content page with which the content association is associated.</td>
</tr>
</tbody>
</table>

Modify content associations

This request updates the content associations defined for a specific web content page.

**URL**  
The web content page is identified either by its object ID or by its unique name.


**HTTP method**  
PUT

**Links**  
Link information is provided for each entry in the Atom feed, as identified by the rel attribute.
Table 162. Link information in the Atom feed for modifying content associations

<table>
<thead>
<tr>
<th>Link</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rel=&quot;self&quot;</td>
<td>Link to this Atom entry.</td>
</tr>
<tr>
<td>rel=&quot;edit&quot;</td>
<td>Link to this item that can be used for POST, PUT, and DELETE operations.</td>
</tr>
<tr>
<td>rel=&quot;related&quot;</td>
<td>Link that can be used to view the web content page with which the content association is associated.</td>
</tr>
</tbody>
</table>

Supported URL parameters

Table 163. Supported parameters for requests for modifying content associations

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>update</td>
<td>The following values are supported for the update parameter:</td>
</tr>
<tr>
<td></td>
<td>merge This mode merges the content associations in the request with the content associations on the server. The request updates existing content associations and adds new content associations, but the request does not delete other content associations already on the server.</td>
</tr>
<tr>
<td></td>
<td>replace This mode replaces all current content associations on the server with the content associations specified in the request. The request updates existing content associations, adds new content associations, and deletes other content associations on the server that are not represented in the request.</td>
</tr>
<tr>
<td></td>
<td>delete This mode deletes the content association specified in the request from the web content page.</td>
</tr>
</tbody>
</table>

Delete content associations

This request deletes either all content associations for a specific web content page or an individual content association to specific content item.

URL The web content page is identified either by its object ID or by its unique name.

http://hostname:port/context_root/
my poc?uri=contentmapping:oid:serialized_object_id

http://hostname:port/context_root/
my poc?uri=contentmapping:oid:unique_name

HTTP method
DELETE

Supported URL parameters
Table 164. Supported parameters for requests for deleting content associations for a web content page

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>content</td>
<td>Indicates the content associations to be deleted. If the content parameter is not specified, all content associations for the web content page are deleted. If the content ID for a content item is specified as the value of the content parameter, only one content association is deleted. The content association deleted is the one that maps the web content page to the specified content item. Any other content associations are unaffected.</td>
</tr>
</tbody>
</table>

Example Atom feed document

```xml
<?xml version="1.0" encoding="UTF-8"?>
  <atom:title>Content Mappings Feed</atom:title>
  <atom:author>
    <atom:name>WebSphere Portal</atom:name>
  </atom:author>
  <atom:link href="/wps/mycontenthandler/!ut/p/contentmapping/objecttype%3aCONTENT_NODE?max-results=2" rel="self" type="application/atom+xml"/>
  <atom:id>contentmapping:objecttype%3aCONTENT_NODE</atom:id>
  <atom:entry>
    <atom:id>contentmapping:oid6_2QC6BB1A0GVJEO1OA0RQ002040</atom:id>
    <atom:title>6_2QC6BB1A0GVJEO1OA0RQ002040</atom:title>
    <atom:link href="/wps/mycontenthandler/!ut/p/contentmapping/oid6_2QC6BB1A0GVJEO1OA0RQ002040" rel="self" type="application/atom+xml"/>
    <atom:link href="/wps/mycontenthandler/!ut/p/contentmapping/oid6_2QC6BB1A0GVJEO1OA0RQ002040" rel="edit" type="application/atom+xml"/>
    <atom:link href="/wps/myportal/!ut/p/c4/04_SB8K8xLMLM9MS5zPyx6x8z9CP0os3ijQ6czCyDWRP3MC9xA90_R4Mgg0ADlwMTA_2cbEdFAMzmm-MI/" type="text/html"/>
    <atom:content type="application/xml">
      <contentmappings:content-mapping-info id="6_2QC6BB1A0GVJEO1OA0RQ002040">
        <contentmappings:content-mapping content-id="f40429bb-0cb5-4cf0-8080-cfb66f0e9b91" default="true"/>
      </contentmappings:content-mapping-info>
    </atom:content>
  </atom:entry>
  <atom:entry>
    <atom:id>contentmapping:oid6_2QC6BB1A00VBC00SHA0A22006</atom:id>
    <atom:title>cnCTFFPortlet</atom:title>
    <atom:link href="/wps/mycontenthandler/!ut/p/contentmapping/oid6_2QC6BB1A00VBC00SHA0A22006" rel="self" type="application/atom+xml"/>
    <atom:link href="/wps/mycontenthandler/!ut/p/contentmapping/oid6_2QC6BB1A00VBC00SHA0A22006" rel="edit" type="application/atom+xml"/>
    <atom:link href="/wps/myportal/!ut/p/c4/04_SB8K8xLMLM9MS5zPyx6x8z9CP0os3ijQ6czCyDWRwOMCdnA0__Y19QA0cjiwN_M_2cbEdFAMzmm-MI/" type="text/html"/>
    <atom:content type="application/xml">
      <contentmappings:content-mapping-info id="6_2QC6BB1A00VBC00SHA0A22006">
        <contentmappings:content-mapping content-id="6f0b7a8d426a9a8d53ed9170f1e3d" default="true"/>
      </contentmappings:content-mapping-info>
    </atom:content>
  </atom:entry>
</atom:feed>
```
Dynamic web content page selection

The web content viewer provides a link broadcasting feature that dynamically determines the best web content page to use when rendering the linked content item. To perform this page selection, the viewer uses the content associations on the web content page.

Understanding dynamic page selection is useful when examining how multiple associations affect link resolution and when troubleshooting why a link is not targeting the expected page.

Dynamic page selection is enabled when the web content viewer is configured to use the Dynamically select a web content page link broadcasting option. To render a content item, the dynamically selected target page must contain at least one web content viewer. The viewer must be configured to receive links from other portlets.

When users select a link to a content item, the chain of content page resolution filters runs to determine which page renders the selected item. The same mechanism is used when previewing web content on pages or when selecting search results produced from the search seedlist 1.0 feature. You can control how the page is determined by adding a custom content page resolution filter into this filter chain. For more information, see Creating a content page resolution filter class.

The default filter plug-ins perform the following tests to determine a web content page.

1. An ordered list with all parent site areas of the selected content item is created. If the selected item itself is a site area, it is part of that list. The order in the list matches the order in the Web Content Manager content hierarchy. The topmost parent is the last item in the list, and the direct parent of the selected item (or the item itself) is the first item.

2. Based on this ordered list, a lookup is performed to find the web content page with the following criteria:
   - The current user has view access to the page.
   - The page is mapped to the first site area for which a page content association exists.

   If the content associations indicate that multiple pages map equally well, the web content viewer selects a single page using the following rules:
   - If the current page is among the pages found, the current page takes precedence over the other results.
   - If one of the pages that is indicated by the content associations is the default association for the page, that page takes precedence over other results.
   - If the previous rules do not identify a page, a page is selected arbitrarily from the set of found pages.

3. The content item that gets rendered depends on the situation.
   - If a web content page can be determined in the previous test, the selected content item is rendered on this page.
• If no page can be identified by the content associations, the web content fallback page is used to render the content item.
• If no fallback page is configured or if the user does not have view rights on the fallback page, the Web Content Manager servlet renders the item.

Related tasks:
“Creating a content page resolution filter class” on page 652
A content page resolution filter is used to customize the behavior of the content page resolution filter chain. This can enable you to tailor the response to a web content request in several ways, including overriding the content item displayed or the portal page used to display a content item in the web content viewer.

Web content viewer best practices and limitations
View some best practices and limitations for using web content viewers.

User authentication
User authentication to web content viewers is managed by IBM WebSphere Portal and IBM WebSphere Application Server. See Enabling step-up authentication, the Remember me cookie, or both for further information.

Security and WSRP
When displaying web content from remote servers using WSRP and a web content viewer, configure security and authentication between the servers:
• The portal server acting as the WSRP Consumer
• The web content server acting as the WSRP Producer
See Security considerations for WSRP services for details.

User access to Web Content Manager content and components
Users are able to view only content and components that can be accessed by either a portal user ID or the user of the WSRP Consumer. This access must be defined in Web Content Manager. If a user ID or the user of the WSRP Consumer does not have sufficient rights to view a content or component, errors can occur.

Content and component limitations
Not all content or components built in a Web Content Manager solution are suitable for inclusion in a portal page:
• Content or components to be shown within a portal page must be self-contained and not rely on other content or components.
• When creating presentation templates or page styles to use when showing content within a portlet, reference only the content you want to show. Add components like menus and navigators in separate portlets, and link the components to other content portlets as required.
• JavaScript URLs are not supported.

Using JavaScript:
When a web page is rendered by Web Content Manager, some tags might be rewritten. Web Content Manager uses double quotation marks for attributes in HTML tags. If you use JavaScript to produce HTML tags, Web Content Manager does not recognize them if you use single quotation marks.
Other limitations

- The results of a POST operation in a form are only displayed within the portlet that sent the POST. You cannot send the result of a POST to another portlet.
- An anonymous portal user is also considered an anonymous Web Content Manager user, so overriding the log-in does not work for anonymous users.
- If a Web Content Manager proxy server is being used with web content viewers, URLs rewritten by the proxy are not fully qualified. Instead, the URLs are relative to the server. To address this issue, redirect mappings can be created in the HTTP Server configuration that pass the URLs to the proxy server.
- Category selection trees cannot be used with the local web content viewer.
- Only advanced caching can be used with a local web content viewer.
- Tagged web content that is displayed in the web content viewer is only available when there is a single instance of the portlet on the page. When you click a tag result, the Tag Center broadcasts the information about what content to display using a public render parameter. If you are displaying multiple instances of content in the viewer, the instances show the content that you tagged rather than their original content.

Related information:
- Security considerations for WSRP services
- Public render parameters
- Enabling step-up authentication, the Remember me cookie, or both

Pre-rendered delivery

You can pre-render a complete IBM Web Content Manager site into HTML and save it to disk. The pre-rendered site can then be used as your live site and displayed to end users using either Web Content Manager or a web server. You deploy a pre-rendered site when you are not using any WebSphere Portal features and your content is static and is only updated periodically.

Restrictions

- Site areas and content item names cannot contain characters that are considered invalid in file names by the operating system on which you are pre-rendering. For example, on a machine running Windows, these characters are invalid: / \ : * ? " < > |.
- The path to the content item, including the directory path to which you are pre-rendering (for example, site area/content) cannot exceed the operating system’s maximum path length:
  - 255 characters in Windows
  - 1024 characters in Linux
  - 1024 else
- The Search component cannot be used in pre-rendered sites.
- The Page navigation component cannot be used in pre-rendered sites.
- Personalization elements can only be pre-rendered if the personalization rule is configured for anonymous access.

Site security

Item security for different users set in an Web Content Manager environment is not transferred to pre-rendered sites. The security for the entire pre-rendered site is based on the
connect.moduleconfig.cacher.rendereruser property as specified in the WCM WCMConfigService service using the WebSphere Integrated Solutions Console.

Related concepts:
"Pre-rendering options" on page 89

You can enable pre-rendering so that content can be viewed either through a IBM Web Content Manager application or as a standalone site that is accessed through a web server.

Related tasks: Setting service configuration properties

IBM WebSphere Portal comprises a framework of services to accommodate the different scenarios that portals need to address. Services are available for both WebSphere Portal and IBM Web Content Manager. You can configure some of these services.

Pre-rendering a website

Pre-rendering can be configured to run automatically, or you can manually pre-render a website using a URL.

Administrator access: In order to pre-render a website, you must have administrator access to the library that contains the site area.

Automatically pre-rendering a website

Pre-rendering can be run according to the cacher settings specified for the WCM WCMConfigService service as part of the pre-rendering configuration.

Manually pre-rendering a website

Pre-rendering can be also be initiated through the URL interface. For example:

http://host_name:port_number/wps/wcm/connect?MOD=Cacher&SRV=cacheSite&sitearea=sitearea_name&library=library_name

Table 165. CacherModule options

<table>
<thead>
<tr>
<th>Service</th>
<th>Required Parameters</th>
<th>Optional Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>$RV=cacheSite</td>
<td>sitearea=&lt;site area name&gt;</td>
<td>DELAY=&lt;delay&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LIBRARY=&lt;library&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Note:</strong> If no library is specified, the default library is used, as specified by the defaultLibrary property in the WCM WCMConfigService service.</td>
</tr>
<tr>
<td>$RV=flushSiteCache</td>
<td>sitearea=&lt;sitearea name&gt;</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>LIBRARY=&lt;library&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Note:</strong> If no library is specified, the default library is used, as specified by the defaultLibrary property in the WCM WCMConfigService service.</td>
</tr>
</tbody>
</table>
Table 165. CacherModule options (continued)

<table>
<thead>
<tr>
<th>Service</th>
<th>Required Parameters</th>
<th>Optional Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>SRV=flushPageCache</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Flushes the page from the site cache. The site area and page are determined from the request URL.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>No SRV specified</td>
<td></td>
<td></td>
</tr>
<tr>
<td>The CacherModule attempts to retrieve the given page from the cache.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

You can also pre-render individual pages using the following URL:

http://host_name:port_number/wps/wcm/connect/library_name/site_area_name/content?MOD=Cacher

Note:
- To pre-render individual content items the site area specified in the URL must either be a site area set in the WCM WCMConfigService service, or you must have previously manually pre-rendered the site area using "SRV=cacheSite".

Related concepts:
- “Pre-rendering options” on page 89

You can enable pre-rendering so that content can be viewed either through an IBM Web Content Manager application or as a standalone site that is accessed through a web server.

Related tasks:
- Setting service configuration properties

IBM WebSphere Portal comprises a framework of services to accommodate the different scenarios that portals need to address. Services are available for both WebSphere Portal and IBM Web Content Manager. You can configure some of these services.

**Accessing the pre-rendered site**

Pre-rendered sites are accessed either through IBM Web Content Manager, or through a web server.

**Accessing the pre-rendered site through a Web Content Manager application**

To enable users to access the pre-rendered site through a Web Content Manager application, specify the connect.businesslogic.module.default.class property in the WCM WCMConfigService service using the WebSphere Integrated Solutions Console.
- Property name: connect.businesslogic.module.default.class
- Value: com.aptrix.cacher.CacherModule

Users can access the site through the following URL:

http://host_name:port_number/wps/wcm/connect/library_name/sitearea_name
The `library_name` parameter is optional. If no library is specified, the default library is used, as specified by the `defaultLibrary` property in the `WCM WCMConfigService` service.

**Connect tags:**
Connect tags are not processed by the CacherModule and are rendered intact. When the pre-rendered page is accessed by a user, only then will the connect tags be processed.

**Pre-rendering JSP components:**
The pre-rendering feature cannot be used to pre-render JSP components.

**Links to content not yet pre-rendered:**
A component, such as a menu, that contains links to content not yet pre-rendered is retrieved by the CacherModule and added to the pre-rendered site. This only applies to content belonging to sites configured to be pre-rendered.

**Custom expiring:**
Custom caching parameters cannot be used in connect tags and URL requests in pre-rendered sites. "EXPIRES=" and "CONNECTORCACHEEXPIRY=" can be used to override your server's default basic and data cache settings.

**Authoring portlet:**
The authoring portlet can still be accessed when the default class is changed from `RendererModule` to `CacherModule` as long as it has not been added to the lists of sites to be pre-rendered.

**CacherModule as default:**
If using the URL interface when the CacherModule is the default, you do not need to specify `?mod=cacher`. Instead, enter the request as follows:

```
http://host_name:port_number/wps/wcm/connect?SRV=cacheSite
&library=library_name&sitearea=sitearea_name
```

**Accessing the pre-rendered site through a web server**

If your web server is not used for WebSphere Portal Server, you can configure your web server to map to the following alias and Web Content Manager directories:

<table>
<thead>
<tr>
<th>Alias</th>
<th>Directory</th>
</tr>
</thead>
<tbody>
<tr>
<td>/wps/wcm/connect</td>
<td>[ILWWCM_HOME]/ilwwcm/cacher</td>
</tr>
</tbody>
</table>

When using a web server to view a pre-rendered site, the previous directories require execute access.

Users access the site via the following URL:

```
http://host_name:port_number/wps/wcm/connect/library_name/sitearea_name
```

Use the following configuration properties in the `WCM WCMConfigService` to change the context of the URLs generated with pre-rendering:

- `connect.moduleconfig.cacher.task.cacherurl`
- `connect.moduleconfig.cacher.task.servletpath`

For example, to set a context of `/sales`, use the following properties:
Cacher URL

- Property name: connect.moduleconfig.cacher.task.cacherurl
- Value: http://${WCM_HOST}:${WCM_PORT}/sales

Servlet path

- Property name: connect.moduleconfig.cacher.task.servletpath
- Value: /connect

If your web server is used for both a WebSphere Portal server and accessing the pre-rendered site, you must change the context of the URLs. Any context that starts with the WebSphere Portal server context, /wps for example, will be redirected to WebSphere Portal server.

Connect tags cannot be used:

Connect tags are not processed by the CacherModule and are rendered intact. When the page is viewed through a web server, connect tags cannot be processed. Therefore, connect tags cannot be used in sites that are to be viewed through a web server.

Pre-rendering JSP components:

The pre-rendering feature cannot be used to pre-render JSP components.

Dynamic elements:

When viewing a pre-rendered site through a web server, any dynamic elements such as menus and navigators are displayed as rendered by the configured CacherModule user, not by the user accessing the site. This means personalization cannot be used.

Related tasks:

Setting service configuration properties

IBM WebSphere Portal comprises a framework of services to accommodate the different scenarios that portals need to address. Services are available for both WebSphere Portal and IBM Web Content Manager. You can configure some of these services.
Chapter 7. Maintaining web content

You use these tools and features to ensure the efficient operation of your Web Content Manager system.

Using the web content member fixer task

Use the member fixer task to check whether any users or groups referenced in IBM Web Content Manager items have been renamed or deleted and fix these references.

Member fixer is used to:

- fix references to users in library and item level access settings that refer to users and groups from a given user repository where the structure of the user repository has been altered. For example, an LDAP transfer may have been executed, or the LDAP schema may have changed, or users and groups may have been moved in the LDAP.
- fix references to users in item level access settings that refer to users and groups who have been deleted from the user repository.

The member fixer task's function is to check all of the items in a specified library for references to users and groups that no longer exist in the current user repository. In report mode, it will report all the references to members. In fix mode, these references can be fixed, either by replacing them with references to members that exist, or by removing the references. The `fix` parameter determines whether the member fixer task runs in report or fix mode.

References to members in library items contain the distinguished name of the member as well as a unique ID for the member. This unique ID is an internal id that is unique over time, and is different to the distinguished name. This means if a member is deleted and another member is created with the same distinguished name, the two members will have different unique IDs. The `mismatchedId` parameter can be used to update or remove references from web content items to users with these unique IDs.

When a member that has been given permissions on a library is deleted, the member permissions are entirely removed from the library, so that any inherited permissions for items in the library will also be removed. Therefore, the member fixer task can not be used to update these permissions to a different member. However, when an LDAP transfer is carried out, the member permissions on the library are maintained. So, the member fixer task can be run after an LDAP transfer to update or remove these permissions.
Enabling the member fixer tool

You must first enable the member fixer by adding the following parameters to the WCM WCMConfigService service using the WebSphere Integrated Solutions Console:

- connect.businesslogic.module.memberfixer.class=com.aptrix.pluto.security.MemberFixerModule
- connect.businesslogic.module.memberfixer.remoteaccess=true
- connect.businesslogic.module.memberfixer.autoload=false

Custom Mapping

To update a reference to a member that does not exist with a member that does exist, member mappings can be defined in a custom mapping file. Where the member fixer task does not find a mapping in this file for a member, it will search the user repository for members with the same ID as the member that no longer exists. If such a member is found, it will update the reference with this user or group, or remove the reference, as specified by the altDn parameter. If no such member is found, this member is classified as ‘invalid’ and will be updated or removed as specified by the invalidDn parameter.

If custom mapping is required you must perform the following steps to map the user and group domain names before running the member fixer task:

1. Update the following properties in the wp_profile_root/PortalServer/wcm/shared/app/config/wcmservices/MemberFixerModule.properties file:
   - cn=contentAuthors,dc=lotus,o=ibm->cn=contentEditors,dc=rational,o=ibm
     This format is used to completely replace one distinguished name with another.
   - cn=[ID],dc=websphere,o=ibm->cn=[ID],dc=tivoli,o=ibm
     This format is used to replace part of a distinguished name. This example will change all of the distinguished name except the common name.

   Further examples are listed in the MemberFixerModule.properties file.

2. You then run the member fixer task using the -DaltDn option as details in the following section.

Running the Member Fixer task:

1. Open a command prompt. Open a UNIX System Services (USS) command prompt.

   Library parameters in steps 2 and 3:
   - The library specified in the command is the library to be scanned by the member fixer task. If the query parameter "library" is omitted, the default library that has been configured with the defaultLibrary property in the WCM WCMConfigService service is used.

2. To create a report of users or groups referenced in Web Content Manager items that need fixing, run the following command from the wp_profile_root/ConfigEngine directory:

   Windows
   ConfigEngine.bat run-wcm-admin-task-member-fixer
   -DPortalAdminId=username -DPortalAdminPwd=password
   -DWasUserId=username -DWasPassword=password -DLibrary="MyLibrary"

   UNIX
   ./ConfigEngine.sh run-wcm-admin-task-member-fixer
Note: An administrator user name and password is not required if you have already specified the portal administrator username and password using the PortalAdminId and PortalAdminPwd settings in the wkplc.properties file.

Note: Before progressing to the next step and running the member fixer task in fix mode, ensure that the report mode indicates that the updates will happen as you require. A summary of the updates will be shown by the command.

Windows, UNIX and IBM i

A detailed report containing the updates that will be made for each item will be shown in the SystemOut.log file located in wp_profile_root\logs\WebSphere_Portal.

z/OS

A detailed report containing the updates that will be made for each item will be shown in the job log for the portal application server.

If the report indicates that the update will not happen as required, change the member fixer task parameters and run the report mode again. Repeat this process until you are satisfied that the fixes will be applied correctly. This is important because the fixes made by the member fixer task when run in fix mode may not be easy to undo if incorrect fixes are applied.

3. If there have been changes to users and groups, update the items that reference them by running the following command:

Windows

ConfigEngine.bat run-wcm-admin-task-member-fixer
-DPortalAdminId=username -DPortalAdminPwd=password
-DWasUserId=username -DWasPassword=password -DLibrary="MyLibrary"
-Dfix=true

UNIX

./ConfigEngine.sh run-wcm-admin-task-member-fixer
-DPortalAdminId=username -DPortalAdminPwd=password
-DWasUserId=username -DWasPassword=password -DLibrary="MyLibrary"
-Dfix=true

IBM i

ConfigEngine.sh run-wcm-admin-task-member-fixer
-DPortalAdminId=username -DPortalAdminPwd=password
-DWasUserId=username -DWasPassword=password -DLibrary="MyLibrary"
-Dfix=true

z/OS

./ConfigEngine.sh run-wcm-admin-task-member-fixer
-DPortalAdminId=username -DPortalAdminPwd=password
-DWasUserId=username -DWasPassword=password -DLibrary="MyLibrary"
-Dfix=true

Note: An administrator user name and password is not required if you have already specified the portal administrator username and password using the PortalAdminId and PortalAdminPwd settings in the wkplc.properties file.
PortalAdminId and PortalAdminPwd settings in the wkplc.properties file.
If the member fixer task indicates that certain mismatched member conditions exist, append the specified parameters to the command:

### Table 167. Member fixer conditions

<table>
<thead>
<tr>
<th>Condition description</th>
<th>Command to correct condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nonexistent users or groups have alternate distinguished names (DNs) available.</td>
<td>- To update references to nonexistent users or groups with the portal administrator user distinguished name, append -DalDn=update to the command.</td>
</tr>
<tr>
<td></td>
<td>- To remove references to nonexistent users or groups append -DalDn=remove to the command.</td>
</tr>
<tr>
<td>If users or groups have invalid distinguished names (DNs) the report will list these as &quot;invalid&quot;. This means the distinguished name doesn't exist and there is no alternate distinguished name available.</td>
<td>- To remove references to users and groups that have invalid distinguished names append -DinvalidDn=remove to the command.</td>
</tr>
<tr>
<td>Users or groups have been found with mismatched unique IDs.</td>
<td>- To fix the mismatched unique IDs append -DmismatchedId=update to the command.</td>
</tr>
<tr>
<td></td>
<td>- To remove references to users and groups with mismatched unique IDs append -DmismatchedId=remove to the command.</td>
</tr>
</tbody>
</table>

4. After the member fixer task has run, review the SystemOut.log to verify that the member fixer task ran correctly. The member fixer task may not be able to save items that fail validation, such as items that contain invalid fields. You must edit these items to make them valid and then run the member fixer task again.

5. After the member fixer task has run, review the job log for the portal application server to verify that the member fixer task ran correctly. The member fixer task may not be able to save items that fail validation, such as items that contain invalid fields. You must edit these items to make them valid and then run the member fixer task again.

### Running the Member Fixer in a federated security environment

In a federated security environment with multiple realms, you can specify the realm to run the member fixer task on by adding -Drealm=realmName to the command. If this parameter is omitted the default realm will be used.

The member fixer task will check whether there are any members and groups referenced in items that contain any of the base distinguished names defined for the specified realm and fix these references. References to members can only be updated with references to members in the specified realm.

Additionally, the member fixer task can be used to check whether there are any members and groups referenced in items that are not under any of the base distinguished names defined for any of the realms in the environment and fix these references. To do this, follow the same steps described for a single realm environment and add -DnoRealmDn=true to the command.
In a federated security environment with multiple realms, the member fixer task should be run for each realm in turn to make sure that all of the references are fixed.

**Preserving dates**

You can preserve the last modified date of items updated by the member fixer task by adding `-DpreserveDates=true` to the command. Otherwise the last modified date will be updated when the member fixer task is run.

**Restricting which items types to fix**

You can restrict which objects types are processed by appending `-DrestrictOn=ItemType` to the command.

For example:

- `content`
- `folder`
- `project`
- `style` for presentation templates
- `template` for authoring templates
- `taxonomy`
- `category`
- `SiteArea`
- `Workflow`
- `WorkflowStage`
- `WorkflowAction`
- `Cmpnt` for components

You can restrict multiple object types by separating the types with a comma (,). For example, to restrict workflows and workflow stages, you can specify `-DrestrictOn=Workflow,WorkflowStage`.

If not specified, all object types will be updated.

**Running the task for all libraries**

You can run this task for all libraries by replacing the option `-Dlibrary=libraryName` with the option `-DallLibraries=true` in the command. If neither option is specified, this task will process the default library that has been configured in the WCM WCMConfigService service using the WebSphere Integrated Solutions Console.

**Running the task on a virtual portal**

When running this task on a virtual portal you must either add `-DVirtualPortalHostName=name` or `-DVirtualPortalContext=context` to the command.

**Parameters to set for large repositories**

To prevent your session timing out before the task has finished, you can append the option `-DsessionTimeOut=timeOut` to the command. This sets the number of
seconds in which the task must complete before its session will timeout. The
default session timeout is 14,440 seconds, which is 4 hours. For large repositories
you should increase this setting. For example: `-DsessionTimeOut=36000`, which is 10
hours.

**Examples**

These options can be combined when the conditions occur at the same time. For
example, if alternate DNs are available for nonexistent users and groups and there
are mismatched unique IDs, you would use the following command:

**Windows**

```
ConfigEngine.bat run-wcm-admin-task-member-fixer
-DPortalAdminId=username -DPortalAdminPwd=password
-DWasUserId=username -DWasPassword=password -Dlibrary="MyLibrary"
-Dfix=true -DaltDn=update -DmismatchedId=update
```

**UNIX**

```
./ConfigEngine.sh run-wcm-admin-task-member-fixer
-DPortalAdminId=username -DPortalAdminPwd=password
-DWasUserId=username -DWasPassword=password -Dlibrary="MyLibrary"
-Dfix=true -DaltDn=update -DmismatchedId=update
```

**IBM i**

```
ConfigEngine.sh run-wcm-admin-task-member-fixer
-DPortalAdminId=username -DPortalAdminPwd=password
-DWasUserId=username -DWasPassword=password -Dlibrary="MyLibrary"
-Dfix=true -DaltDn=update -DmismatchedId=update
```

**z/OS**

```
./ConfigEngine.sh run-wcm-admin-task-member-fixer
-DPortalAdminId=username -DPortalAdminPwd=password
-DWasUserId=username -DWasPassword=password -Dlibrary="MyLibrary"
-Dfix=true -DaltDn=update -DmismatchedId=update
```

If there have been changes to users and groups that are within the specified realm
or that are not within any realm, update the items that reference them by entering
the following command:

**Windows**

```
ConfigEngine.bat run-wcm-admin-task-member-fixer
-DPortalAdminId=username -DPortalAdminPwd=password
-DWasUserId=username -DWasPassword=password -Dlibrary="MyLibrary"
-Drealm=MyRealm -Dfix=true -DnoRealmDn=true
```

**UNIX**

```
./ConfigEngine.sh run-wcm-admin-task-member-fixer
-DPortalAdminId=username -DPortalAdminPwd=password
-DWasUserId=username -DWasPassword=password -Dlibrary="MyLibrary"
-Drealm=MyRealm -Dfix=true -DnoRealmDn=true
```

**IBM i**

```
ConfigEngine.sh run-wcm-admin-task-member-fixer
-DPortalAdminId=username -DPortalAdminPwd=password
-DWasUserId=username -DWasPassword=password -Dlibrary="MyLibrary"
-Drealm=MyRealm -Dfix=true -DnoRealmDn=true
```

**z/OS**

```
./ConfigEngine.sh run-wcm-admin-task-member-fixer
-DPortalAdminId=username -DPortalAdminPwd=password
-DWasUserId=username -DWasPassword=password -Dlibrary="MyLibrary"
-Drealm=MyRealm -Dfix=true -DnoRealmDn=true
```
IBM WebSphere Portal comprises a framework of services to accommodate the different scenarios that portals need to address. Services are available for both WebSphere Portal and IBM Web Content Manager. You can configure some of these services.

### Member fixer with syndication

You can configure your system to automatically run the member fixer tool when syndicating. The member fixer is run on the subscriber during syndication. It is run against items that have just been syndicated. Details of the member fixer operations are included in the syndication report.

To run the member fixer during syndication add or update the following properties in the WCM WCMConfigService service using the WebSphere Integrated Solutions Console:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>deployment.fixMembers</td>
<td>To enable member fixing during syndication, set this parameter to <em>true</em>.</td>
</tr>
<tr>
<td>syndication.memberfixer.altDn</td>
<td>To update references to nonexistent users or groups with the portal administrator user distinguished name, set this parameter to <em>update</em>.</td>
</tr>
<tr>
<td></td>
<td>To remove references to nonexistent users or groups, set this parameter to <em>remove</em>.</td>
</tr>
<tr>
<td>syndication.memberfixer.invalidDn</td>
<td>To update references to users or groups that have invalid distinguished names with the portal administrator user distinguished name, set this parameter to <em>update</em>.</td>
</tr>
<tr>
<td></td>
<td>To remove references to users or groups that have invalid distinguished names, set this parameter to <em>remove</em>.</td>
</tr>
<tr>
<td>syndication.memberfixer.mismatchid</td>
<td>To fix references to users and groups with mismatched unique IDs, set this parameter to <em>update</em>.</td>
</tr>
<tr>
<td></td>
<td>To remove references to users and groups with mismatched unique IDs, set this parameter to <em>remove</em>.</td>
</tr>
<tr>
<td>syndication.memberfixer.fixCase</td>
<td>This parameter is used to define how to treat case differences when updating or fixing distinguished names.</td>
</tr>
<tr>
<td></td>
<td>To leave the case unchanged, set this parameter to <em>update</em>.</td>
</tr>
<tr>
<td></td>
<td>To convert the case to lower-case, set this parameter to <em>lower</em>.</td>
</tr>
<tr>
<td>syndication.memberfixer.realm</td>
<td>In a federated security environment with multiple realms, you must specify the name of the realm to run the member fixer against using this parameter.</td>
</tr>
</tbody>
</table>
Table 168. Member fixer syndication parameters (continued)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>syndication.memberfixer.norealmdn</code></td>
<td>In a federated security environment with multiple realms, the member fixer task can be used to check whether there are any users and groups referenced in items that are not under any of the base distinguished names defined for the realm and fix these references. To enable this, set this parameter to <code>true</code>.</td>
</tr>
</tbody>
</table>

Using the Update Security task

Use the Update Security task to apply inherited access permissions and remove existing item access permissions for all items or all items of a given type. This task is useful as a post-migration step, or if you are applying major changes to your inheritance settings.

Running the Update Security task

1. Open a command prompt.
2. Open a UNIX System Services (USS) command prompt.
3. To apply inherited access permissions to all items in a library named "MyLibrary" for all roles, run the following command from the `wp_profile_root/ConfigEngine` directory:

   Windows
   ```
   ConfigEngine.bat run-wcm-admin-task-update-security
   -DWasPassword=password -DPortalAdminId=username
   -DPortalAdminPwd=password -DWasPassword=password
   -Dlibrary=MyLibrary -DinheritPerms=apply -DlibSecurity=true
   ```

   UNIX
   ```
   ./ConfigEngine.sh run-wcm-admin-task-update-security
   -DWasPassword=password -DPortalAdminId=username
   -DPortalAdminPwd=password -DWasPassword=password
   -Dlibrary=MyLibrary -DinheritPerms=apply -DlibSecurity=true
   ```

   IBM i
   ```
   ConfigEngine.sh run-wcm-admin-task-update-security
   -DWasPassword=password -DPortalAdminId=username
   -DPortalAdminPwd=password -DWasPassword=password
   -Dlibrary=MyLibrary -DinheritPerms=apply -DlibSecurity=true
   ```

   z/OS
   ```
   ./ConfigEngine.sh run-wcm-admin-task-update-security
   -DWasPassword=password -DPortalAdminId=username
   -DPortalAdminPwd=password -Dlibrary=MyLibrary
   -DinheritPerms=apply -DlibSecurity=true
   ```

   Note: An administrator user name and password is not required if you have already specified the portal administrator user name and password using the `PortalAdminId` and `PortalAdminPwd` settings in the `wkplc.properties` file.

4. To remove inherited access permissions to all items in a library named "MyLibrary" for all roles, run the following command:

   Windows
   ```
   ConfigEngine.bat run-wcm-admin-task-update-security
   -DPortalAdminId=username -DPortalAdminPwd=password
   -DWasPassword=password -Dlibrary=MyLibrary -DinheritPerms=remove
   ```
UNIX  ./ConfigEngine.sh run-wcm-admin-task-update-security
       -DPortalAdminId=username -DPortalAdminPwd=password
       -DWasPassword=password -Dlibrary=MyLibrary -DinheritPerms=remove

IBM i  ConfigEngine.sh run-wcm-admin-task-update-security
       -DPortalAdminId=username -DPortalAdminPwd=password
       -DWasPassword=password -Dlibrary=MyLibrary -DinheritPerms=remove

z/OS  ./ConfigEngine.sh run-wcm-admin-task-update-security
       -DPortalAdminId=username -DPortalAdminPwd=password
       -DWasPassword=password -Dlibrary=MyLibrary -DinheritPerms=remove

Note: An administrator user name and password is not required if you have
already specified the portal administrator user name and password using the
PortalAdminId and PortalAdminPwd settings in the wkplc.properties file.

5. To remove existing item access permissions for all items in a library named
   "MyLibrary" for all roles, run the following command:

Windows
  ConfigEngine.bat run-wcm-admin-task-update-security
  -DPortalAdminId=username -DPortalAdminPwd=password
  -DWasPassword=password -Dlibrary=MyLibrary
  -DremoveExistingPerms=true

UNIX  ./ConfigEngine.sh run-wcm-admin-task-update-security
       -DPortalAdminId=username -DPortalAdminPwd=password
       -DWasPassword=password -Dlibrary=MyLibrary
       -DremoveExistingPerms=true

IBM i  ConfigEngine.sh run-wcm-admin-task-update-security
       -DPortalAdminId=username -DPortalAdminPwd=password
       -DWasPassword=password -Dlibrary=MyLibrary
       -DremoveExistingPerms=true

z/OS  ./ConfigEngine.sh run-wcm-admin-task-update-security
       -DPortalAdminId=username -DPortalAdminPwd=password
       -DWasPassword=password -Dlibrary=MyLibrary
       -DremoveExistingPerms=true

Note: An administrator user name and password is not required if you have
already specified the portal administrator user name and password using the
PortalAdminId and PortalAdminPwd settings in the wkplc.properties file.

Running the Update Security task for all libraries

You can run the Update Security task for all libraries by replacing the option
-Dlibrary=libraryName with the option -DallLibraries=true in the command. If
neither option is specified, the Update Security task will process the default library.

Restricting the task to only update specified items types

You can restrict which objects types are processed by appending
-DrestrictOn=ItemType to the command.

For example:
- content
- folder
- project
- style for presentation templates
- template for authoring templates
- taxonomy
- category
- SiteArea
- Workflow
- WorkflowStage
- WorkflowAction
- Cmpnt for components

If not specified, the security of all object types will be updated.

**Running the task on a virtual portal**

When running this task on a virtual portal you must either add 
-DVirtualPortalHostName=name or -DVirtualPortalContext=context to the 
command.

**Preserving dates**

You can preserve the last modified date of items updated by the Update Security 
task by adding -DpreserveDates=true to the command. Otherwise the last 
modified date will be updated when the Update Security task is run.

**Defining the session timeout**

To prevent your session timing out before the task has finished, you can append 
the option -DsessionTimeOut=timeOut to the command. This sets the number of 
seconds in which the task must complete before its session will timeout. The default session timeout is 14,440 seconds, which is 4 hours. For large repositories 
you should increase this setting. For example: -DsessionTimeOut=36000, which is 10 
hours.

**Examples**

All of the options can be combined. For instance, to remove existing item access 
permissions and apply inherited access permissions to Content in the a library 
called 'MyLibrary', whilst preserving the last modified dates of the items, run the 
following command:

**Windows**

`ConfigEngine.bat run-wcm-admin-task-update-security`  
-DWasPassword=password -Dlibrary=MyLibrary  
-DremoveExistingPerms=true -DinheritPerms=apply -DrestrictOn=Content  
-DpreserveDates=true

**UNIX**

`./ConfigEngine.sh run-wcm-admin-task-update-security`  
-DWasPassword=password -Dlibrary=MyLibrary  
-DremoveExistingPerms=true -DinheritPerms=apply -DrestrictOn=Content  
-DpreserveDates=true

**IBM i**

`ConfigEngine.sh run-wcm-admin-task-update-security`  
-DWasPassword=password -Dlibrary=MyLibrary  
-DremoveExistingPerms=true -DinheritPerms=apply -DrestrictOn=Content  
-DpreserveDates=true`
Using the workflow update tool

Use the workflow update tool to add a workflow to existing items that aren’t already workflow enabled.

You must first enable the workflow update tool by adding the following parameters to the WCM WCMConfigService service using the WebSphere Integrated Solutions Console:

- `connect.businesslogic.module.workflowenablement.class=com.aptrix.pluto.workflow.WorkflowEnablementModule`
- `connect.businesslogic.module.workflowenablement.remoteaccess=true`
- `connect.businesslogic.module.workflowenablement.autoload=false`

1. Log in to the portal as an administrator.
2. Open the following URL in the browser and specify which workflow you want to apply and the library containing the items you want to apply the workflow to:
   `http://[HOST]:[PORT]/wps/wcm/myconnect/?MOD=workflowenablement&library=libraryname&workflow=workflowname&fix=true`

   **Note:** If the "library" parameter is omitted, the default library that has been configured in the WCM WCMConfigService service is used.

   **Note:** If the "&fix=true" parameter is omitted, the tool will run in read-only mode and generate a report.

**Specifying a workflow stage:**

You can specify the workflow stage to move the updated items to by adding `&workflowstage=workflowstagename` to the URL. The stage specified here must have a status of published. You cannot assign items to stages with a status of draft. If not specified, items will be assigned to the first stage with a status of published.

**Preserving dates:**

You can preserve the last modified date of items updated by the Workflow update tool by adding `&preserve_dates=true` to the URL used to run the Workflow update tool.

**Restricting which items types to fix:**

You can restrict which objects types are processed by adding `&restrictOn=itemtype` to the URL used to run the Workflow update tool. For example:

- `content`
- `style` for presentation templates
- `template` for authoring templates
- `taxonomy`
- `category`
- `SiteArea`
- `Cmpnt` for components

If not specified, all object types will be fixed.
library
Enter a library name. If the library parameter is omitted, the default library that has been configured in the WCM WCMConfigService service using the WebSphere Integrated Solutions Console.

To run this tool against all libraries you instead use $allLibraries=true. If you have a large number of libraries, this may take a long time to run, so it may be better to run this tool against individual libraries instead of all libraries.

Unlocking items:
To force locked items to be unlocked while running the tool, add $forceUnlock=true to the query. This setting defaults to true.

Restricting which items types to fix:
To prevent your server timing out before the workflow update tool has finished, you can specify $sessionTimeout= to the URL. This is defined as the number of seconds before a session will timeout. For example:

$sessionTimeout=36000. The default session timeout is 14440 seconds.

Running the tool on a virtual portal

There are two methods available when running the tool on a virtual portal:

Using the URL context of a virtual portal:
If your virtual portal has a URL context, you can add this to the URL

http://[HOST]:[PORT]/wps/wcm/myconnect/[url_context]?MOD=workflowenablement&fix=true

Using the hostname of a virtual portal:
If your virtual portal has a hostname you can add this to the URL

http://[Virtual_HOST]:[PORT]/wps/wcm/myconnect?MOD=workflowenablement&fix=true

Related tasks:
Setting service configuration properties
IBM WebSphere Portal comprises a framework of services to accommodate the different scenarios that portals need to address. Services are available for both WebSphere Portal and IBM Web Content Manager. You can configure some of these services.

Clearing item history

You use the clear history tool to clear the history of an item.

You must first enable the clear history tool by adding the following parameters to the WCM WCMConfigService service using the WebSphere Integrated Solutions Console:

- connect.businesslogic.module.clearhistory.class=com.aptrix.history.ClearHistoryModule
- connect.businesslogic.module.clearhistory.remoteaccess=true
- connect.businesslogic.module.clearhistory.autoload=false

1. Log in to the portal as an administrator.
2. Open the following URL in the browser and specify details of what history details to clear:

http://[HOST]:[PORT]/wps/wcm/myconnect?MOD=ClearHistory&day=date&month=month&year=year&keep=number_of_entries&restrictOn=item_type&library=library_name

   day, month and year
   The history details will be cleared prior to the date specified in the day,
month and year parameters. If no date is specified, then the date will
default to one year before the current date.

keep Specify the minimum number of history entries to keep. For example, if
an item has not been updated for over a year, and you specify to clear
all history entries more than a year old, but choose to keep the last five
entries, all the history will be cleared except for the last five entries
even though they are over a year old. If a number is not specified, then
the minimum number of history entries to keep will default to 10.

restrictOn
Select the item types to run the clear history tool against. If no item
types are specified, all item types will be processed. Use the following
parameters for each item-type:

• content
• folder
• project
• style for presentation templates
• template for authoring templates
• taxonomy
• category
• SiteArea
• Workflow
• WorkflowStage
• WorkflowAction
• Cmpnt for components

library
Enter a library name. If the "library" parameter is omitted, the default
library that has been configured in the WCM WCMConfigService service is
used.

To run this tool against all libraries you instead use
alllibraries=true. If you have a large number of libraries, this may
take a long time to run, so it may be better to run this tool against
individual libraries instead of all libraries.

fix If omitted or set to false, a report listing which history items will be
cleared is displayed. If set to true, history items will be cleared as
specified.

Note: You cannot completely clear item history. One history item will always
remain in an item no matter what parameters you select when clearing the item
history.

Running the tool on a virtual portal

There are two methods available when running the tool on a virtual portal:

Using the URL context of a virtual portal:
If your virtual portal has a URL context, you can add this to the URL
http://[HOST]:[PORT]/wps/wcm/myconnect/[url_context]?MOD=ClearHistory&fix=true

Using the hostname of a virtual portal:
If your virtual portal has a hostname you can add this to the URL
http://[Virtual_HOST]:[PORT]/wps/wcm/myconnect?MOD=ClearHistory&fix=true
Related tasks:

**Setting service configuration properties**

IBM WebSphere Portal comprises a framework of services to accommodate the different scenarios that portals need to address. Services are available for both WebSphere Portal and IBM Web Content Manager. You can configure some of these services.

---

**Clearing version history**

You use the clear versions tool to clear the version history of an item.

You must first enable the clear versions tool by adding the following parameters to the WCM WCMConfigService service using the WebSphere Integrated Solutions Console:

- `connect.businesslogic.module.clearversions.class=com.aptrix.versioncontrol.ClearVersionsModule`
- `connect.businesslogic.module.clearversions.remoteaccess=true`
- `connect.businesslogic.module.clearversions.autoload=false`

1. Log in to the portal as an administrator.
2. Open the following URL in the browser and specify details of what history details to clear:

   ```
   http://[HOST]:[PORT]/wps/wcm/myconnect?MOD=ClearVersions&day=day&month=month&year=year&keep=number_of_entries&restrictOn=item_type&library=library
   ```

   **day, month and year**
   
   The version history will be cleared prior to the date specified in the day, month and year parameters. If no date is specified, then the date will default to one year before the current date.

   **keep**
   Specify the minimum number of history versions to keep. For example, if a version has not been created for over a year, and you specify to clear all versions more than a year old, but choose to keep the last five versions, all versions will be cleared except for the last five versions even though they are over a year old. If a number is not specified, then the minimum number of versions to keep will default to 10.

   **restrictOn**
   Select the item types to run the clear versions tool against. If no item types are specified, all item types will be processed. Use the following parameters for each item-type:
   
   - `content`
   - `style` for presentation templates
   - `template` for authoring templates
   - `taxonomy`
   - `category`
   - `SiteArea`
   - `Workflow`
   - `WorkflowStage`
   - `WorkflowAction`
   - `Cmpnt` for components

   **library**
   Enter a library name. If the library parameter is omitted, the default library that has been configured in the WCM WCMConfigService service using the WebSphere Integrated Solutions Console.
To run this tool against all libraries you instead use &allLibraries=true. If you have a large number of libraries, this may take a long time to run, so it may be better to run this tool against individual libraries instead of all libraries.

fix  If omitted or set to false, a report listing which versions will be cleared is displayed. If set to true, versions will be cleared as specified.

Note: You cannot completely clear all versions. One version of an item will always remain no matter what parameters you select when clearing the version history.

Running the tool on a virtual portal

There are two methods available when running the tool on a virtual portal:

Using the URL context of a virtual portal:
   If your virtual portal has a URL context, you can add this to the URL
   http://[HOST]:[PORT]/wps/wcm/myconnect/[url_context]?MOD=ClearVersions

Using the hostname of a virtual portal:
   If your virtual portal has a hostname you can add this to the URL
   http://[Virtual_HOST]:[PORT]/wps/wcm/myconnect?MOD=ClearVersions

Resetting the web content event log

From time to time you may need to reset the web content event log. The event log can be reset only on a syndicator server. Any changes made by resetting the event log are then syndicated to its corresponding subscribers. In most cases you reset the event log on the server you have imported or migrated data onto, or on a syndicator to troubleshoot syndication problems in a syndication relationship.

You must first enable the reset event log module by adding the following parameters to the WCM WCMConfigService service using the WebSphere Integrated Solutions Console:

• connect.businesslogic.module.reseteventlog.class=com.ibm.workplace.wcm.services.eventlog.ResetEventLogModule
• connect.businesslogic.module.reseteventlog.remoteaccess=true
• connect.businesslogic.module.reseteventlog.autoload=false

You should reset the web content event log under these circumstances:
• The contents of the repository have been modified through an external mechanism such as a JCR import or some other custom application.
• As a post migration step during migration prior to syndication.
• In order to troubleshoot syndication problems such as items on the syndicator not being sent.

Note:
• Before resetting the web content event log you should edit the wkplc_dbtype.properties file and ensure the DbSafeMode property is set to false. This is located under wp_profile_root/ConfigEngine/properties
• In clustered environments, you should only reset the event log on the primary node.
• Any objects that were “purged” on the syndicator since the last syndication will not be purged on the subscriber. Purged objects are lost since the event log does
not maintain records of objects that were deleted. To clean up purged items on a subscriber, you will need to go the subscriber and manually delete them.

For Windows, UNIX and IBM i.
Run the run-wcm-admin-task-reset-event-log task from the wp_profile_root/ConfigEngine directory.

Windows
ConfigEngine.bat run-wcm-admin-task-reset-event-log
-Dlibrary=library_name -Dfix=true

UNIX ./ConfigEngine.sh run-wcm-admin-task-reset-event-log
-Dlibrary=library_name -Dfix=true

IBM i ConfigEngine.sh run-wcm-admin-task-reset-event-log
-Dlibrary=library_name -Dfix=true

Note: If -Dfix=true is omitted, then the task will run in report-mode only.

Note: When running this task on a virtual portal you must either add -DVirtualPortalHostName=name or -DVirtualPortalContext=context to the command.

For z/OS:
Submit the EJPSWCMR job to reset the event log. This job is created when you create the jobs for installing the product with the Customization Dialog.

Related tasks:
“Exporting and importing web content libraries” on page 599

IBM Web Content Manager provides two methods for exporting and importing web content libraries: an export or import that operates on one library, and an export or import that enables you to work with a separate copy of a library. With either method, you can export the contents of a web content library to disk and import this data into another web content server. If you're working with a copy of a library, you can also import that library into the same web content server multiple times, resulting in a new library after each import without affecting previous copies. Exporting and importing libraries enables you to make a backup copy of a web content library and can also be used to move data between servers. However, this function cannot be used to send updates, deletes and moves. It is only suitable for populating new items.

Using the export cache settings task

Use the export cache settings task to display a summary of the current cache settings of your system.

When you run the export cache settings task, a summary of your cache settings is generated and set to the SystemOut.log. This includes the type of cache being used, and how it is being applied. For example, basic caching per session, or data caching per site.

Running the export cache settings task

1. Open a command prompt.
2. Open a UNIX System Services (USS) command prompt.
3. Run the following command from the wp_profile_root/ConfigEngine directory:

Windows
ConfigEngine.bat run-wcm-admin-task-export-cache-settings
UNIX  
```
./ConfigEngine.sh run-wcm-admin-task-export-cache-settings
-DWasPassword=password  -DPortalAdminId=username
-DPortalAdminPwd=password  -Dlibrary=MyLibrary
-DinheritPerms=apply  -DlibSecurity=true
```

IBM i  
```
ConfigEngine.sh run-wcm-admin-task-export-cache-settings
-DWasPassword=password  -DPortalAdminId=username
-DPortalAdminPwd=password  -Dlibrary=MyLibrary
-DinheritPerms=apply  -DlibSecurity=true
```

z/OS  
```
./ConfigEngine.sh run-wcm-admin-task-export-cache-settings
-DWasPassword=password  -DPortalAdminId=username
-DPortalAdminPwd=password  -Dlibrary=MyLibrary
-DinheritPerms=apply  -DlibSecurity=true
```

Note: An administrator user name and password is not required if you have already specified the portal administrator user name and password using the PortalAdminId and PortalAdminPwd settings in the wkplc.properties file.

### Displaying cache settings in a browser

You can also display your cache settings in a browser using the following URL:


**Related concepts:**

"Cache expire parameters" on page 238

You use the "expires" parameter in IBM Web Content Manager tags and URLs to specify how long to maintain data in the cache before it is expired. Once data expires from a cache, the next request for the data will be retrieved from the original server. The expires parameter is not mandatory.

### Exporting and importing web content libraries

IBM Web Content Manager provides two methods for exporting and importing web content libraries: an export or import that operates on one library, and an export or import that enables you to work with a separate copy of a library. With either method, you can export the contents of a web content library to disk and import this data into another web content server. If you're working with a copy of a library, you can also import that library into the same web content server multiple times, resulting in a new library after each import without affecting previous copies. Exporting and importing libraries enables you to make a backup copy of a web content library and can also be used to move data between servers. However, this function cannot be used to send updates, deletes and moves. It is only suitable for populating new items.

Before you begin, create an empty shared directory to hold the exported web content library. If moving data between servers, both systems must have write access to this directory. In addition, review the following considerations before exporting or importing web content libraries:

**Importing libraries into different versions**

You can import libraries from a different version of Web Content Manager so long as the version you are importing the library into is later than the version you exported the library from. For example:
• you can import a library exported from version 6.1.0.1 into version 7.0
• you cannot import a library exported from version 7.0 into version 6.1.0.1

It is recommended that you upgrade to the latest version of each release before attempting to import libraries between versions. It is not possible to export libraries from releases before 6.0.

Exporting and importing a web content library versus syndication.
This feature does not replace the syndication feature. Although this feature can be used to transfer data between servers, it is a manual process and is not meant to be used for regular updates between servers. Syndication is instead used to automatically keep two or more servers synchronized. Also, whereas syndication can be used to send updates, deletes and moves, the import feature is only suitable for populating new items.

Limitations of exporting and importing a web content library.
• Saved versions of items are not exported. Only the current version of each item is exported.
• Children are only exported and imported when the parent is successfully exported and imported.
• If an item exists on the target server with the same path, name and ID, then the item is overwritten.
• Library and item level access controls remain unchanged when a library is exported and imported. You need to run the member fixer tool on the imported library to fix references to missing users and groups.
• You cannot import an item if an item on the target server has the same ID but a different parent than the item being imported.

Disabling JCR text search.
It is recommended you disable JCR text search indexing on your WebSphere Portal server before exporting or importing large libraries to reduce the load on the database during export and import. Edit the 
wp_profile_root/PortalServer/jcr/lib/com/ibm/icm/icm.properties file and set the jcr.textsearch.enabled property to false. After the file is updated, restart your server for the changes to take effect. When you have completed exporting or importing your library you must then enable JCR text search again. It can take some time to rebuild the indexes when you re-enable JCR text search indexing.

Exporting and importing large libraries
• When importing web content libraries, a temporary directory is used to store the library files during the upload process. If the size of the uploaded files exceeds the available disk space for the temporary directory, the import operation fails. When uploading large libraries, ensure that there is sufficient disk space to accommodate the import. The location of the temporary directory is specified by the jcr.binaryValueFileDir property in the wp_profile_root/PortalServer/jcr/lib/com/ibm/icm/icm.properties file.
• When exporting or importing large libraries, increase the total transaction lifetime timeout and the maximum transaction timeout of your server to 360 seconds through the WebSphere Integrated Solutions Console. To change these settings, go to Servers > Server Types > WebSphere application servers > portal_server > Container Services > Transaction Service.
Personalization components.
Personalization rules created within a Personalization component are exported and imported along with your web content library.

If you are using Personalization rules created directly in the Personalization portlet you need to export and import your rules to and from Personalization on the same servers as your web content by using the same process as moving WebSphere Portal content from a staging system to a production system. Personalization export and import must be performed before exporting and importing web content.

JSP components
If you are using JSP components you must manually copy any related JSP files to and from the same servers that you are exporting and importing to.

Related tasks:
“Resetting the web content event log” on page 597
From time to time you may need to reset the web content event log. The event log can be reset only on a syndicator server. Any changes made by resetting the event log are then syndicated to its corresponding subscribers. In most cases you reset the event log on the server you have imported or migrated data onto, or on a syndicator to troubleshoot syndication problems in a syndication relationship.

“Exporting and importing a web content library”
You can export the contents of a web content library to disk and import this data into another web content server. This feature enables you to make a backup copy of a web content library, and can also be used to move data between servers. This function cannot be used to send updates, deletes, and moves. It is only suitable for populating new items.

“Exporting and importing a web content library in a z/OS environment” on page 605
You can export the contents of a web content library to disk and import this data into another web content server. This feature enables you to make a backup copy of a web content library, and can also be used to move data between servers. This function cannot be used to send updates, deletes and moves. It is only suitable for populating new items.

“Exporting and importing a web content library copy” on page 607
You can export the contents of a web content library to disk by creating a copy of the web content library. By working with an exported copy, you can import the copied library into the same web content server multiple times, resulting in a new library after each import without affecting previous copies. This is a quick way of creating new libraries that are fully populated with web content that you can easily adapt for other purposes.

Setting service configuration properties
IBM WebSphere Portal comprises a framework of services to accommodate the different scenarios that portals need to address. Services are available for both WebSphere Portal and IBM Web Content Manager. You can configure some of these services.

Exporting and importing a web content library
You can export the contents of a web content library to disk and import this data into another web content server. This feature enables you to make a backup copy of a web content library, and can also be used to move data between servers. This function cannot be used to send updates, deletes, and moves. It is only suitable for populating new items.
Follow these steps to export and import a web content library. The server that the data is being exported from is called the source server, and the server that the data is being imported into is called the target server.

1. **Exporting:**
   a. Log in to the WebSphere Integrated Solutions Console on the source server.
   b. Click **Resources > Resource Environment > Resource Environment Providers > WCM WCMConfigService > Custom properties.**

   **Cluster note:** If you are using this web content server as part of a cluster, ensure that you use the WebSphere Integrated Solutions Console for the deployment manager when manipulating configuration properties.
   c. Create or update the export properties.

   **export.directory**
   The directory on the source server where the exported data is written. The export task creates a subdirectory with the name corresponding to library name within this directory for each exported library. The default value is `${USER_INSTALL_ROOT}/PortalServer/wcm/ilwwcm/system/export`.

   **export.libraryname**
   The name of the web content library to transfer. If exporting multiple libraries enter each library name separated by a semi-colon. For example, `export.library=Lib_1;Lib_2;Lib_3`

   **export.singledirectory**
   If set to `true`, multiple libraries are written into a single directory specified by the `export.directory` property. If set to `false`, the export task created subdirectories with the name corresponding to each exported library names. For example, if `export.directory` is specified as `C:\export` and the library name is `Web Library`, the export task saves the exported library under `C:\export\Web Library`. Set this property to `true` when exporting multiple libraries that contain references between each library.

   **Note:** You must restart your server any time you change these settings.

   d. Export the web content library from the source server:
   - Open a command prompt on the source server.
   - Run the `export-wcm-data` task from the `wp_profile_root/ConfigEngine` directory.

   **Windows**
   ```
   ConfigEngine.bat export-wcm-data -DWasPassword=password
   -DPortalAdminPwd=password
   ```

   **UNIX**
   ```
   ./ConfigEngine.sh export-wcm-data -DWasPassword=password
   -DPortalAdminPwd=password
   ```

   **IBM i**
   ```
   ConfigEngine.sh export-wcm-data -DWasPassword=password
   -DPortalAdminPwd=password
   ```

   By default, this task is performed on the base portal. To run this task on a different virtual portal, identify the virtual portal by adding one of the following parameters to the command line. Each parameter requires the prefix `-D` on the command line.
VirtualPortalHostName
Specify the host name of the virtual portal. For example, vp.example.com.

VirtualPortalContext
Specify the virtual portal context that identifies the virtual portal. For example, vp1.

Note:
- You can override the export.directory property defined in the WCM WCMConfigService service by using the -Dexport.directory parameter. For example: export-wcm-data -Dexport.directory=c:\export
- You can override the export.singledirectory property defined in the WCM WCMConfigService service by using the -Dexport.singledirectory parameter. For example: export-wcm-data -Dexport.singledirectory=false saves the exported libraries under different directories.
- You can override the export.libraryname property defined in the WCM WCMConfigService service by using the -Dexport.libraryname parameter. For example: export-wcm-data -Dexport.libraryname=libraryname
- You can override the export.libraryname property defined in the WCM WCMConfigService service by adding the option -Dexport.allLibraries parameter to export all libraries. If this option is used, the export may take a long time to finish.

Important: To ensure that your exported libraries can be successfully imported, do not change the names of any of the folders or files within the exported data.

e. Verify that this transfer step completed without errors. If any errors occurred, check the portal logs on the target server for extended diagnostic information.

f. Verify that the export directories were populated correctly, including any subdirectories for each exported library.

2. Importing:
a. Log in to the WebSphere Integrated Solutions Console on the target server.
b. Click Resources > Resource Environment > Resource Environment Providers > WCM WCMConfigService > Custom properties.

Cluster note: If you are using this web content server as part of a cluster, ensure that you use the WebSphere Integrated Solutions Console for the deployment manager when manipulating configuration properties.

c. Create or update the import.directory property. This is the directory from where the exported data is read when importing the data to the target server. If exporting and importing across a network, this property can be the same directory as the one specified in export.directory property. Otherwise, you must copy the exported data from the location specified in the export.directory property to the location specified in the import.directory property before running the import task in step 2.

- If you specified true for the export.singledirectory property when you exported your libraries, specify the parent directory where all the exported libraries are located.
- If you specified false for the export.singledirectory property when you exported your libraries, or if you only want to import specific libraries,
then you must list the directory of each library separated by semicolons. For example: c:\import\Lib1;c:\import\Lib2;c:\import\Lib3. If using Linux use /; to separate each library, such as /opt/importdata/Lib1;/opt/importdata/Lib2;/opt/importdata/Lib3.

**Note:** You must restart your server any time you change this setting.

d. Import the web content library to the target server.
   - Open a command prompt on the target server.
   - Run the import-wcm-data task from the `wp_profile_root/ConfigEngine` directory.

   **Windows**
   ```
   ConfigEngine.bat import-wcm-data -DWasPassword=password
   -DPortalAdminPwd=password
   ```

   **UNIX**
   ```
   ./ConfigEngine.sh import-wcm-data -DWasPassword=password
   -DPortalAdminPwd=password
   ```

   **IBM i**
   ```
   ConfigEngine.sh import-wcm-data -DWasPassword=password
   -DPortalAdminPwd=password
   ```

   By default, this task is performed on the base portal. To run this task on a different virtual portal, identify the virtual portal by adding one of the following parameters to the command line. Each parameter requires the prefix `-D` on the command line.

   **VirtualPortalHostName**
   Specify the host name of the virtual portal. For example, `vp.example.com`.

   **VirtualPortalContext**
   Specify the virtual portal context that identifies the virtual portal. For example, `vp1`.

   **Note:** You can override the `import.directory` property defined in the WCM `WCMConfigService` service by using the `-Dimport.directory` parameter. For example: `import-wcm-data -Dimport.directory=c:\import\Lib1;c:\import\Lib2;c:\import\Lib3`.

**Differences in paths between versions:**

When exporting from a version 6.1 system, you can specify a folder to export the library to:
```
/opt/61/folder/jcr_root
```

When importing into version 8.0 from version 6.1, the `jcr_root` is not required to be specified in the import path:
```
/opt/61/folder/
```

When exporting from versions 7.0 or higher, the following structure is used:
```
/opt/70/folder1/folder2
```

When exported, `folder2` is automatically generated.

When importing into version 8.0 from version 7.0 or higher, `folder2` is not required to be specified in the import path:
```
/opt/70/folder1/
```
Verify that the imported libraries have been imported by reviewing the list of libraries listed in the web content libraries section of the administration portlet on the target server. If any errors occurred, check the portal logs on the target server for extended diagnostic information.

f. Reset the web content event log.

g. Restart the server.

Troubleshooting:

- If items are exported and imported twice between the same servers, and items have been moved or deleted between the first and second export and import, then you must manually delete these items from the target server before transferring the items again. If this step is not performed, an error like this example is generated:

```
javax.jcr.ItemExistsException: A node already exists with uuid: 376dba00408608aea231b2c714d0bda6 at path: /contentRoot/icm:libraries[10]/F1/F3/test1.ort
```

- If you receive 500 errors on ext2 and ext3 versions of Linux, you have exceeded the number of children that a parent folder can hold. You cannot store more than 32768 children under one folder on ext2 and ext3 versions of Linux. Move some content items out of the affected site area to another site area so that none of your site areas contain more than 32768 children under one folder and then try exporting again. You can move the content items back to the correct site areas once you have imported the library.

Related tasks:

“Exporting and importing web content libraries” on page 599

IBM Web Content Manager provides two methods for exporting and importing web content libraries: an export or import that operates on one library, and an export or import that enables you to work with a separate copy of a library. With either method, you can export the contents of a web content library to disk and import this data into another web content server. If you’re working with a copy of a library, you can also import that library into the same web content server multiple times, resulting in a new library after each import without affecting previous copies. Exporting and importing libraries enables you to make a backup copy of a web content library and can also be used to move data between servers. However, this function cannot be used to send updates, deletes and moves. It is only suitable for populating new items.

**Exporting and importing a web content library in a z/OS environment**

You can export the contents of a web content library to disk and import this data into another web content server. This feature enables you to make a backup copy of a web content library, and can also be used to move data between servers. This function cannot be used to send updates, deletes and moves. It is only suitable for populating new items.

Follow these steps to export and import a web content library. The server that the data is being exported from is called the source server, and the server that the data is being imported into is called the target server:

1. To export libraries from the source server:
   a. Start the WebSphere Portal customization dialog.
   b. In the Portal configuration panel, select Application configuration tasks.
   c. Select Configure Web Content Management.
d. Select **Export a web content library**.

e. Select **Define variables**.

**Remember**: Press F1 to display the help panel if you need assistance defining the variables.

f. Generate the customization jobs.

g. Follow the customization dialog instructions for submitting the customization jobs.

h. Verify that the export directories were populated correctly, including any subdirectories for each exported library.

**Note**: To ensure that your exported libraries can be successfully imported, do not change the names of any of the folders or files within the exported data.

2. To import libraries to the target server:

a. Start the WebSphere Portal customization dialog.

b. In the **Portal configuration** panel, select **Application configuration tasks**.

c. Select **Configure Web Content Management**.

d. Select **Import a web content library**.

e. Select **Define variables**.

**Remember**: Press F1 to display the help panel if you need assistance defining the variables.

f. Generate the customization jobs.

g. Follow the customization dialog instructions for submitting the customization jobs.

h. Verify that the imported libraries have been imported by reviewing the list of libraries listed in the web content libraries section of the administration portlet on the target server. If any errors occurred, check the portal logs on the target server for extended diagnostic information.

**Troubleshooting**:

- If items are exported and imported twice between the same servers, and items have been moved or deleted between the first and second export and import, then you must manually delete these items from the target server before transferring the items again. If this step is not performed, an error like this example is generated:

  javax.jcr.ItemExistsException: A node already exists with uuid: 376dba00408608aea231b2c714d0bda6 at path: /contentRoot/icm:libraries[10]/F1/F3/test1.ort
IBM Web Content Manager provides two methods for exporting and importing web content libraries: an export or import that operates on one library, and an export or import that enables you to work with a separate copy of a library. With either method, you can export the contents of a web content library to disk and import this data into another web content server. If you’re working with a copy of a library, you can also import that library into the same web content server multiple times, resulting in a new library after each import without affecting previous copies. Exporting and importing libraries enables you to make a backup copy of a web content library and can also be used to move data between servers. However, this function cannot be used to send updates, deletes and moves. It is only suitable for populating new items.

Exporting and importing a web content library copy

You can export the contents of a web content library to disk by creating a copy of the web content library. By working with an exported copy, you can import the copied library into the same web content server multiple times, resulting in a new library after each import without affecting previous copies. This is a quick way of creating new libraries that are fully populated with web content that you can easily adapt for other purposes.

Although many aspects are the same for the standard export and import and the copy export and import, there are some important differences:
- When you export a library as a copy, new IDs are generated for all items in the library. This ensures that there are no conflicts with existing libraries or items when you import the copy into a web content server that already contains the original library. In this way, you can perform multiple imports to the same web content server, resulting in a new library for each import.
- The configuration tasks (export-library-copy and import-library-copy) that work on library copies use properties that can either be added to the wkplc.properties file or manually appended to the command line, for easier scripting.

Follow these steps to export or import a copy of a web content library. The server that the data is being exported from is called the source server, and the server that the data is being imported into is called the target server.
- **Exporting:**
  1. Open a command prompt on the source server.
  2. Run the export-library-copy task from the `wp_profile_root/ConfigEngine` directory.

  **Windows**
  ```
  ConfigEngine.bat export-library-copy
  -DLibraryPath=path_to_export_file
  -DLibraryName=library_name_to_export -DWasPassword=password
  -DPortalAdminPwd=password
  ```

  **UNIX**
  ```
  ./ConfigEngine.sh export-library-copy
  -DLibraryPath=path_to_export_file
  -DLibraryName=library_name_to_export -DWasPassword=password
  -DPortalAdminPwd=password
  ```

  **IBM i**
  ```
  ConfigEngine.sh export-library-copy
  ```
The following properties must be specified either on the command line or in the wkplc.properties file.

Note: If you are specifying properties in the wkplc.properties file, it is not necessary to put quotation marks (" ) around values that contain spaces. These quotation marks are only required when specifying property values on the command line.

LibraryPath
The directory path and file name used to store the exported library. The export process creates a compressed archive file, so it is recommended that you specify a file extension such as .zip. If you are exporting and importing across a network, this location can be a network drive accessible by both the source and target servers.

LibraryName
The name of the web content library to copy. If you are exporting multiple libraries, separate each library name by a semi-colon (;). For example, LibraryName="Web Content;Samples".

WasUserid
The administrator ID for WebSphere Application Server.

WasPassword
The administrator password for WebSphere Application Server.

PortalAdminId
The administrator ID for WebSphere Portal.

PortalAdminPwd
The administrator password for WebSphere Portal.

By default, this task is performed on the default virtual portal. To run this task on a different virtual portal, identify the virtual portal by adding one of the following parameters to the command line. Each parameter requires the prefix -D on the command line.

VirtualPortalHost
Specify the host name of the virtual portal. For example, vp.example.com.

Important: If the host name of the virtual portal is the same as the host name of the default virtual portal, you must also specify the VirtualPortalContext property. You can specify the VirtualPortalHost property by itself only if the host name is unique.

VirtualPortalContext
Specify the virtual portal context that identifies the virtual portal. For example, vp1.

Example commands:
- Windows: ConfigEngine.bat export-library-copy -DLibraryPath=C:\wcm_export\webcontent.zip -DLibraryName="Web Content" -DWasPassword=mypassword -DPortalAdminPwd=mypassword -DVirtualPortalHost=vp.example.com
- UNIX: ./ConfigEngine.sh export-library-copy -DLibraryPath=/opt/wcm_export/webcontent.zip -DLibraryName="Web Content"
-DWasPassword=mypassword -DPortalAdminPwd=mypassword
-DVirtualPortalHost=vp.example.com

- IBM i: ConfigEngine.sh export-library-copy -DLibraryPath=/opt/
wcm_export/webcontent.zip -DLibName="Web Content"
-DWasPassword=mypassword -DPortalAdminPwd=mypassword
-DVirtualPortalHost=vp.example.com

3. Verify that this transfer step completed without errors. If any errors occurred,
check the portal logs on the source server for extended diagnostic
information.

• Importing:
1. Open a command prompt on the target server.
2. Run the import-library-copy task from the wp_profile_root/ConfigEngine
directory.

Windows
    ConfigEngine.bat import-library-copy
    -DLibraryPath=path_to_export_file
    -DLibName=library_name_to_import -DWasPassword=password
    -DPortalAdminPwd=password

UNIX ./ConfigEngine.sh import-library-copy
    -DLibraryPath=path_to_export_file
    -DLibName=library_name_to_import -DWasPassword=password
    -DPortalAdminPwd=password

IBM i ConfigEngine.sh import-library-copy
    -DLibraryPath=path_to_export_file
    -DLibName=library_name_to_import -DWasPassword=password
    -DPortalAdminPwd=password

The following properties must be specified either on the command line or in
the wkplc.properties file.

Note: If you are specifying properties in the wkplc.properties file, it is not
necessary to put quotation marks (") around values that contain spaces.
These quotation marks are only required when specifying property values on
the command line.

Differences in paths between versions:

When exporting from a version 6.1 system, you can specify a folder to export
the library to:
/opt/61/folder/jcr_root

When importing into version 8.0 from version 6.1, the jcr_root is not required
to be specified in the import path:
/opt/61/folder/

When exporting from versions 7.0 or higher, the following structure is used:
/opt/70/folder1/folder2

When exported, folder2 is automatically generated.

When importing into version 8.0 from version 7.0 or higher, folder2 is not
required to be specified in the import path:
/opt/70/folder1/
LibraryPath
The directory path and file name containing the library to be imported. If you are exporting and importing across a network, the value for this property can be the same file path that was used for the LibraryPath property during the export process. Otherwise, you must copy the exported data to a location accessible by the target server before attempting to import.

LibraryName
The name to use for the web content library copy that you are importing. If you are importing multiple libraries, separate each new library name by a semi-colon (;). For example, LibraryName="Web Content Copy;Samples Copy".

LibraryExportName
The sequence of library names used during the original export, as defined by the LibraryName property specified for the export process. For example, LibraryExportName="Web Content;Samples".
This property enables the import process to correctly set the new library names for the imported copies, in conjunction with the LibraryName property specified for the import process. The LibraryExportName property is only required if you are importing multiple libraries at one time.

WasUserid
The administrator ID for WebSphere Application Server.

WasPassword
The administrator password for WebSphere Application Server.

PortalAdminId
The administrator ID for WebSphere Portal.

PortalAdminPwd
The administrator password for WebSphere Portal.

By default, this task is performed on the default virtual portal. To run this task on a different virtual portal, identify the virtual portal by adding one of the following parameters to the command line. Each parameter requires the prefix -D on the command line.

VirtualPortalHost
Specify the host name of the virtual portal. For example, vp.example.com.

Important: If the host name of the virtual portal is the same as the host name of the default virtual portal, you must also specify the VirtualPortalContext property. You can specify the VirtualPortalHost property by itself only if the host name is unique.

VirtualPortalContext
Specify the virtual portal context that identifies the virtual portal. For example, vp1.

The following properties are optional and can be specified either on the command line or in the wkplc.properties file:

LibraryTitle
The title to use for the web content library copy that you are
importing. If you are importing multiple libraries, separate each new library title by a semi-colon (;). For example, LibraryTitle="Web Content Title;Samples Title".

**LibraryDescription**
The description to use for the web content library copy that you are importing. If you are importing multiple libraries, separate each new library description by a semi-colon (;). For example, LibraryDescription="Copy of Web Content library;Copy of Samples library".

**LibraryNameTextProvider**
This property specifies the name of the text provider to use to locate the translated title of the library that you are importing. If you are importing multiple libraries with different text providers, separate each provider name by a semi-colon (;). For example, LibraryNameTextProvider=provider1;provider2.

**LibraryNameTextProviderKey**
This property specifies the key within the associated text provider that identifies the translated title of the library that you are importing. If you are importing multiple libraries with different translated titles, separate each key by a semi-colon (;). For example, LibraryNameTextProviderKey=key1;key2.

**LibraryBaseLocale**
The locale used when importing the web content library copy.

Example commands:
- **Windows:** ConfigEngine.bat import-library-copy -DLibraryPath=C:\wcm_import\webcontent.zip -DLibraryName="Web Content Copy" -DLibraryTitle="Web Content Copy Title" -DLibraryDescription="Copy of Web Content library" -DLibraryNameTextProvider=provider -DLibraryNameTextProviderKey=key -DLibraryBaseLocale=en -DWaspPassword=mypassword -DPortalAdminPwd=mypassword
- **UNIX:** ./ConfigEngine.sh import-library-copy -DLibraryPath=/opt/wcm_import/webcontent.zip -DLibraryName="Web Content Copy" -DLibraryTitle="Web Content Copy Title" -DLibraryDescription="Copy of Web Content library" -DLibraryNameTextProvider=provider -DLibraryNameTextProviderKey=key -DLibraryBaseLocale=en -DWaspPassword=mypassword -DPortalAdminPwd=mypassword
- **IBM i:** ConfigEngine.sh import-library-copy -DLibraryPath=/opt/wcm_import/webcontent.zip -DLibraryName="Web Content Copy" -DLibraryTitle="Web Content Copy Title" -DLibraryDescription="Copy of Web Content library" -DLibraryNameTextProvider=provider -DLibraryNameTextProviderKey=key -DLibraryBaseLocale=en -DWaspPassword=mypassword -DPortalAdminPwd=mypassword

3. Verify that the imported libraries have been imported by reviewing the list of libraries listed in the web content libraries section of the administration portlet on the target server. If any errors occurred, check the portal logs on the target server for extended diagnostic information.
4. Reset the web content event log.
Example commands for exporting and importing multiple libraries

When exporting and importing multiple web content libraries with a single command, the following considerations apply:

- For properties such as LibraryName that reference multiple libraries, separate the respective values for that property with a semi-colon (;).
- The value of the LibraryExportName property must match the value of the LibraryName property used during the export process to indicate the sequence of libraries.

Windows

- Export: ConfigEngine.bat export-library-copy -DLibraryPath=C:\wcm_export\webcontent.zip -DLibraryName="Web Content;Samples" -DWasPassword=mypassword -DPortalAdminPwd=mypassword
- Import: ConfigEngine.bat import-library-copy -DLibraryPath=C:\wcm_import\webcontent.zip -DLibraryName="Web Content Copy;Samples Copy" -DLibraryExportName="Web Content;Samples" -DLibraryTitle="Web Content Copy Title;Samples Copy Title" -DLibraryBaseLocale=en -DWasPassword=mypassword -DPortalAdminPwd=mypassword

UNIX

- Export: ./ConfigEngine.sh export-library-copy -DLibraryPath=/opt/wcm_export/webcontent.zip -DLibraryName="Web Content;Samples" -DWasPassword=mypassword -DPortalAdminPwd=mypassword
- Import: ./ConfigEngine.sh import-library-copy -DLibraryPath=/opt/wcm_import/webcontent.zip -DLibraryName="Web Content Copy;Samples Copy" -DLibraryExportName="Web Content;Samples" -DLibraryTitle="Web Content Copy Title;Samples Copy Title" -DLibraryBaseLocale=en -DWasPassword=mypassword -DPortalAdminPwd=mypassword

IBM i

- Export: ConfigEngine.sh export-library-copy -DLibraryPath=/opt/wcm_export/webcontent.zip -DLibraryName="Web Content;Samples" -DWasPassword=mypassword -DPortalAdminPwd=mypassword
- Import: ConfigEngine.sh import-library-copy -DLibraryPath=/opt/wcm_import/webcontent.zip -DLibraryName="Web Content Copy;Samples Copy" -DLibraryExportName="Web Content;Samples" -DLibraryTitle="Web Content Copy Title;Samples Copy Title" -DLibraryBaseLocale=en -DWasPassword=mypassword -DPortalAdminPwd=mypassword
Related tasks:

“Exporting and importing web content libraries” on page 599

IBM Web Content Manager provides two methods for exporting and importing web content libraries: an export or import that operates on one library, and an export or import that enables you to work with a separate copy of a library. With either method, you can export the contents of a web content library to disk and import this data into another web content server. If you’re working with a copy of a library, you can also import that library into the same web content server multiple times, resulting in a new library after each import without affecting previous copies. Exporting and importing libraries enables you to make a backup copy of a web content library and can also be used to move data between servers. However, this function cannot be used to send updates, deletes and moves. It is only suitable for populating new items.

Cloning a web content repository

Syndicating items from one server to another, either after migration or to roll out a new server, can take a long time. Your database backup and restore features can be used to clone data from one repository to another, making your system ready for syndication to be used from then on for incremental updates.

There are two basic cloning scenarios:

- Cloning all items from one server to another. For example, cloning data from one authoring server to another authoring server.
- Cloning all items from one server to a another, but then removing unrequired data from the cloned server. For example, cloning data from an authoring server to a delivery server where you would want to remove version history and draft items from the delivery server repository.

Note: These procedures only describe how to clone a web content repository. To clone a Portal environment, XMLAccess export and import should be used to transfer the Portal data to the target environment.

Cloning preparation

You must prepare your source and target systems prior to cloning a web content repository.

- The source and target environments must be on the same version and fix level
- Ideally the source and target environments will use the same LDAP
- If the target server already contains data:
  - If you need to use this data later, ensure you take a backup of the target environment before cloning.
  - Note down the syndicator and subscriber setups. The syndication setup on the target environment will be lost during the cloning process, and will need to be recreated.
  - Note down the library access control setup. Library access levels for target environment will be lost during the cloning process, and will need to be reapplied.

Oracle Considerations

When setting up an Oracle database for JCR you would ideally have a separate physical Oracle database for each JCR repository. This will make it very simple to copy a JCR repository from one system to another.
If you do choose to store all of your JCR repositories in a single database then it is best to use the same schema name for each, even if it will be difficult to tell which JCR instance each schema belongs too.

If you instead choose to name the schemas differently for each JCR repository instance, copying will be much more difficult. You would have to do a schema export and import first, then a second export of views and triggers into "sqlexport". Then modify the source schema name with the target schema name in the “sqlexport” and finally make another import of the “sqlexport” into the target schema. This is not the recommended approach.

**Cloning data**

These procedures describe how to clone web content data from one system to another.

1. On the source system:
   a. Disable all syndicators
   b. Stop the Portal server
   c. Backup the data of JCR database name specified by "jcr.DbName" parameter in the wkplc_comp.properties file:

<table>
<thead>
<tr>
<th>System</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows</td>
<td>located in the \wp_profile_root\ConfigEngine\properties directory</td>
</tr>
<tr>
<td>UNIX</td>
<td>located in the \wp_profile_root\ConfigEngine/properties directory</td>
</tr>
<tr>
<td>IBM i</td>
<td>located in the \wp_profile_root\ConfigEngine/properties directory</td>
</tr>
</tbody>
</table>

   Refer to the documentation for your database platform for specific backup instructions.
   d. Restart the Portal server.
   e. Re-enable any syndicators that do not syndicate to the target server.

2. On the target system:
   a. Stop the Portal server.
   b. Remove the existing JCR database by dropping the database. Refer to the documentation for your database platform for specific instructions.
   c. Create a new database, and restore the source system database backup. Refer to the documentation for your database platform for specific instructions.
   d. Restart the Portal server.
   e. Delete all syndicators and subscribers as they will not be valid for the target system.
   f. If the target system is using a different LDAP run the member fixer tool to fix member information to match the new LDAP. First run the module in report mode to see what member information requires fixing and then run the tool in fix mode to fix various potential member information mismatches.
   g. If cloning from an authoring system to a delivery system, run the clear versions tool to remove any versions from the delivery system.
   h. Set up syndication:
Create subscribers and syndicators as appropriate for this system.
If the target system is a syndicator to the original source system, leave that syndicator disabled for now. All other Syndicators can be enabled.

3. On source system:
   a. If the target system is a subscriber to the source system, update the syndicator with the new target subscriber ID and enable syndication.
   b. If the target system is a syndicator to the source system, update the subscriber with the new target syndicator ID.

4. On target system:
   a. If the target system is a syndicator to the source system, enable syndication.

Once you have finished cloning your web content data:
• Validate that the web content data on the target environment is rendering correctly.
• Validate access control settings for both rendering and authoring are set as expected, and working correctly for a selection of users.
• Validate updates are being syndication into and from the target environment as expected.

Related tasks:
“Clearing version history” on page 596
You use the clear versions tool to clear the version history of an item.
Chapter 8. Developing

You can extend the standard features of IBM Web Content Manager using the Web Content Manager API and JSP files.

The IBM Web Content Manager API

You can use the Web Content Manager API to extend functions of Web Content Manager.

- You can use the Web Content Manager API to:
  - create items
  - delete items
  - move items
  - copy items
  - retrieve items
  - approve items in a workflow
  - search for items

The Javadoc documentation can be reviewed for a complete set of the features available using the API. The Javadoc HTML files are located under the `was_profile_root\installedApps\nodename\wcm.ear\ilwwcm.war\webinterface\folder`.

Using the API

The workspace is the heart of the IBM Web Content Manager API. Items are created, saved, deleted and searched for in the workspace item. A workspace is basically an interface to Web Content Manager that is associated with a user. Using a workspace item, the user can perform operations as that user.

To get a workspace item, you must first retrieve the WebContentService:

```java
try {
    // Construct and initial Context
    InitialContext ctx = new InitialContext();

    // Retrieve WebContentService using JNDI name
    WebContentService webContentService = (WebContentService) ctx.lookup("portal:service/wcm/WebContentService");
} catch (NamingException ne) {
    System.out.print("Naming Exception: "+ ne);
}
```

You then request one from the repository singleton with the following call:
webContentService.getRepository().getWorkspace("my username", "my password");

To get a workspace item without specifying a user name and password, use one of the following calls:

- When used in a portlet: `Workspace workspace = webContentService.getRepository().getWorkspace((Principal)portletRequest.getUser());`
- When not used in a portlet: `Workspace workspace = webContentService.getRepository().getWorkspace((Principal)request.getUserPrincipal());`

If the user is not recognized as a Web Content Manager user, or for some other reason could not be authenticated, an "OperationFailedException" will be thrown.

**Note:** Only Web Content Manager users (including external LDAP users if enabled) are recognized. For example, a workspace cannot be retrieved using an LTPA token.

Operations available on the workspace include:

- Searching for items with the provided "findBy" methods.
- Creating new items of available editable types.
- Saving and deleting editable items.

You must call `endWorkspace()` when finished with the workspace item.

```java
webContentService.getRepository().endWorkspace();
```

**Note:** You don't need to call `endWorkspace()` when using a JSP component as rendering and session management is handled by Web Content Manager.

**Note:** You use the `setCurrentDocumentLibrary` method to make calls library-specific. If not specified, the default library that has been configured in the WCM `WCMConfigService` service is used.

**Related tasks:**
- Setting service configuration properties

IBM WebSphere Portal comprises a framework of services to accommodate the different scenarios that portals need to address. Services are available for both WebSphere Portal and IBM Web Content Manager. You can configure some of these services.

**Web Content Manager JSP tags**

You use IBM Web Content Manager JSP tags with the Web Content Manager API to pull Web Content Manager content and components into external JSP applications.

**Note:** A JSP referenced within a JSP component must not include a reference, directly or indirectly, to the same JSP component. This includes references within Web Content Manager tags or the API. If it does, a loop is created and your server will crash.

**Note:** To use the Web Content Manager JSP tags, the following directive must be provided in the JSP:

```html
<%@ taglib uri="/WEB-INF/tld/wcm.tld" prefix="wcm" %>
```
Storing JSP Files:

JSP files can be located:
- within the `was_profile_root`/installedApps/node-name/wcm.ear/ilwwcm.war directory of your server.
- within any other web application running on portal. When referencing JSP files in another web application, use the following path: `contextPath;jspPath`

For example: `/wps/customapplication;/jsp/editor.jsp`

Writing JSP to be referenced within a JSP component:

The `setExplicitContext` and `setContext` tags are not required when displaying a JSP file with a JSP Component. They are only required when directly accessing a JSP file.

Reloading JSP files:

JSP files referenced by Web Content Manager are reloaded once every 10 seconds. If you update a JSP file, you may need to wait for it to be reloaded before your changes will be displayed.

InitWorkspace tag

This is used to set the initial workspace.

```
<wcm:initworkspace username=" " password=" ">
    <Error Message>
</wcm:initworkspace>
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>username</code></td>
<td>The user name of a valid Web Content Manager user.</td>
</tr>
<tr>
<td><code>password</code></td>
<td>The password for the valid Web Content Manager user.</td>
</tr>
</tbody>
</table>

Explicit context tag

This sets the path to your Web Content Manager server. This is not required in JSP that is displayed via a JSP component.

```
<wcm:setExplicitContext wcmWebAppPath=" " wcmServletPath=" " path=" ">
    requestParameters" " prefix=" " project=" " >
    </wcm:setExplicitContext>
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>wcmWebAppPath</code></td>
<td>The URL up to the web application. For example: <code>http://localhost:10040/wps/wcm</code></td>
</tr>
<tr>
<td><code>wcmServletPath</code></td>
<td>The servlet path to the Web Content Manager servlet. For example: <code>/connect</code></td>
</tr>
</tbody>
</table>
Table 171. Parameters (continued)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>path</td>
<td>The path to the content and site areas. For example: /Site Area A/ Site Area B/Content C</td>
</tr>
<tr>
<td>requestParameters</td>
<td>You specify java Map request parameters to set in the context. These parameters can be used by menu components that are rendered via the JSP that use a query string.</td>
</tr>
<tr>
<td>project</td>
<td>The name of the project to set in the context. If the corresponding project cannot be found, it will be ignored and an error will be logged. An empty string is used to clear any project previously set in the context.</td>
</tr>
</tbody>
</table>

**Note:** The `project`, `wcmWebAppPath` and `wcmServletPath` parameters are optional. However, if `wcmWebAppPath` is specified, `wcmServletPath` must also be specified.

Developers can add insert context tags at any place in the page and it will change the context for the rest of the page execution, but the tags cannot be nested.

**Context retrieval tag**

Sets the context given the location of a path string. This is not required in JSP that is displayed via a JSP component.

```xml
<wcm:setContext location=" " wcmWebAppPath=" " wcmServletPath=" " param=" " project=" " defaultPath=" ">
[Error Message]
</wcm:setContext>
```

Table 172. Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>location</td>
<td>This sets the context of the location of a path string. Either:</td>
</tr>
<tr>
<td></td>
<td>location=&quot;query&quot;</td>
</tr>
<tr>
<td></td>
<td>The context is obtained from the query parameter.</td>
</tr>
<tr>
<td></td>
<td>location=&quot;request&quot;</td>
</tr>
<tr>
<td></td>
<td>The context is obtained from the value of the request.</td>
</tr>
<tr>
<td></td>
<td>location=&quot;session&quot;</td>
</tr>
<tr>
<td></td>
<td>The context is obtained from the value of the current session.</td>
</tr>
<tr>
<td></td>
<td>location=&quot;portalContext&quot;</td>
</tr>
<tr>
<td></td>
<td>This is used to define the path of a site area or content item that will be used as the current context of a page. For example: /library1/sitearea3/content4</td>
</tr>
<tr>
<td></td>
<td>location=&quot;portalMapping&quot;</td>
</tr>
<tr>
<td></td>
<td>This is used to define the path of a site area or content item that will be used as the default site area of a page. For example: /library1/sitearea3</td>
</tr>
<tr>
<td>wcmWebAppPath</td>
<td>The URL up to the web application. For example: <a href="http://localhost:10040/wps/wcm">http://localhost:10040/wps/wcm</a></td>
</tr>
<tr>
<td>wcmServletPath</td>
<td>The servlet path to the Web Content Manager servlet. For example: /connect</td>
</tr>
<tr>
<td>param</td>
<td>This is the name of the parameter that the path string will be in.</td>
</tr>
</tbody>
</table>
Table 172. Parameters (continued)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>project</td>
<td>The name of the project to set in the context. If the corresponding project cannot be found, it will be ignored and an error will be logged. An empty string is used to clear any project previously set in the context.</td>
</tr>
<tr>
<td>defaultPath</td>
<td>If the location parameter does not resolve to a valid location, then the value of the defaultPath is used. For example: /library2/sitearea1</td>
</tr>
</tbody>
</table>

Note: The project, wcmWebAppPath, wcmServletPath, and defaultPath parameters are optional. However, if wcmWebAppPath is specified, wcmServletPath must also be specified.

Developers can add context tags at any place in the page and it will change the context for the rest of the page execution, but the tags cannot be nested.

Rendering tags

These are equivalent to element and component tags.

Rendering an element from the current site area, or content item

    <wcm:contentComponent type=" " key=" " >
    [Error Message]
    </wcm:contentComponent>

Table 173. Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>This determines where the element is being referenced from. Either content or sitearea.</td>
</tr>
<tr>
<td>key</td>
<td>This is the name of the element being referenced.</td>
</tr>
</tbody>
</table>

    <wcm:libraryComponent name=" " library=" " >
    [Error Message]
    </wcm:libraryComponent>

Rendering a component from the Component Library

Table 174. Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>This is the name of the component being referenced.</td>
</tr>
<tr>
<td>library</td>
<td>This is the name of the library where the component is stored.</td>
</tr>
</tbody>
</table>

For example:

    <wcm:libraryComponent name="SC Menu Events" library="Showcase" />
    You do not have access to this item.
    </wcm:libraryComponent>

Rendering Content based on the current context of a page

    <wcm:content pageDesign=" " >
    [Error Message]
    </wcm:content>
Table 175. Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>pageDesign</td>
<td>This name of the Presentation Template used to determine context. This parameter is optional.</td>
</tr>
</tbody>
</table>

**Error handling**

The following tag can be added to error messages to enable error handling:

```html
<%=error%>
```

**Plugin tag**

Rendering plug-ins can be referenced within JSP code using a plugin tag:

```html
<wcm:plugin name=" " param1="value" param2="value2">
// Your text.
</wcm:plugin>
```

See [Creating a plug-in tag](#) for further information.

**Web content library management APIs**

You can perform various web content library functions using the Web content API.

- Create a library
- Delete a library
- Copy a library
- Import a library
- Export a library

**Note:** Drafts are not copied or exported when using the API to copy or export libraries.

The Web Content Manager Javadoc documentation should be reviewed for a complete set of the features available using the API. The Javadoc HTML files are located under the `was_profile_root\installedApps\nodename\wcm.ear\ilwwcm.war\webinterface` folder.

**Invoking web content library API methods asynchronously**

Although Web content library API functions can be invoked synchronously, if you run these against web content libraries containing large amounts of data, they may take extremely long to complete execution. For example, if these methods are invoked from a JSP page, this may result in the JSP page being invalidated due to a session timeout.

WebSphere Application Server uses a mechanism known as asynchronous beans. An asynchronous bean is a Java object that can be run asynchronously. The "Work object" asynchronous bean is used to invoke web content library API methods asynchronously.

A Work object (which is represented by the interface `com.ibm.websphere.asynchbeans.Work`) extends `java.langRunnable`. It is used to run a block of code as an independent thread. WebSphere Application Server
maintains a pool of independent threads that can be assigned to run code encapsulated in Work instances. This pool of threads is managed by the WorkManager. This is used to spawn threads to run Work objects and to monitor them. WebSphere Application Server maintains default Work Managers for each of the application servers that are contained on a particular node. The sample in this topic makes use of the default Work Manager (wm/wpsWorkManager) for the WebSphere Portal application server. This maintains a pool of 300 threads. It is possible to create new Work manager instances with customized thread pools. This is done using the WebSphere Integrated Solutions Console for the WebSphere Portal server.

The example uses the DeleteWork class to implement the Work interface:

```java
group deletesample;
import com.ibm.workplace.wcm.api.*;
import javax.naming.*;
public class DeleteWork implements com.ibm.websphere.asynchbeans.Work {
    private String m_username = null;
    private String m_password = null;
    private String m_libraryToDelete = null;
    public DeleteWork(String username, String password, String library) {
        m_username = username;
        m_password = password;
        m_libraryToDelete = library;
    }
    public void release() {
    }
    public void run() {
        try {
            // Construct and initial Context
            InitialContext ctx = new InitialContext();

            // Retrieve WebContentService and WebContentLibraryService using JNDI name
            WebContentService wcs = (WebContentService) ctx.lookup("portal:service/wcm/WebContentService");
            WebContentLibraryService wcls = (WebContentLibraryService) ctx.lookup("portal:service/wcm/WebContentLibraryService");
            Workspace ws = wcs.getRepository().getWorkspace(m_username, m_password);
            DocumentLibrary docLib = ws.getDocumentLibrary(m_libraryToDelete);
            LibraryTaskResult res = wcls.deleteLibrary(ws, docLib);

            // Once you get the result object back, print status to StandardOut
            if (res.getResultType() == ResultTypes.OPERATION_SUCCESS) {
                System.out.println("Successfully Deleted Library " + m_libraryToDelete);
            } else {
                System.out.println("Failed To Delete Library " + m_libraryToDelete);
            }
        } catch (Exception e) {
        }
    }
}
```
The run method is what is required in order to implement this interface. This is where you wrap the library method that you want to run in a thread separate from the calling thread. In the previous example, DeleteWork is instantiated passing in credentials as well as the library to be deleted. In run(), the repository is logged into and a Workspace instance is obtained as is a DocumentLibrary corresponding to the library that is to be deleted. deleteLibrary() is then called to perform the actual deletion. Once this method is completed, the Result object can be queried to determine the status of the deletion. This can then be logged or processed as required.

The following JSP file is used to invoke the DeleteWork object:

```jsp
<%@ page import="java.util.*" %>
<%@ page import="java.io.*" %>
<%@ page import="java.lang.*" %>
<%@ page import="com.ibm.workplace.wcm.api.*" %>
<%@ page import="com.aptrix.identity.*" %>
<%@ page import="com.ibm.workplace.wcm.services.library.*" %>
<%@ page import="com.ibm.workplace.wcm.services.*" %>
<%@ page import="com.ibm.workplace.wcm.services.repository.*" %>
<%@ page import="com.ibm.websphere.asynchbeans.*" %>
<%@ page import="javax.naming.*" %>
<%@ page import="deletesample.DeleteWork" %>

<%@ page import="java.util.*" %>
<%@ page import="java.io.*" %>
<%@ page import="java.lang.*" %>
<%@ page import="com.ibm.workplace.wcm.api.*" %>
<%@ page import="com.aptrix.identity.*" %>
<%@ page import="com.ibm.workplace.wcm.services.library.*" %>
<%@ page import="com.ibm.workplace.wcm.services.*" %>
<%@ page import="com.ibm.workplace.wcm.services.repository.*" %>
<%@ page import="com.ibm.websphere.asynchbeans.*" %>
<%@ page import="javax.naming.*" %>
<%@ page import="deletesample.DeleteWork" %>

<% try {
    //obtain a work manager instance - the work manager manages a pool of threads which can be used to invoke
    //the functionality encapsulated within a work instance
    InitialContext ctx = new InitialContext();
    com.ibm.websphere.asynchbeans.WorkManager wm =
    (com.ibm.websphere.asynchbeans.WorkManager) ctx.lookup("wm/wpsWorkManager");

    //create a new work instance
    DeleteWork workItem = new DeleteWork(request.getParameter("username"), request.getParameter("password"), request.getParameter("library"));
    //spawn a thread to run the create work instance
    wm.startWork(workItem);

} catch (Exception e) {
    <%= e.toString() %>
}%>
```

In order to run a Work object, it is necessary to do a JNDI lookup to obtain the default WebSphere Portal Server Work Manager instance. Once this is done, the DeleteWork class can be instantiated. To run DeleteWork on a separate thread, call startWork() on the WorkManager passing in the DeleteWork instance. For example,
wm.startWork(workItem); The System.out log can be checked to see when DeleteWork finishes.

Syndication APIs

You can perform various syndication functions using the web content API.

- Retrieve, enable, or disable syndicators and subscribers.
- Start full and partial syndication updates.
- Dynamically review syndication progress.
- View syndicator and subscriber details such as the name of the libraries being syndicated.
- Query syndication details such as which items were updated, saved, modified or removed.

The Web Content Manager Javadoc information should be reviewed for a complete set of the features available using the API. The Javadoc HTML files are located under the \was_profile_root\installedApps\nodeName\wcm.ear\ilwcm.war\webinterface\ folder.

Converting an IBM API web content viewer to the JSR 286 API

As installed by default, the web content viewer is based on the JSR 286 API. If you have a web content viewer based on the older IBM API, you can convert the viewer to the JSR 286 API. Use the convert-wcm-rendering-portlet task to convert the IBM API web content viewer settings and instances to the JSR 286 web content viewer portlet.

The convert-wcm-rendering-portlet task converts portlet settings of the IBM API web content viewer to portlet preferences of the JSR 286 web content viewer. The task also converts instances of the IBM API web content viewer to instances of the web content viewer. User customized portlet data that is associated with the portlet instance is converted into portlet preferences.

To convert the instances and settings of the IBM API web content viewer to the JSR 286 viewer, complete the following steps:

1. Update the file \wp_profile_root\ConfigEngine\wkplc.properties. Confirm that the user IDs and passwords are set as required or modify them if necessary.
2. Update or verify the properties in the file \wp_profile_root\PortalServer\wcm\config\portletconversion.properties.

**Note:** If the following conditions are true, no changes are required and you can use the default values in the properties file.

- There are no clones of the IBM API web content viewer.
- You want to convert all instances of the portlet on all pages in the default virtual portal.

For specific situations, you can update the additional properties described in the following table.
### Scenario Properties to modify

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Properties to modify</th>
</tr>
</thead>
<tbody>
<tr>
<td>You want to convert instances of the portlet on specific pages in the default virtual portal.</td>
<td>pages.uniquename&lt;br&gt;Specify a list of unique names of pages, separated by commas. If you specify this property, only portlets on these pages and their descendants are converted. If this property is empty or missing, instances of the IBM API web content viewer on all pages are converted.</td>
</tr>
<tr>
<td>You want to convert instances of the portlet in a virtual portal that is not the default virtual portal.</td>
<td>xmlaccess.url&lt;br&gt;Specify the URL of the virtual portal to the portal XML configuration interface servlet. You can use this property to run conversions for specific virtual portals. If this property is empty or missing, the default portal is used to run the conversion. Example: xmlaccess.url=<a href="http://www.example.com:10039/wps/config/vp1">http://www.example.com:10039/wps/config/vp1</a></td>
</tr>
<tr>
<td>You cloned the web content viewer portlet and want to convert instances of the clone.</td>
<td>Identify the clone by specifying one of the following properties:&lt;br&gt;• ibmportlet.portletname&lt;br&gt;• ibmportlet.uniquename&lt;br&gt;• ibmportlet.objectid&lt;br&gt;Only one of the properties is required to identify the portlet.&lt;br&gt;For a complete list of properties for the portlet conversion task, see Converting portlet instances and settings from the IBM API to the standard API.</td>
</tr>
</tbody>
</table>

3. Change to the directory `wp_profile_root/ConfigEngine`.
4. Run the task `ConfigEngine convert-wcm-rendering-portlet`.
   - **Windows**<br>`ConfigEngine.bat convert-wcm-rendering-portlet`
   - **UNIX**<br>`./ConfigEngine.sh convert-wcm-rendering-portlet`
   - **IBM i**<br>`ConfigEngine.sh convert-wcm-rendering-portlet`
   - **z/OS**<br>`./ConfigEngine.sh convert-wcm-rendering-portlet`
5. Verify the conversion by reviewing the console. The message **Build successful** indicates a successful conversion. If the message **Build failed** is displayed upon completion of the task, review the previous steps.
6. Verify the configuration of the converted web content viewer. For more information about configuring a local web content viewer, see the portlet help.
7. After successful conversion, you can uninstall the IBM API web content viewer.
Related tasks:

Converting portlet instances and settings from the IBM API to the standard API

The portal provides a portlet conversion task that allows you to convert the settings and instances of IBM API portlets to the corresponding standard API portlets. This is useful when you intend to replace IBM API portlets by standard API portlets.

Using remote actions

Remote actions are used to trigger actions from the IBM Web Content Manager application.

You can reference remote actions using plugin tags using the following format:

```
[plugin:RemoteAction action="" docid=""
dialog="" dialogSize="" dialogTitle="" useCurrentContext="" showInfoMsg="" ]
```

- **action**: This is the remote action to perform.
- **docid**: This is the document ID of the item to run the remote action against.
- **useCurrentContext**: If set to true, then the document ID is obtained from the rendering context instead of the docid attribute.
- **dialog**: If set to true, when rendered within a web content viewer portlet the remote action is rendered as a URL that redirects the user to a hidden portal page that is used by the web content viewer for inline editing.
- **dialogSize**: This optional setting defines the size of the dialog executing the remote action. The value must be in the format "width,height". For example, `dialogSize="200,300"` for a dialog of width 200 pixel and a height of 300 pixel. If omitted, the dialog size is calculated from the content displayed in the dialog. This setting is only used if dialog="true".
- **dialogTitle**: This optional setting sets the title of the dialog executing the remote action. If omitted, the action name is used instead. This setting is only used if dialog="true".
- **showInfoMsg**: Set this to true to display success status and other information messages after the remote action has finished. If omitted, this parameter is set to false and only warning and error status messages are displayed. This setting can only be used if dialog="true".

Remote actions can also be appended to the URL of an authoring portlet. For example:

```
http://[host]/wps/myportal/wcmAuthoring?wcmAuthoringAction=action&param=value
```

You can also append remote actions to the URL of a local web Content Viewer portlet. This can be useful in sites that feature inline editing of content items.

Custom authoring interfaces: There are limitations to the functionality delivered using remote actions. For example, remote actions only support plain text. You cannot use remote actions to add markup into elements such as HTML elements. To create custom authoring interfaces, use the Web Content Manager API in combination with remote actions.
Remote action types

`new` This is used to open a new item form. You must also specify a "type" parameter.

For example:

- `[plugin:RemoteAction action="new"
  type="com.ibm.workplace.wcm.api.WCM_Content"
  atid="com.ibm.workplace.wcm.api.WCM_ContentTemplate/ID1"]`

The following type parameters can be used:

- com.ibm.workplace.wcm.api.WCM_ContentTemplate
- com.ibm.workplace.wcm.api.WCM_SiteAreaTemplate
- com.ibm.workplace.wcm.api.WCM_Category
- com.ibm.workplace.wcm.api.WCM_Content
- com.ibm.workplace.wcm.api.WCM_DateComponent
- com.ibm.workplace.wcm.api.WCM_FileComponent
- com.ibm.workplace.wcm.api.WCM_HTMLComponent
- com.ibm.workplace.wcm.api.WCM_ImageComponent
- com.ibm.workplace.wcm.api.WCM_NumericComponent
- com.ibm.workplace.wcm.api.WCM_PresentationTemplate
- com.ibm.workplace.wcm.api.WCM_Project
- com.ibm.workplace.wcm.api.WCM_ProjectTemplate
- com.ibm.workplace.wcm.api.WCM_RichTextComponent
- com.ibm.workplace.wcm.api.WCM_ShortTextComponent
- com.ibm.workplace.wcm.api.WCM_SiteArea
- com.ibm.workplace.wcm.api.WCM_Taxonomy
- com.ibm.workplace.wcm.api.WCM_TextComponent
- com.ibm.workplace.wcm.api.WCM_Workflow
- com.ibm.workplace.wcm.api.WCM_WorkflowStage

When creating a new content item or site area, you can specify an authoring template by providing the document ID of the authoring template in the `atid` parameter:

- `[plugin:RemoteAction action="new"
  type="com.ibm.workplace.wcm.api.WCM_Content"
  atid="com.ibm.workplace.wcm.api.WCM_ContentTemplate/ID1"]`

- `[plugin:RemoteAction action="new"
  type="com.ibm.workplace.wcm.api.WCM_Project"
  atid="com.ibm.workplace.wcm.api.WCM_ProjectTemplate/ID1"]`

**Note:** If no authoring template is specified for a new site area, the default site area template is used.

When creating a new project, you can specify a project template by providing the document ID of the project template in the `atid` parameter:
• [plugin:RemoteAction action="new"
  type="com.ibm.workplace.wcm.api.WCM_Project"
  atid="com.ibm.workplace.wcm.api.WCM_ProjectTemplate/ID1"]

**Note:** If no project template is specified for a new project, the default project template is used.

When creating a new site area or project, you can add an atselection="true" parameter to allow users to select a template themselves:

• [plugin:RemoteAction action="new"
  type="com.ibm.workplace.wcm.api.WCM_SiteArea" atselection="true"]
• [plugin:RemoteAction action="new"
  type="com.ibm.workplace.wcm.api.WCM_Project" atselection="true"]

**Note:** When creating content items, if no content template is specified, users will automatically be prompted to select a content template.

When creating site areas, taxonomies, authoring templates, presentation templates, component types and workflow items, you can specify the library to save the new item under using the library parameter:

• [plugin:RemoteAction action="new"
  type="com.ibm.workplace.wcm.api.WCM_Taxonomy" library="ABC"]

If the library is not specified in the URL, it will be set as the current library, which can be set using the API method `Workspace.setCurrentDocumentLibrary()`. If the current library is not set, the default library will be used, as set using the `defaultLibrary` property in the WCMConfigService.

When creating site areas, content items and categories, you can specify the document ID of the parent item to save the new item under. Specify this ID in the `pid` parameter:

• [plugin:RemoteAction action="new"
  type="com.ibm.workplace.wcm.api.WCM_Content"
  pid="com.ibm.workplace.wcm.api.WCM_SiteArea/ID"]
• [plugin:RemoteAction action="new"
  type="com.ibm.workplace.wcm.api.WCM_SiteArea"
  pid="com.ibm.workplace.wcm.api.WCM_SiteArea/ID"]
• [plugin:RemoteAction action="new"
  type="com.ibm.workplace.wcm.api.WCM_Category"
  pid="com.ibm.workplace.wcm.api.WCM_Taxonomy/ID"]

When creating content items or site areas you can specify the position of the new site area using a `position` parameter. You can specify to save the new item at the start or end relative to any existing site areas:

• [plugin:RemoteAction action="new"
  type="com.ibm.workplace.wcm.api.WCM_Content" position="start"
  pid="com.ibm.workplace.wcm.api.WCM_SiteArea/ID1"]
• [plugin:RemoteAction action="new"
  type="com.ibm.workplace.wcm.api.WCM_SiteArea" position="end"
  pid="com.ibm.workplace.wcm.api.WCM_SiteArea/ID1"]

**delete** This is used to delete an item. You must also specify the `docid` of the item.

For example:
**edit** This is used to open an item form in edit mode. You must also specify the docid of the item.

For example:

- [plugin:RemoteAction action="edit"
docid="com.ibm.workplace.wcm.api.WCM_Content/ID1"]

**read** This is used to open an item form in read-only mode. You must also specify the docid of the item.

For example:

- [plugin:RemoteAction action="read"
docid="com.ibm.workplace.wcm.api.WCM_Content/ID1"]

**openmainview**

This is used to open a view within an authoring portlet. The following parameters are used with this parameter to open different views of the library explorer.

To open an "All Items" view, you must specify the filter to use when opening the "All Items" view. For example, to open the "Content by Title" view in "All Items".

- [plugin:RemoteAction action="openmainview" view="contentbytitle"]

The following view parameters open the all items view with a type filter applied:

- contentbytitle
- portalpagesbytitle
- siteareasbytitle
- componentsbytype
- categoriesbytitle
- authoringtemplates
- presentationtemplates
- workflows
- workflowactions
- workflowstages

The following view parameters open the all items view with a workflow filter applied:

- mydraft
- mypublishpend
- mypublished
- myexpirepend
- myexpired
- alldraftitems
- allpublishpend
- allpublisheditems
- allexpirepend
- allexpireditems

The following view parameters are used to open personal views:
This view parameter opens the all deleted items view:
• alldeleteditems
This view parameter opens the external links view:
• externallinks
The following view parameters open a projects view:
• allprojects
• activeproject
• syndicatingproject
• pendingproject
• publishingproject
• publishedproject
• publishfailedproject
• reviewingproject
This view parameter opens the library explorer:
• explorer
This view parameter opens the home page or launch page:
• launchpage
To open the library explorer at the root view you add root="true" to the tag:
• [plugin:RemoteAction action="openmainview" root="true"]
You can also specify the library to open either by including the library name or id:
• [plugin:RemoteAction action="openmainview" libraryName="name"]
• [plugin:RemoteAction action="openmainview" libraryId="id"]
To open one of item type views you add pfolder="name" to the tag. You must also specify a library name or ID. For example, to open the content view:
• [plugin:RemoteAction action="openmainview" libraryName="name" pfolder="content"]
Accepted values for the pfolder parameter are:
• content
• categories
• authoringtemplates
• presentationtemplates
• workflowitems
To open one of item type sub-views you add psfolder="name" to the tag. For example, to open the workflow view:
• [plugin:RemoteAction action="openmainview" libraryName="name" pfolder="workflowitems" psfolder="workflows"]
Accepted values for the psfolder parameter are:
- workflows
- workflowstages
- workflowactions

You can also open the library explorer at a specific item by adding an itemId parameter and specifying the UUID of the item:

```
[plugin:RemoteAction action="openmainview" itemId="UUID"]
```

**move**

This is used to move a site area or content item.

For example, to open the move dialog for a content item or site area:

```
[plugin:RemoteAction action="move"

docid="com.ibm.workplace.wcm.api.WCM_Content/ID1"]
```

A move direction is specified as "1" for up and "-1" for down. For example, to move a content item up one position:

```
[plugin:RemoteAction action="move"

docid="com.ibm.workplace.wcm.api.WCM_Content/ID1"

pid="com.ibm.workplace.wcm.api.WCM_SiteArea/ID1"

moveDirection="1"]
```

**link**

This will link a content item to a site area.

For example:

```
[plugin:RemoteAction action="link"

docid="com.ibm.workplace.wcm.api.WCM_Content/ID1"

pid="com.ibm.workplace.wcm.api.WCM_SiteArea/ID2"]
```

When linking items you can specify the path to the parent item using the **ppath** parameter instead of the **pid** parameter:

```
[plugin:RemoteAction action="link"

docid="com.ibm.workplace.wcm.api.WCM_Content/ID1"

ppath="library1/sitearea1/sitearea2"]
```

When linking items you can create a new parent item by using the **autoCreateParent** parameter. You must also specify the library where the item being linked is located using the **slibrary** parameter. The **ppath** parameter is used to specify the existing parent that the new parent item is created under:

```
[plugin:RemoteAction action="link"

docid="com.ibm.workplace.wcm.api.WCM_Content/ID1"

autoCreateParent="true" slibrary="libraryname" ppath="library1/

sitearea1/sitearea2"]
```

**copy**

This is used to make a copy of an item.

For example, to copy a content item to a new site area:

```
[plugin:RemoteAction action="copy"

docid="com.ibm.workplace.wcm.api.WCM_Content/ID1"

pid="com.ibm.workplace.wcm.api.WCM_SiteArea/ID2"]
```

You can use the following additional parameters when copying:

- **copyAsDraft="true"**
  This will restart the workflow of the copy being creating. In most cases this would result in the copy being created with a status of draft.

- **wid="com.ibm.workplace.wcm.api.WCM_Workflow/ID1"**
Use this to specify a different workflow to use when creating the copy. This will also restart the workflow of the copy being creating. In most cases this would result in the copy being created with a status of draft.

- **position**="start"
  This will create the copy as the first item under the specified parent item. If not specified the item will be copied as the last child of the specified parent item.

When copying items you can specify the path to the parent item using the **ppath** parameter instead of the **pid** parameter:

- [plugin:RemoteAction action="copy" docid="com.ibm.workplace.wcm.api.WCM_Content/ID1" ppath="library1/sitearea1/sitearea2"]

When copying items you can create a new parent item by using the **autoCreateParent** parameter. You must also specify the library where the item being copied is located using the **slibrary** parameter. The **ppath** parameter is used to specify the existing parent that the new parent item is created under:

- [plugin:RemoteAction action="copy" docid="com.ibm.workplace.wcm.api.WCM_Content/ID1" autoCreateParent="true" slibrary="libraryname" ppath="library1/sitearea1/sitearea2"]

approve

This is used to approve an item in a workflow. You must also specify the **docid** of the item.

For example:

- [plugin:RemoteAction action="approve" docid="com.ibm.workplace.wcm.api.WCM_Content/ID1"]

decline

This is used to decline an item in a workflow. You must also specify the **docid** of the item.

For example:

- [plugin:RemoteAction action="decline" docid="com.ibm.workplace.wcm.api.WCM_Content/ID1"]

saveandapprove

This is used to approve an item in a workflow where that item is currently open in edit mode within the same session. You must also specify the **docid** of the item.

For example:

- [plugin:RemoteAction action="saveandapprove" docid="com.ibm.workplace.wcm.api.WCM_Content/ID1"]
- [plugin:RemoteAction action="saveandapprove" docid="com.ibm.workplace.wcm.api.WCM_Content/ID1" isdraft="true"]

previousstage

This is used to move an item to the previous stage in a workflow. You must also specify the **docid** of the item.

For example:

- [plugin:RemoteAction action="previousstage" docid="com.ibm.workplace.wcm.api.WCM_Content/ID1"]
**viewversions**

This is used to open the versions dialog for an item. You must also specify the `docid` of the item.

For example:
- `[[plugin:RemoteAction action="viewversions" docid="com.ibm.workplace.wcm.api.WCM_Content/ID1"]]

**viewhistory**

This is used to open the history dialog for an item. You must also specify the `docid` of the item.

For example:
- `[[plugin:RemoteAction action="viewhistory" docid="com.ibm.workplace.wcm.api.WCM_Content/ID1"]]

**Populating fields when creating or editing content items and site areas**

When using the "new" or "edit" parameters with content items and site areas, you can also add data to different fields in the item using a URL.

For example, to add "newcontent" as the name of the content item, you would use this URL:
- `[[plugin:RemoteAction action="new" type="com.ibm.workplace.wcm.api.WCM_Content" wcmfield.content.name="newcontent"]]

The following parameters can be used to populate fields when creating or editing content items and site areas:
- `wcmfield.content.name=`
- `wcmfield.content.title=`
- `wcmfield.content.description=`
- `wcmfield.content.authors=`
- `wcmfield.content.owners=`
- `wcmfield.content.publishDate=`
- `wcmfield.content.expiryDate=`
- `wcmfield.content.generalDateOne=`
- `wcmfield.content.generalDateTwo=`
- `wcmfield.content.workflow=` (This can only be used when creating a new content item and site areas.)
- `wcmfield.content.categories=`
- `wcmfield.content.keywords=`
- `wcmfield.element.elementname=`

**Note:** You replace `elementname` with the name of the element you are populating.

The element parameter can only be used with the following element types:
- Text
- Html
- Rich text
- Option Selection
- User Selection
Date and time
Number
JSP
Link
Component Reference

For example, when populating content item fields with user ids you must use this format:

- [plugin:RemoteAction action="new"
  type="com.ibm.workplace.wcm.api.WCM_Content"
  wcmfield.content.authors="uid=usera,cn=cn-name,dc=dc-name"]

For example, when populating content item workflow and category fields you must use the document IDs as their values:

- [plugin:RemoteAction action="new"
  type="com.ibm.workplace.wcm.api.WCM_Content"
  wcmfield.content.workflow="ID1"]
- [plugin:RemoteAction action="new"
  type="com.ibm.workplace.wcm.api.WCM_Content"
  wcmfield.content.categories="ID1"]

For example, when populating content item date fields, the date format must be US English. Either a date and time, or just a date can be specified. If only a date is specified, the time used will be 12:00:00 AM. For example:

- [plugin:RemoteAction action="new"
  type="com.ibm.workplace.wcm.api.WCM_Content"
  wcmfield.content.generalDateOne="Feb 14, 2008 12:53:03 PM"]
- [plugin:RemoteAction action="new"
  type="com.ibm.workplace.wcm.api.WCM_Content"
  wcmfield.content.generalDateOne="Feb 14, 2008"]

The date and time set here are based on the server's timezone, not the timezone of the user's computer.

For example, when populating a content item JSP element, you need to specify the path to the JSP file:

- [plugin:RemoteAction action="new"
  type="com.ibm.workplace.wcm.api.WCM_Content"
  wcmfield.element.jspelementname="/wps/wcm/jsp/html/example.jsp"]

For example, when populating a content item component reference element, you specify the component to reference. For example:

- [plugin:RemoteAction action="new"
  type="com.ibm.workplace.wcm.api.WCM_Content"
  wcmfield.element.mycopref.type="com.ibm.workplace.wcm.api.LibraryNavigatorComponent"
  wcmfield.element.mycopref.id="e4bdf10042d0769698c2b9e25cc973"]

For example, when populating a content item option selection element, you specify each selection option. For example:

- [plugin:RemoteAction action="new"
  type="com.ibm.workplace.wcm.api.WCM_Content"
  wcmfield.element.elementname="AA" wcmfield.element.elementname="BB"]
For example, when populating a content item user selection element, you specify each user. For example:

- `[plugin:RemoteAction action="new"
  type="com.ibm.workplace.wcm.api.WCM_Content"
  wcmfield.element.elementname="uid=wpsadmin,o=defaultWimFileBasedRealm"
  wcmfield.element.elementname="uid=wpsadmin2,o=defaultWimFileBasedRealm"]`

For example, when populating a content item link element, you can specify the following parameters:

**Adding a link to a content item:**

```
[plugin:RemoteAction action="new"
  type="com.ibm.workplace.wcm.api.WCM_Content"
  wcmfield.element.elementname.type="content"
  wcmfield.element.elementname.id="contentID"]
```

**Adding a link to a link component:**

```
[plugin:RemoteAction action="new"
  type="com.ibm.workplace.wcm.api.WCM_Content"
  wcmfield.element.elementname.type="link"
  wcmfield.element.elementname.id="linkcomponentID"]
```

**Adding a link to an image or file resource component:**

```
[plugin:RemoteAction action="new"
  type="com.ibm.workplace.wcm.api.WCM_Content"
  wcmfield.element.elementname.type="resource"
  wcmfield.element.elementname.id="componentID"]
```

**Adding a link to a URL:**

```
[plugin:RemoteAction action="new"
  type="com.ibm.workplace.wcm.api.WCM_Content"
  wcmfield.element.elementname.type="external"
  wcmfield.element.elementname.externalReference="myurl"]
```

To specify whether to use the name of the item you are linking to as the link text, add this to the tag:

```
wcmfield.element.elementname.useReferenceLinkText="true"
```

When specifying an image to display as the link, add this to the tag:

```
wcmfield.element.elementname.linkImage="imagecomponentID"
```

When specifying the text of the link, add this to the URL:

```
wcmfield.element.elementname.linkText="text"
```

When specifying the description of the link, add this to the URL:

```
wcmfield.element.elementname.linkDescription="text"
```

When specifying a link target, add this to the URL:

```
wcmfield.element.elementname.linkTarget=
  _blank
  _parent
  _self
  _top
  targetname
```

**Save parameters**

You can add the following "save" parameters to a remote action tag.
**autoSave**

This is used to save a controllable. This happens in the background and is not displayed to users.

For example:

- `wcmfield.autosave="true"`

**saveValidate**

This parameter determines if warning and error messages resulting from the autosave will be displayed to the user. If set to "true", warning and error messages will be displayed to the user. If set to false, messages are suppressed. The default is true.

For example:

- `wcmfield.saveValidate="false"`

**Adding comments to the item history**

When creating items that use a workflow with "Enter comment on approval" set to true, you can add a comment to the item history by adding `comment="comment text"` to the URL.

For example:

```
[plugin:RemoteAction action="edit" docid="com.ibm.workplace.wcm.api.WCM_Content/ID1" createDraft="true" comment="comment text"]
```

**Examples**

Open the versions view for an item:

- Tag: `[plugin:RemoteAction action="viewversions" docid="com.ibm.workplace.wcm.api.WCM_Content/ID1"]`

Open the history view for an item:

- Tag: `[plugin:RemoteAction action="viewhistory" docid="com.ibm.workplace.wcm.api.WCM_Content/ID1"]`

Open a content item in read mode:

- Tag: `[plugin:RemoteAction action="read" docid="com.ibm.workplace.wcm.api.WCM_Content/ID1"]`

Open a content item in edit mode:

- Tag: `[plugin:RemoteAction action="edit" &docid=com.ibm.workplace.wcm.api.WCM_Content/ID1]`

Move a content item up:
Move a site area down:

- Tag: [plugin:RemoteAction action="move"
docid="com.ibm.workplace.wcm.api.WCM_SiteArea/ID1" moveDirection="-1"
pid="com.ibm.workplace.wcm.api.WCM_SiteArea/ID1"]
  

Create a new content item with title of 'newcontent':

- Tag: [plugin:RemoteAction action="new"
type="com.ibm.workplace.wcm.api.WCM_Content"
wcmfield.content.title="newcontent"]
  

To open a content item in edit mode and automatically change keywords:

- Tag: [plugin:RemoteAction action="edit"
docid="com.ibm.workplace.wcm.api.WCM_Content/ID1"
wcmfield.content.keywords="keyword1" wcmfield.content.keywords="keyword2"]
  

To edit a content item, automatically change the keywords and use autosave to automatically save the content (no dialog opens):

- Tag: [plugin:RemoteAction action="edit"
docid="com.ibm.workplace.wcm.api.WCM_Content/ID1"
wcmfield.content.keywords="keyword1" wcmfield.content.keywords="keyword2" wcmfield.autosave="true"]
  

To edit a content item, automatically save the item and prevent any validation exception from being displayed, use autosave with saveValidate=false:

- Tag: [plugin:RemoteAction action="edit"
docid="com.ibm.workplace.wcm.api.WCM_Content/ID1"
wcmfield.content.keywords="keyword1" wcmfield.autosave="true" wcmfield.saveValidate="false"]
  
To create a content item, set the name and use autosave to automatically save the content (no dialog opens). The authoring template used by the content item must have a workflow pre-selected:

- **Tag**: [plugin:RemoteAction action="new"
  type="com.ibm.workplace.wcm.api.WCM_Content"
  atid="com.ibm.workplace.wcm.api.WCM_ContentTemplate/ID1"
  pid="com.ibm.workplace.wcm.api.WCM_SiteArea/ID2"
  wcmfield.content.name="newcontent" wcmfield.autosave="true"
  wcmfield.saveValidate="true"]

  &type=com.ibm.workplace.wcm.api.WCM_Content
  &atid=com.ibm.workplace.wcm.api.WCM_ContentTemplate/ID1
  &pid=com.ibm.workplace.wcm.api.WCM_SiteArea/ID2
  &wcmfield.content.name=newcontent&wcmfield.autosave=true
  &wcmfield.saveValidate=true

To edit a content item and create a draft on the edit and set the history log comment:

- **Tag**: [plugin:RemoteAction action="edit"
  docid="com.ibm.workplace.wcm.api.WCM_Content/ID1" createDraft="true"
  comment="comment"]

  &docid=com.ibm.workplace.wcm.api.WCM_Content/ID1&createDraft=true
  &comment=comment

Creating a custom launch page

You can configure an authoring portlet to use a launch page of your own design instead of the default user interface.

A custom launch page can either be a JSP or HTML file. You use remote actions to call different views and functions from with the authoring portlet's user interface. You can also use the web content API to add other functions to your launch page. Once you have created a custom launch page, you then configure your authoring portlet to use the custom launch page instead of the default authoring portlet user interface.

Storing JSP Files:

JSP files can be located:

- within the `was_profile_root`/installedApps/cellname/
  PA_WCM_Authoring_UI.ear/llwwcm-authoring.war/jsp/html directory of your server where `cellname` is unique to your installation.

- within any other web application running on portal. When referencing JSP files in another web application, use the following path: `contextPath;jspPath`
  For example: `/wps/customapplication;/jsp/editor.jsp

A custom launch page example

This is a simple example of some code you can add to a JSP or HTML file, to allow users to create and view content items using remote actions.

```jsp
<%--
/* Sample Launch Page */
--%>
```
Creating custom plug-ins

A custom plugin is a reusable Java class that you create to perform a task. You can create custom plug-ins such as custom workflow actions, plug-ins to run when a page is rendered, plug-ins to store multi-locale text strings and plug-ins to run when a file is uploaded.

Creating a rendering plug-in class

A rendering plug-in is a reusable class that you create to perform a task at render time. It can be referenced within web content using a plug-in tag. For example, you could write a plug-in that uses attributes from the current user's profile to determine whether the body of the plug-in tag is rendered or not. A rendering plug-in class requires you to reference a set of web content API methods.

Creating a plug-in class

1. Create a Java class that implements the interface
   com.ibm.workplace.wcm.api.plugin.rendering.RenderingPlugin. This class must implement the following methods:
   * public String getName() .

   Note: This is the name used in the "pluginname" parameter of the plug-in tag. See [Creating a plug-in tag](/WEB-INF/tld/wcm.tld) for further information.
2. Implement `render()` method. This method contains the code that is run when the plug-in is invoked during rendering of a layout that contains a "plug-in" tag that references the custom plug-in. Returning true renders the body markup defined in the plug-in tag. If false is returned, the body of the plug-in tag is skipped. If the plug-in tag has no body markup then the return value is ignored.

3. Methods inherited from `com.ibm.portal.Localized` must also be implemented.

   ```java
   public String getTitle(Locale displayLocale) {}
   ```
   
   This method returns the title for the rendering plugin that will be used to allow selection of the rendering plugin.

   ```java
   public ListModel<Locale> getLocales()
   ```
   
   This method returns a list of locales that are supported by this rendering plugin.

   ```java
   public String getDescription(Locale p_arg0)
   ```
   
   This method returns a description of the rendering plugin.

   See the Javadoc documentation for further information.

For example:

```java
package test;
import java.io.IOException;
import java.io.Writer;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;
import java.util.Locale;
import java.util.Map;
import java.util.Set;
import com.ibm.portal.ListModel;
import com.ibm.portal.ModelException;
import com.ibm.workplace.wcm.api.plugin.rendering.RenderingPlugin;
import com.ibm.workplace.wcm.api.plugin.rendering.RenderingPluginException;
import com.ibm.workplace.wcm.api.plugin.rendering.RenderingPluginModel;
/**
 * A simple rendering plugin to demonstrate the use of the <code>RenderingPlugin</code> API.
 */
public class SimpleRenderingPlugin implements RenderingPlugin
{
    /**
     * A simple list model holding locales.
     */
    protected static class SimpleLocaleListModel<K> implements ListModel<Locale>
    {
        /** the list of locales of this list model */
        final List<Locale> m_localeList = new ArrayList<Locale>();

        /**
         * Constructs this simple list model holding the given locales.
         * @param locales
         *     the locales of this list model. May be <code>null</code>.
         */
        public SimpleLocaleListModel(final Locale[] plocales)
        {
            if (plocales != null) {
                for (int i = 0; i < plocales.length; i++) {
                    this.m_localeList.add(plocales[i]);
                }
            }
        }
    }
```
for (int i = 0; i < p_locales.length; ++i)
{
    m_localeList.add(p_locales[i]);
}

/**
 * (non-Javadoc)
 *
 * @see com.ibm.portal.ListModel#iterator()
 */
@Override
public Iterator<Locale> iterator() throws ModelException
{
    return m_localeList.iterator();
}

/** a list model that only contains the English language locale */
private static final ListModel<Locale> ENGLISH_ONLY = new SimpleLocaleListModel<Locale>(new Locale[]{Locale.ENGLISH});

/**
 * @see com.ibm.portal.Localized#getDescription(java.util.Locale)
 */
@Override
public String getDescription(final Locale p_locale)
{
    return "This is a simple rendering plugin.";
}

/**
 * @see com.ibm.portal.Localized#getLocales()
 */
@Override
public ListModel<Locale> getLocales()
{
    return ENGLISH_ONLY;
}

/**
 * @see com.ibm.workplace.wcm.api.plugin.rendering#getName()
 */
@Override
public String getName()
{
    return "SimpleRenderingPlugin";
}

/**
 * @see com.ibm.portal.Localized#getTitle(java.util.Locale)
 */
@Override
public String getTitle(final Locale p_locale)
{
    return "SimpleRenderingPlugin";
}
Create a plugin.xml file

A plugin.xml file is needed whether the deployment is done using a WAR or EAR, or using a loose jar. If deploying an application in a WAR or EAR, include the plugin.xml file in the application's "WEB-INF" folder. When using a jar, include the plugin.xml in the root of the jar.
Creating a custom workflow action class

You can create custom workflow action classes to enable you to use custom workflow actions in a workflow.

Creating the custom workflow action class

1. Create a java class that implements the interface
   com.ibm.workplace.wcm.api.custom.CustomWorkflowAction. This class must implement the following methods:
   - public Date getExecuteDate(Document p_document) {} (This specifies when the custom action will be executed)
   - public CustomWorkflowActionResult execute(Document p_document) {} (This method contains the code that will run when the custom action is executed.)

2. Implement execute() method. This method contains the code that will be executed against the supplied Document. This method must return a com.ibm.workplace.wcm.api.custom.CustomWorkflowActionResult object to indicate the result of the custom code through the use of com.ibm.workplace.wcm.api.custom.Directives.
   - A custom workflow action result object is created by first retrieving a reference to the WebContentCustomWorkflowService object, and then calling the method
     webContentCustomWorkflowService.getCustomeWorkflowService().createResult. If the CustomWorkflowActionResult does not indicate a failure, changes to the document will be saved. See the Web Content Manager Javadoc for further
information. The Javadoc HTML files are located under the `was_profile_root\installedApps\nodename\wcm.ear\ilwwcm.war\webinterface\` folder.

- Also see the Web Content Manager Javadoc for further information on valid directives.

3. Create a custom workflow action factory class that implements the interface `com.ibm.workplace.wcm.api.custom.CustomWorkflowActionFactory`.

### Create a plugin.xml file

A plugin.xml file is needed whether the deployment is done using a WAR or EAR, or using a loose jar. If deploying via an application in a WAR or EAR, include the plugin.xml file in the application's "WEB-INF" folder. When using a jar, include the plugin.xml in the root of the jar.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<plugin id="com.ibm.workplace.wcm.sample.customworkflowaction"
   name="Sample Custom Workflow Action Factory"
   version="1.0.0"
   provider-name="IBM">

   <extension point="com.ibm.workplace.wcm.api.CustomWorkflowActionFactory"
      id="SimpleCustomWorkflowActionFactory">
      <provider class="com.ibm.workplace.wcm.sample.customworkflowaction.SimpleCustomWorkflowActionFactory"/>
   </extension>

</plugin>
```

- The ID of each plugin must be unique. You must replace the plugin ID specified in this example, `com.ibm.workplace.wcm.sample.customworkflowaction`, with a different ID for each custom workflow you create.
- Each custom workflow action factory is represented by a single `<extension>` tag.
- The value of the point attribute must be "com.ibm.workplace.wcm.api.CustomWorkflowActionFactory".
- Provide an id value of your choice.
- Specify the provider class for your custom workflow action factory.

**Naming conventions:**

If you create a new plugin application with the same names and IDs as an existing plugin, the new plugin may override the first. When creating plugin applications ensure that the following are unique across your system:

- The plugin ID, plugin name and extension ID of the plugin.xml file.
- The fully qualified class name plus path of all classes within the application.
- The file path of any files within the application.

### Creating a Text Provider class

A text provider is used to provide localized text that can be used within web content item forms. For example, a text provider can be used to localize the field labels or help text within an authoring template so that each user sees the labels or help text in their own language.

To use a text provider, you must create a text provider class and then register the text provider by deploying it on the server.
1. Create a java class that implements the interface
   `com.ibm.workplace.wcm.api.plugin.textprovider.TextProvider`. This class must
   implement the following methods:

   ```java
   public String getProviderName() {}
   This method returns the unique name of the text provider.

   public String getString(String key, Locale displayLocale) {}
   This method returns some translated text, given a key identifying the
   message and a locale.

   public Collection<String> getProviderKeys() {}
   This method returns a list of keys used when accessing the text
   provider. These keys are displayed in the authoring UI when a user is
   configuring the text provider.

   public boolean isShownInAuthoringUI() {}
   This method allows you to prevent your text provider from appearing
   in the authoring UI.

   See the Javadoc documentation for further information.
   ```

2. Methods inherited from `com.ibm.portal.Localized` must also be implemented.

   ```java
   public String getTitle(Locale displayLocale) {}
   This method returns the title for the text provider that will be used to
   allow selection of the text provider.

   public ListModel<Locale> getLocales()
   This method returns a list of locales that are supported by this text
   provider.

   public String getDescription(Locale p_arg0)
   This method returns a description of the text provider.

   See the Javadoc documentation for further information.
   ```

3. A plugin.xml file is needed whether the deployment is done using a WAR or
   EAR, or using a loose jar. If deploying via an application in a WAR or EAR,
   include the plugin.xml file in the application's "WEB-INF" folder. When using a
   jar, include the plugin.xml in the root of the jar.

   ```xml
   <?xml version="1.0" encoding="UTF-8"?>
   <plugin id="com.acme"
     name="Sample Text Provider"
     version="1.0.0"
     provider-name="IBM">
     <extension
       point="com.ibm.workplace.wcm.api.TextProvider"
       id="SampleTextProvider">
       <provider class="com.acme.SampleTextprovider"/>
     </extension>
   </plugin>
   ```

   Each plug-in is represented by a single `<extension>` tag.
   The value of the point attribute must be
   "com.ibm.workplace.wcm.api.TextProvider".
   Provide an id value of your choice.
   Specify the provider class for your plug-in.

   **Naming conventions:**
If you create a new plugin application with the same names and IDs as an existing plugin, the new plugin may override the first. When creating plugin applications ensure that the following are unique across your system:

- The plugin ID, plugin name and extension ID of the plugin.xml file.
- The fully qualified class name plus path of all classes within the application.
- The file path of any files within the application.

### Creating a file upload validation class

A file upload validation plugin is invoked anytime a file is uploaded into Web Content Manager. This includes uploading files into file resource, image and style sheet elements, and images uploaded into rich text or HTML elements. The plugin is called within the "validation" processing used by Web Content Manager when uploading files.

To create a file upload validation plugin, you must create a file upload validation class and then register the file upload validation class by deploying it on the server.

1. Create a java class that implements the interface
   ```java
   com.ibm.workplace.wcm.api.extensions.validation.FileUploadValidationPlugin
   ```
   This class must implement the following methods:

   - **public String getName()**
     - This method returns the unique name of the file upload validation plugin.
   
   - **public boolean validate(InputStream p_inptStream, FileUploadValidationContext p_context)**
     - This method throws the FileUploadValidationException.
     - See the Javadoc documentation for further information.

2. Implement validate() method. This method contains the code that will be executed when the plug-in is invoked when uploading a file. If validated, the file will continue to upload. If not validated then the file upload is stopped.
   
   You can display a message in the user interface by including the following code in validate method:

   ```java
   throw new FileUploadValidationException( your own message );
   ```
   
   For example:

   ```java
   package pers.smp.extension.test.validation;

   import java.io.InputStream;
   import java.util.logging.Logger;
   import com.ibm.workplace.wcm.api.extensions.validation.FileUploadValidationContext;
   import com.ibm.workplace.wcm.api.extensions.validation.FileUploadValidationException;
   import com.ibm.workplace.wcm.api.extensions.validation.FileUploadValidationPlugin;
   import com.ibm.workplace.wcm.services.validation.FileUploadValidationContextImpl;

   public class SMPValidation1 implements FileUploadValidationPlugin
   {
     private final long MAX_SIZE_IMAGES = 512 * 1024;
     private final long MAX_SIZE_FILES = 1024 * 1024;

     private static Logger s_log = Logger.getLogger(SMPValidation1.class.getName());

     public String getName()
     {
       return "SMPValidation1";
     }
   }
   ```
public boolean validate(InputStream p_inptStream, FileUploadValidationContext p_context) throws FileUploadValidationException
{
    s_log.info("File Name : " + p_context.getFileName() );
    s_log.info("File Type : " + p_context.getMimeType() );
    s_log.info("File Size : " + p_context.getFileSize() );
    s_log.info("Document Type : " + p_context.getDocumentType() );
    boolean valid = true;
    String message = null;
    String mimeType = p_context.getMimeType();
    if ( mimeType != null && mimeType.startsWith( "image/" ) )
    {
        if ( ! (mimeType.equalsIgnoreCase( "image/gif") || mimeType.equalsIgnoreCase( "image/jpeg") ) )
        {
            throw new FileUploadValidationException( "Invalid image type: " + mimeType + " will only accept GIF and JPG images" );
        }
        if ( p_context.getFileSize() > MAX_SIZE_IMAGES )
        {
            throw new FileUploadValidationException( "Image is too big 500K is maximum size allowed for images. Size is " + p_context.getFileSize( );
        }
    } else
    {
        if ( p_context.getFileSize() > MAX_SIZE_FILES )
        {
            throw new FileUploadValidationException( "File is too big 1M is maximum size allowed for " + mimeType + ". Size is " + p_context.getFileSize( );
        }
    }
    return valid;
}

3. A plugin.xml file is needed whether the deployment is done using a WAR or
   EAR, or using a loose jar. If deploying via an application in a WAR or EAR,
   include the plugin.xml file in the application's WEB-INF folder. When using a
   jar, include the plugin.xml in the root of the jar.

<<?xml version="1.0" encoding="UTF-8"?>
<plugin id="pers_smp_extension_test"
    name="SMP Test Extensions"
    version="1.0.0"
    provider-name="IBM">
    <extension
        point="com.ibm.workplace.wcm.api.FileUploadValidationPlugin" id="SMPValidation1"
        <provider class="pers.smp.extension.test.validation.SMPValidation1"/>
</extension>
</plugin>

- Each plug-in is represented by a single <extension></extension> tag.
- The value of the point attribute must be
  "com.ibm.workplace.wcm.api.FileUploadValidationPlugin".
- Provide an id value of your choice.
- Specify the provider class for your plug-in.

Naming conventions:

If you create a new plugin application with the same names and IDs as an existing
plugin, the new plugin may override the first. When creating plugin applications
ensure that the following are unique across your system:
The plugin ID, plugin name and extension ID of the plugin.xml file.
The fully qualified class name plus path of all classes within the application.
The file path of any files within the application.

Creating a subscriber class

A subscriber plugin is used to run additional functionality on the subscriber that can be used to determine if the subscriber is ready for syndication when a syndication event is invoked.

To create a subscriber plugin, you must create a subscriber class and then register the subscriber class by deploying it on the server.

1. Create a java class that implements the interface
   com.ibm.workplace.wcm.api.extensions.syndication.SubscriberReady. This class must implement the following methods:

   public ResultDirective onSubscriberReady(SubscriberEvent eventInfo)
   - This method contains the code that will be executed when the syndication run starts.
   - This method is executed on the subscriber.
   - The extensions are executed only when there are changes in the syndicator.
   - The extension is not executed every time automatic syndication queues the syndicator.

   See the javadoc for further information.

2. Implement the onSubscriberReady method. This method must return a
   com.ibm.workplace.wcm.api.extensions.syndication.ResultDirective object to indicate whether the syndication engine can continue or stop the syndication process.

3. A plugin.xml file is needed whether the deployment is done using a WAR or EAR, or using a loose jar. If deploying via an application in a WAR or EAR, include the plugin.xml file in the application's "WEB-INF" folder. When using a jar, include the plugin.xml in the root of the jar.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<plugin id="com.ibm.workplace.wcm.sample.subscriberready"
   name="Sample Subscriber Ready Extension"
   version="1.0.0"
   provider-name="IBM">
  <extension
    point="com.ibm.workplace.wcm.api.SubscriberReady"
    id="SubscriberReadyExtension"/>
  <provider class="com.ibm.workplace.wcm.sample.subscriberready.SubscriberReadyExtension"/>
</extension>
</plugin>
```

- The ID of each plugin must be unique.
- You must replace the plugin ID specified in this example, com.ibm.workplace.wcm.sample.subscriberready, with a different ID for each SubscriberReady extension you create.
- Each SubscriberReady extension is represented by a single <extension></extension> tag.
- The value of the point attribute must be com.ibm.workplace.wcm.api.SubscriberReady.
- Provide an id value of your choice.
- Specify the provider class for your SubscriberReady extension.
Naming conventions:

If you create a new plugin application with the same names and IDs as an existing plugin, the new plugin may override the first. When creating plugin applications ensure that the following are unique across your system:

- The plugin ID, plugin name and extension ID of the plugin.xml file.
- The fully qualified class name plus path of all classes within the application.
- The file path of any files within the application.

Creating a syndicator class

A syndicator plugin is used to run additional functionality on the syndicator when a syndication event is invoked.

To create a syndicator plugin, you must create a syndicator class and then register the syndicator class by deploying it on the server.

1. Create a java class that implements the interface
   com.ibm.workplace.wcm.api.extensions.syndication.SyndicatorStarted. This class must implement the following methods:

   public ResultDirective onSyndicatorStarted(SyndicatorEvent eventInfo)
   
   - This method contains the code that will be executed when the syndication run starts.
   - This method is executed after the plug-ins for the SubscriberReady extension point are run.
   - The extensions are executed only when there are changes on the syndicator.
   - The extension is not executed every time automatic syndication queues the syndicator.

   See the Javadoc documentation for further information.

2. Implement the onSyndicatorStarted method.
   
   - This method contains the code that is run on the syndicator when there are changes that are available for syndication to the subscriber.
   - This method must return a
     com.ibm.workplace.wcm.api.extensions.syndication.ResultDirective object to indicate whether the syndication engine can continue or stop the syndication process.

3. A plugin.xml file is needed whether the deployment is done using a WAR or EAR, or using a loose jar. If deploying via an application in a WAR or EAR, include the plugin.xml file in the application's "WEB-INF" folder. When using a jar, include the plugin.xml in the root of the jar.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<plugin id="com.ibm.workplace.wcm.sample.syndicatorstarted"
   name="Sample Syndicator Started Extension"
   version="1.0.0"
   provider-name="IBM">
  <extension
    point="com.ibm.workplace.wcm.api.SyndicatorStarted"
    id=" SyndicatorStartedExtension">
    <provider class="com.ibm.workplace.wcm.sample.syndicatorstarted.SyndicatorStartedExtension"/>
  </extension>
</plugin>
```

- The ID of each plugin must be unique.
• You must replace the plugin ID specified in this example, com.ibm.workplace.wcm.sample.syndicatorstarted, with a different ID for each SyndicatorStarted extension you create.
• Each SyndicatorStarted extension is represented by a single <extension></extension> tag.
• The value of the point attribute must be com.ibm.workplace.wcm.api.SyndicatorStarted.
• Provide an id value of your choice.
• Specify the provider class for your SyndicatorStarted extension.

Naming conventions:
If you create a new plugin application with the same names and IDs as an existing plugin, the new plugin may override the first. When creating plugin applications ensure that the following are unique across your system:
• The plugin ID, plugin name and extension ID of the plugin.xml file.
• The fully qualified class name plus path of all classes within the application.
• The file path of any files within the application.

Creating a context processor class
When configured, a context processor plug-in is invoked by the web content viewer portlet before rendering and allows the current context, such as the item to display, to be modified.

To create a context processor plug-in, you must create a context processor class and then register the context processor class by deploying it on the server and selecting it from within a Web content viewer portlet.
1. Create a Java class that implements the interface com.ibm.workplace.wcm.api.ContextProcessor. This class must implement the following method:

```java
/**
 * Processes the supplied <i>ContextProcessorParams</i> and updates parameters within
 * as necessary
 *
 * @param p_currentSession The current Http Session
 * @param p_contextProcessorParams The editable <i>ContextProcessorParams</i> object
 */
public void process(HttpSession p_currentSession, ContextProcessorParams p_contextProcessorParams);
```
See the Javadoc for further information.
2. Implement the process method. This method contains the code that is executed when the plug-in is invoked and allows you to modify the current context using the ContextProcessorParams object before the current context is rendered.
3. A plugin.xml file is needed whether the deployment is done using a WAR or EAR, or using a loose jar. If deployed using an application in a WAR or EAR, include the plugin.xml file in the application “WEB-INF” folder. When using a jar, include the plugin.xml in the root of the jar.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<plugin id="SampleContextProcessorPluginId" name="SampleContextProcessor" provider-name="IBM" version="1.0.0">
  <extension point="com.ibm.workplace.wcm.api.ContextProcessor" id="SampleContextProcessorPlugin">
    <processor class="com.acme.SampleContextProcessor"/>
  </extension>
</plugin>
```
4. Edit the settings of the web content viewer portlet you want to associate with your context processor:
   a. Go to the "configuration" or "edit shared settings" view of a web content viewer portlet.
   b. Go to Advanced Options > Plugins
   c. Select a context processor plug-in.

   • Each plug-in is represented by a single <extension/></extension> tag.
   • The value of the point attribute must be "com.ibm.workplace.wcm.api.ContextProcessor".
   • Provide an ID value of your choice.
   • Specify the provider class for your plug-in.

Naming conventions:

If you create a plug-in application with the same names and IDs as an existing plug-in, the new plug-in may override the first. When creating plug-in applications ensure that the following are unique across your system:

   • The plug-in ID, plug-in name, and extension ID of the plugin.xml file.
   • The fully qualified class name plus path of all classes within the application.
   • The file path of any files within the application.

Creating a content page resolution filter class

A content page resolution filter is used to customize the behavior of the content page resolution filter chain. This can enable you to tailor the response to a web content request in several ways, including overriding the content item displayed or the portal page used to display a content item in the web content viewer.

The content page resolution filter chain is composed of filters that are based on the Intercepting Filter design pattern, which provides a mechanism for intercepting a request and manipulating the request and its response. When used with requests for web content, the content page resolution filter chain has default filters that process any URL parameters contained in the web content request and then determine which portal page has a matching web content association. The default filters occur at the end of the filter chain.

You can customize the content page resolution filter chain by creating custom filters that are registered with the portal through the Eclipse plug-in framework with the extension ID com.ibm.workplace.wcm.api.ContentPageResolutionFilter. The sequence of filters in the filter chain is specified by a weight value associated with each filter. To insert custom filters into the filter chain before the default filters, you can use the weight attribute in the plugin.xml file. If the weight attribute is not present, filter sequence is determined by the getFilterChainWeight method of each custom filter.

Custom filters can perform various actions:

   • Modify parameters before calling the default filters.
   • Modify the result of the default filters.
   • Handle exceptions generated by the default filters.
   • Determine whether the default filters should be invoked at all.
   • Modify the content path that is used as input for the default filters.
   • Explicitly set a target page for displaying content.
• Determine which web content page should be used, if the default filters find more than one matching web content page for the request.
• Modify the presentation template selection.
• Set HTTP response status codes.
• Send redirects to external web resources.

To use a content page resolution filter, you must create a content page resolution filter class and then register the filter by deploying it on the server.

1. Create a java class that implements the interface
   com.ibm.workplace.wcm.api.extensions.resolution.ContentPageResolutionFilter. This class must implement the following methods:

   public int getFilterChainWeight() {}
   This method returns the weight applied to the content page resolution filter elements in the filter chain. The lower the weight, the earlier the filter is inserted into the chain. If the weight parameter is defined in the plugin.xml file, that value overrides the value returned by this method.

   public void resolve(ContentPageResolutionRequest request,
                        ContentPageResolutionResponse response,
                        ContentPageResolutionFilterChain chain) {}
   This method is invoked during ContentPageResolution processing. The response parameter enables you to modify the content item displayed, the portal page where the content is displayed, and the presentation template used to render the content item. The request extends the resolver interface with an additional method that gets the content item that has been addressed. The filter chain contains the subsequent filters that can be invoked if needed.

   See the Javadoc documentation for further information.

2. A plugin.xml file is needed whether the deployment is done using a WAR or EAR, or using a loose jar. If deploying with an application in a WAR or EAR, include the plugin.xml file in the application's "WEB-INF" folder. When using a jar, include the plugin.xml in the root of the jar.

   <plugin id="com.example" name="Sample Content Page Resolution Filter" version="1.0.0" provider-name="IBM">
     <extension point="com.ibm.workplace.wcm.api.ContentPageResolutionFilter" id="SampleContentPageResolutionFilter">
       <provider class="com.example.SampleContentPageResolutionFilter" weight="1"/>
     </extension>
   </plugin>

When creating plug-ins, note the following:
• Each plug-in is represented by a single <extension> tag.
• The value of the extension point attribute must be com.ibm.workplace.wcm.api.ContentPageResolutionFilter.
• Provide an ID value of your choice.
• Specify the filter class for your plug-in.
• The weight parameter overrides the value of the getFilterChainWeight method.
**Naming conventions:**

If you create a new plug-in application with the same names and IDs as an existing plug-in, the new plug-in may override the first. When creating plug-in applications ensure that the following are unique across your system:

- The plug-in ID, plug-in name and extension ID of the plugin.xml file.
- The fully qualified class name and path of all classes within the application.
- The file path of any files within the application.

**Examples**

- The `LocaleDependantSelectionFilter` example performs the default page resolution and selects a target page from the candidate pages, according to the user's locale.

```java
package com.ibm.workplace.wcm.extension.resolution;

import java.util.List;
import java.util.Locale;
import javax.naming.InitialContext;
import javax.naming.NamingException;
import com.ibm.portal.ObjectID;
import com.ibm.portal.model.CorLocalizedContextHome;
import com.ibm.portal.model.LocalizedContext;
import com.ibm.workplace.wcm.api.extensions.resolution.ContentPageResolutionFilter;
import com.ibm.workplace.wcm.api.extensions.resolution.ContentPageResolutionFilterChain;
import com.ibm.workplace.wcm.api.extensions.resolution.ContentPageResolutionRequest;
import com.ibm.workplace.wcm.api.extensions.resolution.ContentPageResolutionResponse;
import com.ibm.workplace.wcm.api.extensions.resolution.exceptions.ContentPageResolutionException;

public class LocaleDependantSelectionFilter implements ContentPageResolutionFilter {
    CorLocalizedContextHome localizedContextHome;

    public LocaleDependantSelectionFilter() {
        try {
            InitialContext ctx = new InitialContext();
            localizedContextHome = (CorLocalizedContextHome) ctx.lookup(CorLocalizedContextHome.JNDI_NAME);
        } catch (NamingException e) {
            e.printStackTrace();
        }
    }

    public int getFilterChainWeight() {
        return 2;
    }

    public void resolve(ContentPageResolutionRequest request, ContentPageResolutionResponse response,
            ContentPageResolutionFilterChain chain) throws ContentPageResolutionException {
        // do standard resolution first
        chain.resolve(request, response);

        // check if more than one candidate pages
        List<ObjectID> candidates = response.getCandidatePageIds();
        if (candidates.size() > 1) {
            // ...}
```
The `GetLocalizedContextFromRequest` method returns the `LocalizedContext` object associated with the request.

```java
LocalizedContext context = localizedContextHome.getLocalizedContext(request.getContext());
Locale locale = context.getPreferredSupportedLocale();
String lang = locale.getLanguage();
// find page with matching unique name
for (ObjectID pageId : candidates)
{
    if (pageId.getUniqueName().endsWith(".*" + lang))
    {
        response.setPageID(pageId);
        break;
    }
}
}
```

- The `ChangeContentPathFilter` example changes the content path and performs the default resolution on the new content path.

```java
public class ChangeContentPathFilter implements ContentPageResolutionFilter
{
    public int getFilterChainWeight()
    {
        return 3;
    }

    public void resolve(ContentPageResolutionRequest request, ContentPageResolutionResponse response,
                        ContentPageResolutionFilterChain chain)
    throws ContentPageResolutionException
    {
        // instead of england render UK
        if ("/countries/world/europe/england".equals(response.getContentPath()))
        {
            response.setContentPath("/countries/world/europe/uk");
        }
        chain.resolve(request, response);
    }
}
```

- The `ResolveToSpecificPageFilter` example resolves to a specific page. No default resolution is performed in this case.

```java
public class ResolveToSpecificPageFilter implements ContentPageResolutionFilter
{
    private Identification identification;

    public ResolveToSpecificPageFilter()
    {
        // init
    }
}
```
try {
    InitialContext ctx = new InitialContext();
    identification = (Identification) ctx.lookup("portal:service/Identification");
} catch (NamingException nx) {
    nx.printStackTrace();
}

public int getFilterChainWeight() {
    return 4;
}

public void resolve(ContentPageResolutionRequest request, ContentPageResolutionResponse response, ContentPageResolutionFilterChain chain) throws ContentPageResolutionException {
    try {
        // always resolve to page with unique name my.default.page
        ObjectID pageId = identification.deserialize("my.default.page");
        response.setPageID(pageId);
        response.setCandidatePageIds(Arrays.asList(new ObjectID[]{pageId}));
    } catch (SerializationException e) {
        throw new ContentPageResolutionException(e);
    }
}

• The SetResponseCodePageResolutionFilter example checks for the requested web content item. If it does not exist, the code throws an exception that results in sending a 404 (Not found) HTTP response status code. If the content exists, resolution is delegated to other filters in the chain.

package com.ibm.workplace.wcm.extension.resolution;

import java.util.Arrays;
import java.util.HashSet;
import java.util.Iterator;
import java.util.Locale;
import java.util.Set;
import javax.servlet.http.HttpServletResponse;
import com.ibm.portal.ListModel;
import com.ibm.portal.LocalizedStatus;
import com.ibm.portal.ModelException;
import com.ibm.workplace.wcm.api.extensions.resolution.ContentPageResolutionFilter;
import com.ibm.workplace.wcm.api.extensions.resolution.ContentPageResolutionFilterChain;
import com.ibm.workplace.wcm.api.extensions.resolution.ContentPageResolutionRequest;
import com.ibm.workplace.wcm.api.extensions.resolution.ContentPageResolutionResponse;
import com.ibm.workplace.wcm.api.extensions.resolution.ResolvedItem;
import com.ibm.workplace.wcm.api.extensions.resolution.exceptions.ContentPageResolutionException;

public class SetResponseCodePageResolutionFilter implements ContentPageResolutionFilter {
    public void resolve(ContentPageResolutionRequest request, ContentPageResolutionResponse response, ContentPageResolutionFilterChain chain) throws ContentPageResolutionException {
        if (!itemExists(request.getItem())) {
            // send 404
            throw new ContentPageResolutionException(new ContentNotFoundException());
        }
    }
}
else
{
    // forward to the chain if the web content exists
    chain.resolve(request, response);
}

private boolean itemExists(ResolvedItem item)
{
    return (item.getItemID() != null) && (item.getItemPath() != null);
}

public int getFilterChainWeight()
{
    return 1;
}

private static class ContentNotFoundException extends Exception implements LocalizedStatus
{
    private static final long serialVersionUID = 70L;
    private static final Set<Locale> SUPPORTED_LOCALES = new HashSet<Locale>(Arrays.asList(new Locale[]{ Locale.ENGLISH }));
    private static final String MESSAGE = "The requested web content does not exist";

    public ContentNotFoundException()
    {
        super(MESSAGE);
    }

    public int getStatus()
    {
        return HttpServletResponse.SC_NOT_FOUND;
    }

    public String getTitle(Locale locale)
    {
        return MESSAGE;
    }

    public String getDescription(Locale locale)
    {
        return MESSAGE;
    }

    public ListModel<Locale> getLocales()
    {
        return new ListModel<Locale>()
        {
            public Iterator<Locale> iterator() throws ModelException
            {
                return SUPPORTED_LOCALES.iterator();
            }
        };
    }
}

• The SendRedirectPageResolutionFilter example checks for the requested web content item. If the content does not exist, the code sends a redirect to http://www.ibm.com. If the content exists, the resolution is delegated to other filters in the chain.

package com.ibm.workplace.wcm.extension.resolution;

import java.io.UnsupportedEncodingException;
import java.net.URI;
import java.net.URISyntaxException;
import java.net.URLDecoder;
import com.ibm.portal.resolver.exceptions.ResolutionException;
import com.ibm.portal.resolver.helper.CORResolutionService;
import com.ibm.portals.state.exceptionsStateException;
import com.ibm.workplace.wcm.api.extensions.resolution.ContentPageResolutionFilter;
```java
import com.ibm.workplace.wcm.api.extensions.resolution.ContentPageResolutionFilterChain;
import com.ibm.workplace.wcm.api.extensions.resolution.ContentPageResolutionRequest;
import com.ibm.workplace.wcm.api.extensions.resolution.ContentPageResolutionResponse;
import com.ibm.workplace.wcm.api.extensions.resolution.ResolvedItem;
import com.ibm.workplace.wcm.api.extensions.resolution.exceptions.ContentPageResolutionException;

public class SendRedirectPageResolutionFilter implements ContentPageResolutionFilter
{
    // the URL to redirect to
    private static final String URL = "http://www.ibm.com";

    public void resolve(ContentPageResolutionRequest request, ContentPageResolutionResponse response, ContentPageResolutionFilterChain chain) throws ContentPageResolutionException
    {
        if (!itemExists(request.getItem()))
        {
            try
            {
                // encode to URL, this is important to prevent that the '://'
                // character appears in the path
                String encodedURL = URLEncoder.encode(URL, "UTF-8");
                final URI redirectURI = new URI(com.ibm.portal.resolver.Constants.SCHEME_REDIRECT, encodedURL, null);
                CORResolutionService.SINGLETON.resolve(request.getResolved(), redirectURI, request.getVerb(), request.getResolutionParameters(), request.getAcceptValues());
            }
            catch (UnsupportedEncodingException uenc)
            {
                // should never happens as long as UTF-8 is supported
                throw new ContentPageResolutionException(uenc);
            }
            catch (URISyntaxException e)
            {
                throw new ContentPageResolutionException(e);
            }
            catch (StateException e)
            {
                throw new ContentPageResolutionException(e);
            }
            catch (ResolutionException e)
            {
                // do not catch the resolution exception
                // as this is used internally to trigger the redirect
                throw new ContentPageResolutionException(e);
            }
        } else
        {
            chain.resolve(request, response);
        }
    }

    private boolean itemExists(ResolvedItem item)
    {
        return (item.getItemId() != null) && (item.getItemPath() != null);
    }

    public int getFilterChainWeight()
    {
        return 1;
    }
}
```

The following example provides the plugin.xml file that registers the previous sample filters.

Registering all of these filters in one system is not recommended. The filters perform overlapping operations and are also exclusive in some cases. To use one of the filters in your system, remove the other unused filters in the plugin.xml file before deploying the file.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<?eclipse version="3.0"?>
<plugin
    id="com.ibm.workplace.wcm.extension.resolution"
    name="Content Page Resolution Filter"
```
Creating a content URL generation filter class

A content URL generation filter is used to customize the URLs that are generated by a web content viewer. By creating a plug-in that implements a content URL generation filter, you can tailor the URLs to content items.

The web content viewer generates a content URL whenever there is a URL to web content within content that the viewer is displaying.

The content URL generation filter chain is a chain of filters that are based on the Intercepting Filter design pattern. This design pattern provides a mechanism for intercepting a request and manipulating the request and its response. When used with requests for content URL generation, the content URL generation filter chain has default filters that perform the following actions:

- Process any request to generate a URL that target a web content item.
- Create a URL based on the configuration of the web content viewer and any parameters set on the URL generation tags, such as the URLCmpnt tag.

The default filters occur at the end of the filter chain.

You can customize the content URL generation filter chain by creating custom filters. These filters are registered with the portal through the Eclipse plug-in framework with the extension ID com.ibm.workplace.wcm.api.ContentUrlGenerationFilter. The sequence of filters in the filter chain is specified by a weight value associated with each filter plug-in.
To insert custom filters into the filter chain before the default filters, you can use the weight attribute in the plugin.xml file. If the weight attribute is not present, filter sequence is determined by the getFilterChainWeight method of each custom filter factory.

Custom content URL generation filters can perform various actions:
- Modify parameters before calling the default URL generation filters.
- Modify the URL that is generated by the default filters.
- Handle exceptions that are generated by the default filters.
- Determine whether the default filters are invoked.
- Modify the content path that is used as input for the default URL generation filters.
- Generate any type of URL without using the default URL generation filters.

To use a content URL generation filter, you must create two classes:
- A content URL generation filter that is used to create URLs.
- A content URL generation filter factory that is used to create new instances of the filter.

You must deploy both classes on the server and register the filter factory using a plugin.xml file.

1. Create a Java class that implements the interface com.ibm.workplace.wcm.api.extensions.url.ContentUrlGenerationFilterFactory. This class must implement the following methods:

   public ContentUrlGenerationFilter getFilter(RenderRequest portletRequest, RenderResponse portletResponse) throws ContentUrlFilterInstantiationException
   
   This method is called by the content URL generation chain once for each content item that is rendered. The method creates an instance of a content URL generation filter, which generates all URLs that appear in the rendered web content.

   public int getFilterChainWeight() {}
   
   This method returns the weight applied to the content URL generation filter elements in the filter chain. The lower the weight, the earlier the filter is inserted into the chain. If the weight parameter is defined in the plugin.xml file, that value overrides the value returned by this method.

   For more information about this class, see the Javadoc documentation.

2. Create a Java class that implements the interface com.ibm.workplace.wcm.api.extensions.url.ContentUrlGenerationFilter. This class must implement the following methods:

   public void writeURL(ContentUrlGenerationRequest request, ContentUrlGenerationResponse response, ContentUrlGenerationFilterChain chain) throws ContentUrlGenerationException, IOException
   
   This method is invoked during content URL generation processing and is invoked once per content URL. The response parameter enables you to write to the URL. The request parameter contains the following information:
   - Information about the item for which the URL is generated.
   - Any web content viewer configuration and related information that might affect the generation of the URL.
The filter chain contains the subsequent filters that can be invoked when needed.

**public void dispose()**

This method is invoked by the filter chain after all URLs that appear in the rendered content items are written. The method enables filters to perform a cleanup of their internal state.

For more information about this class, see the Javadoc documentation.

3. A plugin.xml file is needed whether the deployment is done using a WAR or EAR file, or by using a JAR file. If you deploy with an application in a WAR or EAR file, include the plugin.xml file in the "WEB-INF" folder. When using a JAR file, include the plugin.xml in the root of the JAR file.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<plugin id="com.example.content.url"
  name="Sample content URL generation filter"
  version="1.0.0"
  provider-name="IBM">
  <extension point="com.ibm.workplace.wcm.api.ContentUrlGenerationFilter"
    id="SampleContentUrlGenerationFilter">
    <factory class="com.example.SampleContentUrlGenerationFilterFactory"
      weight="1"/>
  </extension>
</plugin>
```

When creating plug-ins, note the following characteristics:

- Each plug-in is represented by a single <extension> tag.
- The value of the extension point attribute must be com.ibm.workplace.wcm.api.ContentUrlGenerationFilter.
- Provide an ID value of your choice.
- Specify the filter factory class for your plug-in.
- The weight parameter overrides the value of the getFilterChainWeight method

**Naming conventions:**

If you create a new plug-in application with the same names and IDs as an existing plug-in, the new plug-in might override the first. When creating plug-in applications ensure that the following elements are unique across your system:

- The plug-in ID, plug-in name, and extension ID of the plugin.xml file.
- The fully qualified class name and path of all classes within the application.
- The file path of any files within the application.

**Related concepts:**

“Friendly URLs and web content viewers” on page 551

Friendly URLs provide a way for you to define a custom address for a portal page that is easy to remember and share. The web content viewer expands on friendly URL support by enabling you to specify additional path information in the friendly URL.

**Example 1: Append a prefix to a content URL**

This example demonstrates a content URL generation filter that appends a prefix to each content URL that is written. This type of content URL generation filter is useful when used with an HTTP server that dynamically rewrites incoming URLs.
Factory class
/******************************************************************
* Copyright IBM Corp. 2011
*
******************************************************************/
package com.ibm.workplace.wcm.api.samples;
import javax.portlet.*;
import com.ibm.workplace.wcm.api.extensions.url.*;
public class RewriteUrlGenerationFilterFactory implements ContentUrlGenerationFilterFactory {
@Override
public ContentUrlGenerationFilter getFilter(RenderRequest portletRequest, RenderResponse portletResponse)
throws ContentUrlFilterInstantiationException {
return new RewriteUrlGenerationFilter();
}
@Override
public int getFilterChainWeight() {
return 5;
}
}

Filter class
**************************************
* Copyright IBM Corp. 2011
*
******************************************************************/
package com.ibm.workplace.wcm.api.samples;
import java.io.*;
import com.ibm.workplace.wcm.api.extensions.url.*;
public class RewriteUrlGenerationFilter implements ContentUrlGenerationFilter {
/** that static prefix that is prepended to all URLs */
private static final String PREFIX = "/content/";
@Override
public void dispose() {
// no cleanup required for this filter
}
@Override
public void writeURL(ContentUrlGenerationRequest request, ContentUrlGenerationResponse response,
ContentUrlGenerationFilterChain chain) throws ContentUrlGenerationException,
IOException {
final String contentPath = request.getContentPath(true);
if (contentPath != null && !contentPath.isEmpty()) {
// write the prefix
final Writer out = response.getWriter();
out.write(PREFIX);
// write path to content
if (contentPath.charAt(0) == ’/’) {
// Omit a leading / to avoid 2 / characters
out.write(contentPath, 1, contentPath.length() - 1);
} else {
out.write(contentPath);
}
} else {
// let the other filters handle prefix URLs
chain.writeURL(request, response);
}
}
}

662

Web Content Manager Version 8.0


Example 2: Generate a friendly URL for web content
This example demonstrates a content URL generation filter that generates a friendly URL for web content.

The filter determines the target portal page that is used to render the web content. If there is a web content page that has a content association that references a parent site area of the content, the filter writes out a friendly URL to the content.

Friendly URLs for web content are constructed according to several factors. For example, the friendly URL /wps/myportal/home/news/news+1 is determined by the following details:

- Content path: /Library1/News/News 1
- Content association on the web content page where the content is rendered:
  /Library1/News site area
- Friendly URL of this page: /home/news

To use the sample filter that is described here, each page for which a content URL is generated must have these characteristics:

- The page must be a web content page with a friendly name.
- The page must have a default content association that references the parent site area of the content item.

The friendly URLs that are generated by this sample filter do not contain any portlet state information. When a user clicks one of these friendly URLs from within a web content viewer that does maintain state information, the state information in the viewer is cleared.

Important: These sample filter classes are not provided by default with IBM® Web Content Manager. Example code is provided here that you can take and use to implement the classes yourself.

Related concepts:

“Friendly URLs and web content viewers” on page 551
Friendly URLs provide a way for you to define a custom address for a portal page that is easy to remember and share. The web content viewer expands on friendly URL support by enabling you to specify additional path information in the friendly URL.

Example 2: Filter factory class:
This sample demonstrates a filter factory class. This filter class creates an instance of the filter for each request to render web content that contains URLs to other web content items.
The factory also obtains references to portal services and models that are required by the filter. These references are required to look up web content pages according to these criteria:

- The path of the web content for which the URL is generated.
- The configuration of the web content viewer.

By default, this sample filter is enabled after you deploy the sample classes. However, you can disable the sample filter by setting the following portlet preference in the web content viewer configuration:

- Preference: DISABLE_FriendlyUrlGenerationFilter
- Value: true

```java
package com.ibm.workplace.wcm.api.samples;
import java.util.logging.*;
import javax.naming.*;
import javax.portlet.*;
import com.ibm.portal.*;
import com.ibm.portal.identification.*;
import com.ibm.portal.portlet.service.*;
import com.ibm.portal.portlet.service.model.*;
import com.ibm.portal.resolver.friendly.service.*;
import com.ibm.portal.services.contentmapping.*;
import com.ibm.portal.state.exceptions.*;
import com.ibm.portal.state.service.*;
import com.ibm.portal.state.exceptions.*;
import com.ibm.portal.state.service.*;
import com.ibm.workplace.wcm.api.*;
import com.ibm.workplace.wcm.api.exceptions.*;
import com.ibm.workplace.wcm.api.extensions.url.*;

/**
 * Factory for the friendly content URL generation filter.
 *
 * The filter factory is used to create a new instance of the filter for each
 * request to render a web content that contains URLs to other web content
 * items. This sample filter can be disabled by setting a portlet preference on
 * the Web Content Viewer. To disable the filter set a preference with the name
 * {@value #DISABLE_FILTER} and the value "true". By default the filter will be
 * enabled after the deployment of the sample classes.
 *
 * Furthermore the factory is used to obtain references to different portal
 * services and models that are required by the filter to do the lookup of web
 * content pages based on the path of the web content the URL is generated for
 * and the configuration of the Web Content Viewer portlet.
 */
public class FriendlyUrlGenerationFilterFactory implements ContentUrlGenerationFilterFactory {

    /**
     * name of the portlet preference that can be used to check if the filter is
     * disabled
     */
    public final static String DISABLE_FILTER = "DISABLE_FriendlyUrlGenerationFilter";

    /** logger */
    private static final Logger LOGGER = Logger.getLogger(FriendlyUrlGenerationFilterFactory.class.getName());

    /** lock object for Identification lookup */
    private static final Object LOCK_IDENTIFICATION = new Object();

    /** the identification service */
    private static volatile Identification IDENTIFICATION;

    /** lock object for MappingURLTreeModelProvider lookup */
```
private static final Object LOCK_MAPPING_URL_MODEL_PROVIDER = new Object();

/** provider for url mapping model */
private static volatile MappingURLTreeModelProvider MAPPING_URL_MODEL_PROVIDER;

/** lock object for StateManagerHome lookup */
private static final Object LOCK_STATE_MGR_SRV = new Object();

/** state manager service */
private static volatile PortletStateManagerService STATE_MGR_SRV;

/** lock object for friendly selection service lookup */
private static final Object LOCK_FRIENDLY_SEL_SRV = new Object();

/** friendly selection service */
private static volatile PortletFriendlySelectionServiceHome FRIENDLY_SEL_SRV;

/** lock object for friendly selection service lookup */
private static final Object LOCK_CONTENT_MAPPING_HOME = new Object();

/** content mapping info home */
private static volatile ContentMappingInfoHome CONTENT_MAPPING_HOME;

/** lock object for web content service lookup */
private static final Object LOCK_WEB_CONTENT_SRV = new Object();

/** web content service */
private static volatile WebContentService WEB_CONTENT_SRV;

/** lock object for content model provider lookup */
private static final Object LOCK_CONTENT_MODEL_PROVIDER = new Object();

/** content model provider */
private static volatile ContentModelProvider CONTENT_MODEL_PROVIDER;

@Override
public ContentUrlGenerationFilter getFilter(final RenderRequest request, final RenderResponse response) throws ContentUrlFilterInstantiationException {
    final boolean isLogging = LOGGER.isLoggable(Level.FINEST);
    if (isLogging) {
        LOGGER.entering(getClass().getName(), "getFilter");
    }
    ContentUrlGenerationFilter result = null;
    // check if the filter is enabled. This filter can be disabled by
    // setting portlet preference to a value of 'true'
    if (!Boolean.valueOf(request.getPreferences().getValue(DISABLE_FILTER, Boolean.FALSE.toString()))) {
        try {
            Workspace workspace = getWebContentService().getRepository().getWorkspace();
            // user access is sufficient to lookup a item
            workspace.useUserAccess(true);
            result = new FriendlyUrlGenerationFilter(getFriendlySelectionServiceHome().
                      getPortletFriendlySelectionService(request, response),
                      getContentModelProvider().
                      getContentModel(request, response),
                      workspace, getIdentification(),
                      getStateManageService().
                      getPortletStateManager(request, response),
                      getMappingURLTreeModelProvider().
                      getMappingURLTreeModel(request, response),
                      getContentMappingInfoHome());
        } catch (UnknownAccessorTypeException e) {
            throw new ContentUrlFilterInstantiationException(e);
        } catch (CannotInstantiateAccessorException e) {
            throw new ContentUrlFilterInstantiationException(e);
        } catch (PortletServiceUnavailableException e) {
            throw new ContentUrlFilterInstantiationException(e);
        } catch (ServiceNotAvailableException e) {
            throw new ContentUrlFilterInstantiationException(e);
        } catch (OperationFailedException e) {
            throw new ContentUrlFilterInstantiationException(e);
        }
    }
    return result;
}
throw new ContentUrlFilterInstantiationException(e); } catch (StateManagerException e) {
    throw new ContentUrlFilterInstantiationException(e); } catch (ModelException e) {
    throw new ContentUrlFilterInstantiationException(e); } catch (StateException e) {
    throw new ContentUrlFilterInstantiationException(e); } catch (NamingException e) {
    throw new ContentUrlFilterInstantiationException(e); }

if (isLogging) {
    LOGGER.exiting(getClass().getName(), "getFilter", result);
} return result;

@Override
public int getFilterChainWeight() {
    return 4;
}

/**
 * Get the identification service
 *
 * @return a reference to the identification service
 * @throws NamingException
 * if creation of {link InitialContext} or context lookup fails
 */
private static Identification getIdentification() throws NamingException {
    if (IDENTIFICATION == null) {
        synchronized (LOCK_IDENTIFICATION) {
            if (IDENTIFICATION == null) {
                final Context ctx = new InitialContext();
                IDENTIFICATION = (Identification) ctx.lookup(Identification.JNDI_NAME);
            }
        }
    }
    return IDENTIFICATION;
}

/**
 * Get the provider for URL mapping model
 *
 * @return A reference to the provider for the URL mapping model
 * @throws NamingException
 * @throws PortletServiceUnavailableException
 * if creation of {link InitialContext} or context lookup fails
 * if the service is not available
 */
private static MappingURLTreeModelProvider getMappingURLTreeModelProvider() throws NamingException, PortletServiceUnavailableException {
    if (MAPPING_URL_MODEL_PROVIDER == null) {
        synchronized (LOCK_MAPPING_URL_MODEL_PROVIDER) {
            if (MAPPING_URL_MODEL_PROVIDER == null) {
                final Context ctx = new InitialContext();
                final PortletServiceHome psh = (PortletServiceHome) ctx
                        .lookup(MappingURLTreeModelProvider.JNDI_NAME);
                MAPPING_URL_MODEL_PROVIDER = psh.getPortletService(MappingURLTreeModelProvider.class);
            }
        }
    }
    return MAPPING_URL_MODEL_PROVIDER;
}

/**
private static PortletStateManagerService getStateManageService() throws NamingException, PortletServiceUnavailableException {
    if (STATE_MGR_SRV == null) {
        synchronized (LOCK_STATE_MGR_SRV) {
            if (STATE_MGR_SRV == null) {
                final Context ctx = new InitialContext();
                final PortletServiceHome psh = (PortletServiceHome) ctx
                        .lookup(PortletStateManagerService.JNDI_NAME);
                STATE_MGR_SRV = psh.getPortletService(PortletStateManagerService.class);
            }
        }
    }
    return STATE_MGR_SRV;
}

private static PortletFriendlySelectionServiceHome getFriendlySelectionServiceHome() throws NamingException, PortletServiceUnavailableException {
    if (FRIENDLY_SEL_SRV == null) {
        synchronized (LOCK_FRIENDLY_SEL_SRV) {
            if (FRIENDLY_SEL_SRV == null) {
                final Context ctx = new InitialContext();
                final PortletServiceHome psh = (PortletServiceHome) ctx
                        .lookup(PortletFriendlySelectionServiceHome.JNDI_NAME);
                FRIENDLY_SEL_SRV = psh.getPortletService(PortletFriendlySelectionServiceHome.class);
            }
        }
    }
    return FRIENDLY_SEL_SRV;
}

private static ContentMappingInfoHome getContentMappingInfoHome() throws NamingException {
    if (CONTENT_MAPPING_HOME == null) {
        synchronized (LOCK_CONTENT_MAPPING_HOME) {
            if (CONTENT_MAPPING_HOME == null) {
                final Context ctx = new InitialContext();
                CONTENT_MAPPING_HOME = (ContentMappingInfoHome) ctx.lookup(ContentMappingInfoHome.JNDI_NAME);
            }
        }
    }
    return CONTENT_MAPPING_HOME;
}
* Get the web content service
  *
  * @return Reference to the web content service
  * @throws NamingException
  * if creation of {@link InitialContext} or context lookup fails
  */
private static WebContentService getWebContentService() throws NamingException {
  if (WEB_CONTENT_SRV == null) {
    synchronized (LOCK_WEB_CONTENT_SRV) {
      if (WEB_CONTENT_SRV == null) {
        final Context ctx = new InitialContext();
        WEB_CONTENT_SRV = (WebContentService) ctx.lookup("portal:service/wcm/WebContentService");
      }
    }
  }
  return WEB_CONTENT_SRV;
}

/**
 * Get the content model provider
 *
 * @return Reference to the content model provider
 * @throws NamingException
 * if creation of {@link InitialContext} or context lookup fails
 * @throws PortletServiceUnavailableException
 * if the service is not available
 */
private static ContentModelProvider getContentModelProvider() throws NamingException,
  PortletServiceUnavailableException {
  if (CONTENT_MODEL_PROVIDER == null) {
    synchronized (LOCK_CONTENT_MODEL_PROVIDER) {
      if (CONTENT_MODEL_PROVIDER == null) {
        final Context ctx = new InitialContext();
        final PortletServiceHome psh = (PortletServiceHome) ctx.lookup(ContentModelProvider.JNDI_NAME);
        CONTENT_MODEL_PROVIDER = psh.getPortletService(ContentModelProvider.class);
      }
    }
  }
  return CONTENT_MODEL_PROVIDER;
}

Example 2: Filter class:

This sample filter generates the friendly URL.

The filter performs several steps to create the friendly URL:
1. The filter determines the target portal page from one of the following sources:
   • The configuration of the web content viewer.
   • Any web content pages that have a content association to the content for which the URL is generated.
   • A target page that is specified by the UrlCmpnt tag.
2. If a target page is identified, the filter verifies that the page is a web content page with a content association. The filter then validates that the content for which the URL is generated is a child of the site area that is mapped to the page. If the content to render is not a child of the site area that is associated with the page, the filter writes a new URL.
3. The filter then writes the friendly URL by combining the following information:
   • The friendly URL name of the target page.
   • The path to the content, relative to the site area that is associated with the target page.
package com.ibm.workplace.wcm.api.samples;

import java.io.*;
import java.net.*;
import java.util.*;
import java.util.logging.*;
import java.util.regex.*;
import javax.portlet.*;
import com.ibm.portal.*;
import com.ibm.portal.content.*;
import com.ibm.portal.identification.*;
import com.ibm.portal.mappingurl.*;
import com.ibm.portal.resolver.friendly.*;
import com.ibm.portal.resolver.friendly.accessors.url.*;
import com.ibm.portal.resolver.friendly.helper.*;
import com.ibm.portal.resolver.friendly.service.*;
import com.ibm.portal.serialize.*;
import com.ibm.portal.services.contentmapping.*;
import com.ibm.portal.services.contentmapping.exceptions.*;
import com.ibm.portal.state.*;
import com.ibm.portal.state.accessors.selection.*;
import com.ibm.portal.state.exceptions.*;
import com.ibm.workplace.wcm.api.*;
import com.ibm.workplace.wcm.api.exceptions.*;
import com.ibm.workplace.wcm.api.extensions.url.*;

/**
 * Content URL generation filter that tries to generate stateless friendly URLs
 * for web content pages.
 * The filter that writes the friendly URL does the following steps to generate
 * the friendly URL
 *<ol>
 * <li>It determines the target portal page from one of the following sources The
 * Web Content Viewer configuration Web content pages that have a content
 * mapping for the content the URL is generated for A target page specification
 * from the WCM [UrlCmpnt] tag
 * </li>
 * <li>If a page could be determined it checks if the page is a web content page
 * i.e. if the page has a content mapping assign. It then validates that the
 * content the URL is generated for is a children of the site area mapped to the
 * page. In case the content is not a children of the site area mapped to the
 * page new URL is written by this filter.
 * </li>
 * <li>Finally the friendly URL is written that is build from the friendly URL
 * name of the target page appended with the content path relative to the site
 * area mapped to the target page.</li>
 * </ol>
 *<p>Note:</b> In order to use the following sample filter all pages a content URL is
 * generated for need to be web content pages with a friendly name assigned and
 * a default content mapping that points to a parent of the content.</p>
 */
public class FriendlyUrlGenerationFilter implements ContentUrlGenerationFilter {

/** logger */
private static final Logger LOGGER = Logger.getLogger(FriendlyUrlGenerationFilter.class.getName());

/** the path separator */
private static final String PATH_SEPARATOR = "/";

/** regular expression pattern to split a path into segments */
private static final Pattern PATH_SEPARATOR_PATTERN = Pattern.compile(PATH_SEPARATOR);

/** friendly selection service */
private final FriendlySelectionService friendlySelectionService;

/** content model */
private final ContentModel<ContentNode> contentModel;

/** WCM workspace */
private final Workspace workspace;

/** identification service */
private final Identification identification;

/** state manager */
private final PortletStateManager stateManager;

/** url mapping model */
private final MappingURLTreeModel urlMappingModel;

/** content mapping info home */
private final ContentMappingInfoHome contentMappingInfoHome;

/** selection accessor */
private final SelectionAccessorFactory selectionFactory;

/** factory for friendly URLs */
private final FriendlyURLFactory friendlyUrlFactory;

/** the currently selected page */
private ObjectID currentPage;

/** Create a new filter instance. This should be called once per render request */
/**
 * @param friendlySelectionService
 * The friendly selection service
 * @param contentModel
 * The content model
 * @param workspace
 * The WCM workspace
 * @param identification
 * The identification service
 * @param stateManager
 * The state manager service
 * @param urlMappingTreeModel
 * The url mapping model
 * @param contentMappingInfoHome
 * The content mapping home interface
 * @throws CannotInstantiateAccessorException
 * If instantiation of state selection accessor factory fails
 * @throws UnknownAccessorTypeException
 * If instantiation of state selection accessor factory fails
 */
public FriendlyUrlGenerationFilter(final FriendlySelectionService friendlySelectionService,
                                    final ContentModel<ContentNode> contentModel, final Workspace workspace,
                                    final Identification identification, final PortletStateManager stateManager,
                                    final MappingURLTreeModel urlMappingModel, final ContentMappingInfoHome contentMappingInfoHome)
                                    throws UnknownAccessorTypeException, CannotInstantiateAccessorException {

    final boolean isLogging = LOGGER.isLoggable(Level.FINEST);
    if (isLogging) {
        LOGGER.entering(getClass().getName(), "<init>", new Object[] { friendlySelectionService, contentModel,
            workspace, identification, stateManager, urlMappingModel, contentMappingInfoHome });
    }
    }
this.friendlySelectionService = friendlySelectionService;
this.friendlyUrlFactory = friendlySelectionService.getURLFactory();
this.contentModel = contentModel;
this.workspace = workspace;
this.identification = identification;
this.stateManager = stateManager;
this.urlMappingModel = urlMappingModel;
this.contentMappingInfoHome = contentMappingInfoHome;
this.selectionFactory = stateManager.getAccessorFactory(SelectionAccessorFactory.class);

if (isLogging) {
    LOGGER.exiting(getClass().getName(), "<init>");
}

@Override
public void dispose() {
    final boolean isLogging = LOGGER.isLoggable(Level.FINEST);
    if (isLogging) {
        LOGGER.entering(getClass().getName(), "dispose");
    }

    // dispose all request specific services
    this.friendlySelectionService.dispose();
    this.stateManager.dispose();

    if (isLogging) {
        LOGGER.exiting(getClass().getName(), "dispose");
    }
}

@Override
public void writeURL(final ContentUrlGenerationRequest request, final ContentUrlGenerationResponse response,
                      final ContentUrlGenerationFilterChain chain) throws ContentUrlGenerationException, IOException {
    final boolean isLogging = LOGGER.isLoggable(Level.FINEST);
    if (isLogging) {
        LOGGER.entering(getClass().getName(), "writeURL", new Object[]{ request.getContentPath(false) });
    }

    // as we need to use the path to lookup the item in WCM we need the
    // decoded version
    final String contentPath = request.getContentPath(false);
    if (contentPath != null) {
        // Check if we should generate a URL that publishes to the
        // current
        // or another page or uses the dynamic publishing
        final PortletContextSharingConfig ctxSharingConfig = request.getPortletContextSharingConfig();
        final PublishConfig publishConfig = ctxSharingConfig.getPublishConfig();
        final PortletRequest portletRequest = request.getPortletRenderRequest();
        final PortletResponse portletResponse = request.getPortletRenderResponse();

        ObjectID targetPageId = null;
        try {
            // determine the target page. The target page is determined
            // from either a dynamic target page override (i.e. on the
            // UrlCmpnt tag), a web content mapping on a page or from the
            // portlet configuration

            // check if a dynamic page target as been set as it can be
            // set on the WCM UrlCmpnt tag
            final TargetPageConfig targetPageDynamic = request.getDynamicTargetPageOverride();
            if (targetPageDynamic != null) {
                // lookup the page from the dynamic target page override
                targetPageId = getTargetPage(portletRequest, portletResponse, targetPageDynamic);
            } else {

        ...
if (publishConfig.getMode() == PublishConfig.MODE_DYNAMIC) {
  // lookup the target page from content mappings
  targetPageId = lookupTargetPage(portletRequest, portletResponse, contentPath);
} else {
  // target page is determined from portlet configuration
  final TargetPageConfig targetPagePortletConfig = publishConfig.getTargetPage();
  if (targetPagePortletConfig != null) {
    // lookup the page from the portlet target page configuration
    targetPageId = getTargetPage(portletRequest, portletResponse, targetPagePortletConfig);
  }
}

if (targetPageId != null) {
  // check if the path of the content is a children of the site area mapped to the page and get the path relative to this site area
  final String relativePathInfo = getRelativePathInfo(contentPath, targetPageId);
  if (relativePathInfo != null && !relativePathInfo.isEmpty()) {
    // write the friendly URL to the page and the relative path information added
    final FriendlyURL url = this.friendlyUrlFactory.newURL(com.ibm.portal.state.Constants.Clone.EMPTY_COPY);
    url.setSelection(targetPageId);
    url.setPathInfo(relativePathInfo);
    url.writeDispose(response.getWriter());
  } else {
    if (isLogging) {
      LOGGER.logp(Level.FINEST, getClass().getName(), "writeURL", "Content [{0}] is not a children of the site area mapped to page with ID [{1}]",
               new Object[] { contentPath, targetPageId });
    }
    // the content is not a children of the site area mapped to the target page so forward the request to the chain
    chain.writeURL(request, response);
  }
} else {
    if (isLogging) {
      LOGGER.logp(Level.FINEST, getClass().getName(), "writeURL", "No target page could be determined for content [{0}]", new Object[] { contentPath });
    }
    // no target page could be determined let the content URL generation chain handle the request
    chain.writeURL(request, response);
  }
}

} catch (SerializationException e) {
  throw new ContentUrlGenerationException(e);
} catch (ModelException e) {
  throw new ContentUrlGenerationException(e);
} catch (StateException e) {
  throw new ContentUrlGenerationException(e);
} catch (ContentMappingException e) {
  throw new ContentUrlGenerationException(e);
} catch (WCMException e) {
  throw new ContentUrlGenerationException(e);
}

} else {
  // no content path was given
}
// let the content URL generation chain handle the request
chain.writeURL(request, response);
}

if (isLogging) {
LOGGER.exiting(getClass().getName(), "writeURL");
}

/**
 * Lookup the best matching target web content page for the content
 * @param portletRequest
 * The current portlet request
 * @param portletResponse
 * The current portlet request
 * @param contentPath
 * The path of the content
 * @return The {@link ObjectID} of page found or <code>null</code>
 * @throws ContentMappingException
 * If an error occurred loading a content mapping
 * @throws ModelException
 * If an exception occurred while accessing a model object
 * @throws WCMException
 * If an exception occurred while accessing the WCM repository
 * @throws StateException
 * If an error occurred working with the portal state objects
 */
protected ObjectID lookupTargetPage(final PortletRequest portletRequest, final PortletResponse portletResponse,
final String contentPath) throws ContentMappingException, ModelException, WCMException, StateException {

final boolean isLogging = LOGGER.isLoggable(Level.FINEST);
if (isLogging) {
LOGGER.entering(getClass().getName(), "lookupTargetPage", new Object[] { contentPath });
}

ObjectID result = null;

// get the ID of the published item addressed by the content path
final DocumentIdIterator documentsIt = this.workspace.findByPath(contentPath,
Workspace.WORKFLOWSTATUS_PUBLISHED);
if (documentsIt.hasNext()) {
// get the IDs of the content and all its parents
final LinkedList<String> resourceIds = new LinkedList<String>();
final DocumentId documentId = documentsIt.next();
resourceIds.push(documentId.getId());
// load the IDs of the parents of the item
DocumentId parentID = documentId;
do {
Document doc = this.workspace.getById(parentId);
parentId = null;
if (doc instanceof Content) {
parentId = ((Content) doc).getDirectParent();
} else if (doc instanceof ContentLink) {
parentId = ((ContentLink) doc).getParentId();
} else if (doc instanceof SiteFrameworkContainer) {
parentId = ((SiteFrameworkContainer) doc).getParent();
}
if (parentId != null) {
resourceIds.push(parentId.getId());
}
} while (parentId != null);
// add the library of the content to the beginning
resourceIds.push(documentId.getContainingLibrary().getId());

Chapter 8. Developing
if (isLogging) {
    LOGGER.logp(Level.FINEST, getClass().getName(), "lookupTargetPage",
            "Lookup up best matching web content page for resources [{0}] using the following IDs [{1}]",
            new Object[] { contentPath, resourceIds });
}

// lookup the best matching web content page
final ContentMappingLocator contentMappinglocator = this.contentMappingInfoHome.getContentMappingLocator();
final LongestPathMatch match = contentMappinglocator.getLongestPathMatch(resourceIds,
        getCurrentPage(portletRequest, portletResponse), new ContentMappingFilter() {
            public void filterEntitledMappings(List<? extends ContentMapping> mappings) {
                // filter out pages we cannot locate e.g. the
                // user doesn't have access to or if the page is
                // disabled
                final Locator<ContentNode> contentNodeLocator = FriendlyUrlGenerationFilter.this.contentModel
                        .getLocator();
                final Iterator<? extends ContentMapping> mappingsIt = mappings.iterator();
                while (mappingsIt.hasNext()) {
                    if (contentNodeLocator.findByID(mappingsIt.next().getResourceID()) == null) {
                        mappingsIt.remove();
                    }
                }
            }
        });

// if at least one match was found take the suggest content
// mapping further candidates might be found
final ContentMapping contentMapping = match.getContentMapping();
if (contentMapping != null) {
    result = contentMapping.getResourceID();
}

if (isLogging) {
    LOGGER.exiting(getClass().getName(), "lookupTargetPage", result);
}
return result;

/**
 * Get the [ObjectID] of the target page from a target page
 * configuration.
 *
 * @param portletRequest
 * The current portlet request
 * @param portletResponse
 * The current portlet response
 * @param targetPageConfig
 * The target page configuration
 * @return The [ObjectID] of the target page
 *
 * @throws SerializationException
 * If the a page ID given as a character string cannot be
 * serialized to an [ObjectID]
 * @throws ModelException
 * If an exception occurred while accessing a model object
 * @throws StateException
 * if an error occurred working with the portal state objects
 */
protected ObjectID getTargetPage(final PortletRequest portletRequest, final PortletResponse portletResponse,
        final TargetPageConfig targetPageConfig) throws SerializationException, ModelException, StateException {
    final boolean isLogging = LOGGER.isLoggable(Level.FINEST);
    if (isLogging) {
        LOGGER.entering(getClass().getName(), "getTargetPage", new Object[] { targetPageConfig });
    }
}
```java
ObjectId result = null;
if (targetPageConfig != null) {
    if (targetPageConfig.useCurrentPage()) {
        result = getCurrentPage(portletRequest, portletResponse);
    } else {
        final String pagePath = targetPageConfig.getPagePath();
        if (pagePath != null && !pagePath.isEmpty()) {
            // try to lookup the page treating the path as a URL mapping
            result = getPageByUrlMapping(portletRequest, portletResponse, pagePath);
            if (result == null) {
                // if no mapping was found, check if the path is a
                // valid friendly URL
                final List<ObjectId> pages = getPagesByFriendlyUrl(portletRequest, portletResponse, pagePath);
                if (pages != null && !pages.isEmpty()) {
                    // if multiple pages are found for simplicity use
                    // the
                    // first page more advance URL generation filter
                    // could
                    // do a disambiguation here and e.g. let the user
                    // choose
                    // what page to use
                    result = pages.get(0);
                }
            }
        } else {
            result = getPageById(targetPageConfig.getPageId());
        }
    }
}
if (isLogging) {
    LOGGER.exiting(getClass().getName(), "getTargetPage", result);
} return result;
```

```java
public ObjectId getPageById(final String pageId) throws SerializationException {
    final boolean isLogging = LOGGER.isLoggable(Level.FINEST);
    if (isLogging) {
        LOGGER.entering(getClass().getName(), "getPageById", new Object[] { pageId });
    }
    ObjectId result = null;
    if (pageId != null && !pageId.isEmpty()) {
        // de-serialize the ID
        result = this.identification.deserialize(pageId);
    }
    if (isLogging) {
        LOGGER.exiting(getClass().getName(), "getPageById", result);
    }
    return result;
}
```
/**
 * Get the [ObjectID] of the current page
 * @param portletRequest
 * The current portlet request
 * @param portletResponse
 * The current portlet request
 * @return The [ObjectID] of the current page
 * @throws StateException
 * if an error occurred working with the portal state objects
 */
protected ObjectID getCurrentPage(final PortletRequest portletRequest, final PortletResponse portletResponse)
throws StateException {
    final boolean isLogging = LOGGER.isLoggable(Level.FINEST);
    if (isLogging) {
        LOGGER.entering(getClass().getName(), "getCurrentPage");
    }
    if (currentPage == null) {
        final SelectionAccessor selectionAcc = this.selectionFactory.getSelectionAccessor(
            this.stateManager.getStateHolder());
        try {
            currentPage = selectionAcc.getSelection();
        } finally {
            selectionAcc.dispose();
        }
    }
    if (isLogging) {
        LOGGER.exiting(getClass().getName(), "getCurrentPage", currentPage);
    }
    return currentPage;
}

/**
 * Get the list of [ObjectID] of all page that are addressed by the
 * passed friendly name
 * @param portletRequest
 * The current portlet request
 * @param portletResponse
 * The current portlet request
 * @param friendlyName
 * The friendly name
 * @return List of all pages that are addressed by the passed friendly name
 * @throws ModelException
 * If looking up the page from a friendly URL fails
 * @throws StateException
 * if the state could not be accessed
 */
protected List<ObjectID> getPagesByFriendlyUrl(final PortletRequest portletRequest,
    final PortletResponse portletResponse, final String friendlyName) throws ModelException, StateException {
    final boolean isLogging = LOGGER.isLoggable(Level.FINEST);
    if (isLogging) {
        LOGGER.entering(getClass().getName(), "getPagesByFriendlyUrl", new Object[] { friendlyName });
    }
    List<ObjectID> result = null;
    if (friendlyName != null && !friendlyName.isEmpty()) {
        final SelectionResult bean = new DefaultSelectionResult();
        if (isLogging) {
            LOGGER.exiting(getClass().getName(), "getPagesByFriendlyUrl", new Object[] { friendlyName });
        }
    }
}
// the resulting node list is already AC filtered as a
// result of using a performing navigation model.
final List<ObjectID> nodelist = bean.getNodes();
if (nodelist != null && !nodelist.isEmpty() && bean.getFriendlyPath() != null) {
    result = nodelist;
}
}

if (isLogging) {
    LOGGER.exiting(getClass().getName(), "getPagesByFriendlyUrl", result);
}
return result;
}

/**
 * Get the {ObjectID} of the page addressed by the passed compound
 * name of a url mapping or <code>null</code> if no corresponding URL
 * mapping or page exists or if the current user does not have access to it.
 *
 * @param portletRequest
 * The current portlet request
 * @param portletResponse
 * The current portlet request
 * @param urlMapping
 * The compound name of the url mapping
 * @return {ObjectID} of the page or <code>null</code>
 * @throws ModelException
 * If an exception occurred while accessing the url mapping
 * model
 */
protected ObjectID getPageByUrlMapping(final PortletRequest request, final PortletResponse response,
final String urlMapping) throws ModelException {
final boolean isLogging = LOGGER.isLoggable(Level.FINEST);
if (isLogging) {
    LOGGER.entering(getClass().getName(), "getPageByUrlMapping", new Object[] { urlMapping });
}

ObjectID result = null;
if (urlMapping != null && !urlMapping.isEmpty()) {
    final BestMatchResult searchResult;
    // different to friendly names a URL mapping must not begin with a /
    if (urlMapping.charAt(0) == PATH_SEPARATOR.charAt(0)) {
        searchResult = this.urlMappingModel.getLocator().findBestMatch(urlMapping.substring(1));
    } else {
        searchResult = this.urlMappingModel.getLocator().findBestMatch(urlMapping);
    }
    if (searchResult != null) {
        final Context mappingCtx = searchResult.getContext();
        if (ObjectTypeConstants.PORTAL_URL.getType().equals(mappingCtx.getAssignedObjectType())) {
            final PortalURL url = (PortalURL) mappingCtx.getAssignedObject();
            result = url.getReferencedResourceID();
        }
    }
}

if (isLogging) {
    LOGGER.exiting(getClass().getName(), "getPageByUrlMapping", result);
}
return result;
}

/**
 * Returns the path for the given content path relative to the site area
 * mapped to the target page.
 */
* Returns <code>null</code> if there is no content mapping set for the
* target page that is appropriate for the targeted content item.
* 
* @param contentPath
* The fully qualified path of the target content item. Must not
* be <code>null</code>.
* @param pageId
* The object ID of the target page. Must not be
* <code>null</code>.
* @return The relative path which is the remainder of the content path
* after cutting off the content mapping prefix. May return
* <code>null</code>.
* @throws ContentMappingException
* If an exception occurred during lookup of the content mapping
* @throws WCMException
* If an exception occurred while accessing the WCM repository
* @throws UnsupportedEncodingException
* A requested character encoding is not supported
*/
protected String getRelativePathInfo(final String contentPath, final ObjectID pageId)
throws ContentMappingException, WCMException, UnsupportedEncodingException {
    final boolean isLogging = LOGGER.isLoggable(Level.FINEST);
    if (isLogging) {
        LOGGER.entering(getClass().getName(), "getRelativePathInfo", new Object[]{ contentPath, pageId });
    }

    String result = null;
    final ContentMapping contentMapping = getDefaultContentMapping(pageId);
    if (contentMapping != null) {
        // lookup the path of the site area mapped to the page
        String pathMapping = contentMapping.getContentPath();
        if (pathMapping == null || pathMapping.isEmpty()) {
            // lets lookup the path from the id
            final String mappedId = contentMapping.getContentID();
            if (mappedId != null && !mappedId.isEmpty()) {
                pathMapping = this.workspace.getPathById(this.workspace.createDocumentId(mappedId), false, true);
            }
        }
        if (isLogging) {
            LOGGER.logp(Level.FINEST, getClass().getName(), "getRelativePathInfo",
                        "Page with ID [{0}] is mapped to [{1}]", new Object[] { pageId, pathMapping });
        }

        if (isLogging) {
            LOGGER.logp(Level.FINEST, getClass().getName(), "getRelativePathInfo",
                        "Calculate relative path = contentPath - mappingPath"
                        "Page with ID [{0}] is mapped to [{1}]", new Object[] { pageId, pathMapping });
        }

        // calculate relative path = contentPath - mappingPath
        if (pathMapping != null && !pathMapping.isEmpty()) {
            // check if the content path is a children of the mapped path
            // to do this split the path into its segments
            if (pathMapping.charAt(0) == PATH_SEPARATOR.charAt(0)) {
                pathMapping = pathMapping.substring(1);
            }
            final String[] partsPathMapping = PATH_SEPARATOR_PATTERN.split(pathMapping);
            // also split path of content
            String pathContent = contentPath;
            if (pathContent.charAt(0) == PATH_SEPARATOR.charAt(0)) {
                pathContent = pathContent.substring(1);
            }
            final String[] partsPathContent = PATH_SEPARATOR_PATTERN.split(pathContent);

            // check if the content is a children of the mapped path
            if (partsPathMapping.length <= partsPathContent.length) {
                boolean isDescendant = true;
                for (int i = 0; i < partsPathMapping.length && isDescendant; i++) {
                    if (!partsPathMapping[i].equalsIgnoreCase(partsPathContent[i])) {
                        isDescendant = false;
                    }
                }
            }
        }
    }
    return result;
}
isDescendant = false;
}
if (isDescendant) {
    // determine how many descendant levels are between the
    // content and the mapped site area
    final int descendantLevels = partsPathContent.length - partsPathMapping.length;
    if (descendantLevels > 0) {
        // build children path which is everything after the
        // parent
        final StringBuilder tmp = new StringBuilder();
        for (int i = 0; i < descendantLevels; i++) {
            tmp.append(PATH_SEPARATOR);
            tmp.append(URLEncoder.encode(partsPathContent[partsPathMapping.length + i], "UTF-8"));
        }
        result = tmp.toString();
    }
}
if (isLogging) {
    LOGGER.exiting(getClass().getName(), "getRelativePathInfo", result);
} return result;
}
/**
 * Get the default content mapping of a page or <code>null</code> if no such
 * mapping exists
 *
 * @param pageId
 * @return The default mapping of the page or <code>null</code> if no
 * default mapping could be determined.
 *
 * @throws ContentMappingDataBackendException
 * If an exception occurred during lookup of the content mapping
 */
protected ContentMapping getDefaultContentMapping(final ObjectID pageId) throws ContentMappingDataBackendException {
    final boolean isLogging = LOGGER.isLoggable(Level.FINEST);
    if (isLogging) {
        LOGGER.entering(getClass().getName(), "getDefaultContentMapping", new Object[] { pageId });
    }

    // get the page default content mapping as friendly url path info is
    // only set for default or system content mapping
    final ContentMappingInfo contentMappingInfo = this.contentMappingInfoHome.getContentMappingInfo(pageId);
    ContentMapping result = contentMappingInfo.getDefaultContentMapping();
    if(result == null) {
        // use system mapping as default
        result = contentMappingInfo.getSystemContentMapping();
    }
    if (isLogging) {
        LOGGER.exiting(getClass().getName(), "getDefaultContentMapping", result);
    }
    return result;
}

Example 2: plugin.xml file:

You can use this sample plugin.xml file to register the sample filter.
Deploying custom plug-in applications

You must deploy your custom plug-in applications on your server before they can be used in your web content system.

To ensure that the new web content class is available each time your server is started, register the ear file in the WebSphere Integrated Solutions Console:

1. Select Applications.
2. Select New Application.
6. Select the tick box for the custom action module.
7. In the Clusters and servers section, select WebSphere_Portal and select Apply.
8. Keep selecting Next until the end when you select Finish.
9. The screen titled Installing is shown. Select Manage Applications.
10. Locate and click the application you installed. The default name is the display name defined in the application.xml file in your ear file.
11. Under Detail Properties select Startup behavior.
12. Under General Properties modify the Startup order to be the same weight as "wcm" and select Apply. By default, the weight is 20.
13. Select Save directly to master configuration.
14. To immediately start the application you installed, select the application and click Start.

To update an existing ear file:
1. Select Applications.
2. Select Application Types.
3. Select WebSphere enterprise applications.
4. Search for the Application name.
5. Select the tick box for the custom action application and click Update.
6. Select Replace the entire application and browse for the ear file.
7. Keep selecting Next until the end when you select Finish.
8. The screen titled 'updating' is shown. Select Save to Master Configuration and then click save.

Instrumenting web content for Active Site Analytics

You can collect information from web content for Active Site Analytics.
For collecting information about the web content that the portal renders, use the following microformat tags:

**asa.wcm.content_item.path**  
Use this tag to identify the content path. The tag contains the unique identifier of the content item in IBM Web Content Manager.

**asa.wcm.content_item.title**  
Use this tag to identify the content title. The tag contains the display title of the content item.

**asa.wcm.content_item.authors**  
Use this tag to identify the authors of the content. The tag contains one of the authors of the content item.

**asa.wcm.content_item.lastmodified**  
Use this tag to identify the last modification date of the content. The tag contains the date on which the content item was last modified.

Web Content Manager provides the following methods for collecting information about web content:

- Using the analytics data rendering plug-in tag.
- Using the sample HTML component for Active Site Analytics that is provided with Web Content Manager.
- Using the default microformat tags that are supported by web content viewers.

**Related tasks:**  
[Analyzing user behavior by Active Site Analytics](#)

### Using the analytics data rendering plug-in tag

Use the AnalyticsData rendering plug-in tag to inject microformats for Active Site Analytics into your web content.

The microformats gather information about the content items that are rendered on your portal pages. You can insert more than one tag into your content.

Add the AnalyticsData plug-in tag to your content or presentation template design. Each tag can contain only one attribute. Use the following syntax:

```
[Plugin:AnalyticsData property="" | element="" | value=""
  css-class=""]
```

To select a value that is used in the microformat for Active Site Analytics, use one of the following attributes:

**property**  
Use this attribute to select one of the supported properties of the web content item. This attribute can take the following values:

- **id**  
The unique identifier of the content.
- **title**  
The display title of the content.
- **path**  
The unique path of the content in your library.
- **lastmodified**  
The date when the content was last modified.
- **authors**  
The author or authors of the content. If the content has more than one author, one microformat tag is written for each author.
**element**

Use this attribute to select an element from the content. The element must be a text or short text element type. For the value of this attribute, specify the name of the element that you want to be selected.

**value**

Use this attribute to pass a text value to the plug-in. If you set this attribute, you must also set the CSS class value by using the attribute `css-class`. See the following description of this attribute for details.

**Note**: In a single reference to a tag, you can use only one of the previous attributes. If you want to use more than one attribute, add another reference to the tag.

**css-class**

Use this attribute to control which CSS class is used for the microformat tags. This attribute is mandatory when you specify the `value` attribute. The attribute is optional when you use the `property` attribute or the `element` attribute. If you do not specify this attribute with the `property` attribute or the `element` attribute, the CSS class is determined from the value of the `property` attribute or the `element` attribute. In this case, the CSS class is generated by prefixing `asa.wcm.content_item.` and the value of the `property` attribute or the `element` attribute.

**Examples**

```
[Plugin:AnalyticsData property="title"]
This setting selects the display title of the content item. The following microformat is inserted in the page:
<span style="display:none" class="asa.wcm.content_item.title">Display title</span>
```

```
[Plugin:AnalyticsData property="authors" css-class="com.acme.content.authors"]
This setting selects the authors of the content item. If you use this setting and a custom CSS class, the following microformats are inserted in the page:
<span style="display:none" class="com.acme.content.authors">Author 1</span>
<span style="display:none" class="com.acme.content.authors">Author 2</span>
```

```
[Plugin:AnalyticsData element="element1"]
This setting selects the text value of the element on the content item. The following microformat is inserted in the page:
<span style="display:none" class="asa.wcm.content_item.element1">Value of element with name element1</span>
```

```
[Plugin:AnalyticsData element="element1" css-class="asa.wcm.content_item.path"]
This setting selects the text value of the element on the content item. The following microformat is inserted in the page:
<span style="display:none" class="asa.wcm.content_item.path">Value of element with name element1</span>
```

```
[Plugin:AnalyticsData value="Some text" css-class="asa.wcm.content_item.title"]
This setting uses the text specified by the `value` attribute. The following microformat is inserted in the page:
<span style="display:none" class="asa.wcm.content_item.title">Some text</span>
```

```
[Plugin:AnalyticsData css-class="asa.wcm.content_item.title" value="[Property context='current' type='content' field='title']"]
This setting passes the value of the nested `[Property]` tag to the plug-in. The following microformat is inserted in the page:
<span style="display:none" class="asa.wcm.content_item.title">Display title from nested property tag</span>
```
Using the sample HTML component for Active Site Analytics

The HTML – Analytics component is a sample HTML component that you can use to instrument web content for Active Site Analytics. You can use this component to insert the supported microformat tags for web content into your content or presentation templates.

The HTML – Analytics component is located in the Web Content Templates library. You can adapt this sample to the requirements of your aggregator and Active Site Analytics setup. By default, the sample inserts the following microformat tags:

- asa.wcm.content_item.path
- asa.wcm.content_item.title
- asa.wcm.content_item.id
- asa.wcm.content_item.authors
- asa.wcm.content_item.lastmodified

Insert the HTML – Analytics component in your content or presentation template design by adding the [Component] tag. For example:

[Component name="Web Content Templates/HTML - Analytics"]

Enabling default microformat support in web content viewers

Web content viewers provide support for Active Site Analytics microformats by default. You can use this support to inject microformats into your content design or presentation templates.

The viewer supports the following microformat tags:

- asa.wcm.content_item.title
- asa.wcm.content_item.path

Edit the portlet preferences for the web content viewer, and set the value of the WCM_ENABLE_ASA_TAGS preference to true. By default, the preference value is false.

Note: The default viewer has the unique name of ibm.portal.Web.Content Viewer.Jsr286.

- To use the administration user interface to set the preference for all instances of the viewer, complete these steps:
  1. Click Administration > Portlet Management > Portlets.
  2. Locate the web content viewer from the list.
  3. Click the Configure portlet icon, and set the preference value.

- To set the preference for all instances or only a single instance of the viewer, you can also use the XML configuration interface.

After setting the preference, the viewer automatically inserts the microformat tags into each piece of content that it renders.
Helper class samples for web content context

You can create the helper classes PortletWCMContextHelper, PortalWCMContextHelper and WCMContextHelper from the sample code that is provided here to programmatically determine the current web content context. The context indicates a content item or site area that is rendered by a web content viewer.

**Important:** These helper classes are not provided by default with IBM Web Content Manager. Example code is provided here that you can take and use to implement the classes yourself.

The helper classes can be used by custom themes or custom portlets that need to render information that is related to the current context.

The web content viewer determines the context to be rendered by evaluating several conditions in the following order:

1. Private render parameter
2. `path-info` parameter
3. Public render parameter
4. Portlet configuration setting for the web content viewer
5. Web content association defined for the page

The viewer evaluates each condition in turn until it finds a valid context. As soon as the viewer finds a context, remaining conditions are not evaluated.

Private render parameters and portlet configuration settings are visible only to the web content viewer. However, `path-info` parameters, public render parameters, and content associations are visible to all portlets on a page and to portal code (for example, in a theme).

**PortletWCMContextHelper**

Use the PortletWCMContextHelper class to determine the web content context from within a portlet.

**Preparation**

To use the PortletWCMContextHelper class, configure the portlet to receive a public render parameter and a **path-info** parameter. Update the portlet.xml file for the portlet, and add the following entries inside the `<portlet-app>` tag:

```xml
<public-render-parameter>
  <description>WCM public context</description>
  <identifier>PUBLIC_CONTEXT</identifier>
</public-render-parameter>

<public-render-parameter>
  <description>Shared path-info parameter of WebSphere Portal</description>
  <identifier>PATH_INFO</identifier>
</public-render-parameter>
```

In addition, update the `<portlet>` tag with the following lines:

```xml
<supported-public-render-parameter>PUBLIC_CONTEXT</supported-public-render-parameter>
<supported-public-render-parameter>PATH_INFO</supported-public-render-parameter>
```
Implementation

After a new instance of the PortletWCMContextHelper is created, the portlet can invoke the getCurrentWCMContext method, passing in the parameters

`PortletRequest` and `PortletResponse`:

```java
public String getCurrentWCMContext(PortletRequest, PortletResponse)
```

This method returns a string that contains the content path that defines the current web content context. To determine the content path, the `getCurrentWCMContext` method performs the following checks:

1. The method reads the value of the `path-info` public render parameter. The method constructs the content path from the `path-info` parameter and the content association of the current page, when these conditions are true:
   - The `path-info` parameter is present.
   - The `friendly.pathinfo.enabled` property is enabled in the portal configuration service.

2. If the `path-info` parameter is not present or if the `friendly.pathinfo.enabled` property is disabled, the method reads the public render parameter. If the public render parameter is present, the method returns the value of this parameter.

3. If no public render parameter is present, the method evaluates the current page for a default content association. The method then returns the path of the content item that is mapped to the page.

Source of PortletWCMContextHelper

```java
package com.ibm.portal.extension;
import java.util.Map;
import javax.naming.Context;
import javax.naming.InitialContext;
import javax.naming.NamingException;
import javax.portlet.RenderRequest;
import javax.portlet.RenderResponse;
import com.ibm.portal.portlet.service.PortletServiceHome;
import com.ibm.portal.portlet.service.PortletServiceUnavailableException;
import com.ibm.portal.services.contentmapping.exceptions.ContentMappingDataBackendException;
import com.ibm.portal.state.PortletStateManager;
import com.ibm.portal.state.accessors.exceptions.InvalidSelectionNodeIdException;
import com.ibm.portal.state.accessors.exceptions.StateNotInRequestException;
import com.ibm.portal.state.exceptions.CannotInstantiateAccessorException;
import com.ibm.portal.state.exceptions.StateManagerException;
import com.ibm.portal.state.exceptions.UnknownAccessorTypeException;
import com.ibm.portal.state.service.PortletStateManagerService;
import com.ibm.workplace.wcm.api.exceptions.DocumentIdCreationException;
import com.ibm.workplace.wcm.api.exceptions.DocumentRetrievalException;
import com.ibm.workplace.wcm.api.exceptions.IllegalDocumentTypeException;
import com.ibm.workplace.wcm.api.exceptions.OperationFailedException;
import com.ibm.workplace.wcm.api.exceptions.ServiceNotAvailableException;

/**
 * Helper class to determine the current WCM context. This class can only be used from portlet code.
 * From portal code (e.g. theme) please use PortalWCMContextHelper instead.
 */

public class PortletWCMContextHelper extends WCMContextHelper {
    private final PortletStateManagerService stateManagerService;
```
public PortletWCMContextHelper() throws NamingException, PortletServiceUnavailableException {
    // this initialization needs to be done only once
    final Context ctx = new InitialContext();
    PortletServiceHome psh = (PortletServiceHome) ctx.lookup(PortletStateManagerService.JNDI_NAME);
    stateManagerService = psh.getPortletService(PortletStateManagerService.class);
}

public String getCurrentWCMContext(final RenderRequest request, final RenderResponse response)
    throws StateManagerException, UnknownAccessorTypeException, CannotInstantiateAccessorException,
    InvalidSelectionNodeIdException, StateNotInRequestException, ContentMappingDataBackendException,
    ServiceNotAvailableException, OperationFailedException, DocumentIdCreationException,
    DocumentRetrievalException, IllegalDocumentTypeException
{
    String contentPath = null;
    Map<String, String[]> publicParameter = request.getPublicParameterMap();
    // check path info
    if (publicParameter.containsKey("PATH_INFO")) {
        final String[] pathInfo = publicParameter.get("PATH_INFO");
        if (pathInfo != null && pathInfo.length > 0) {
            final PortletStateManager portletStateManager = stateManagerService.getPortletStateManager(request, response);
            String contentMapping = getContentMapping(portletStateManager, portletStateManager.getStateHolder());
            contentPath = assembleContentPath(contentMapping, pathInfo);
        }
    }
    if (contentPath == null) {
        // check public WCM context
        contentPath = request.getParameter("PUBLIC_CONTEXT");
        if (contentPath == null) {
            // check content mapping
            final PortletStateManager portletStateManager = stateManagerService.getPortletStateManager(request, response);
            contentPath = getContentMapping(portletStateManager, portletStateManager.getStateHolder());
        }
    }
    return contentPath;
}
PortalWCMContextHelper

Use the PortalWCMContextHelper class to determine the web content context from within portal code, such as a theme.

To use the PortalWCMContextHelper class, create an instance of the helper class and invoke the getCurrentWCMContext method, passing in the parameters HttpServletRequest and HttpServletResponse:

```java
public String getCurrentWCMContext(HttpServletRequest request, HttpServletResponse response)
```

This method returns a string that contains the content path that defines the current web content context. To determine the content path, the getCurrentWCMContext method performs the following checks:

1. The method reads the value of the path-info public render parameter. The method constructs the content path from the path-info parameter and the content association of the current page, when these conditions are true:
   - The path-info parameter is present.
   - The friendly.pathinfo.enabled property is enabled in the portal configuration service.

2. If the path-info parameter is not present or if the friendly.pathinfo.enabled property is disabled, the method reads the public render parameter. If the public render parameter is present, the method returns the value of this parameter.

3. If no public render parameter is present, the method evaluates the current page for a default content association. The method then returns the path of the content item that is mapped to the page.

Source of PortalWCMContextHelper

```java
package com.ibm.portal.extension;
import java.util.Map;
import javax.naming.Context;
import javax.naming.InitialContext;
import javax.naming.NamingException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.xml.namespace.QName;
import com.ibm.portal.MetaData;
import com.ibm.portal.ModelException;
import com.ibm.portal.ObjectID;
import com.ibm.portal.content.ContentMetaDataModel;
import com.ibm.portal.content.ContentModel;
import com.ibm.portal.content.ContentNode;
import com.ibm.portal.model.ContentMetaDataModelHome;
import com.ibm.portal.model.ContentModelHome;
import com.ibm.portal.services.contentmapping.exceptions.ContentMappingDataBackendException;
import com.ibm.portal.state.StateHolder;
import com.ibm.portal.state.accessors.StateAccessor;
import com.ibm.portal.state.accessors.StateAccessorFactory;
import com.ibm.portal.state.accessors.portlet.PortletAccessorFactory;
import com.ibm.portal.state.accessors.portlet.SharedStateAccessor;
import com.ibm.portal.state.exceptions.CannotInstantiateAccessorException;
import com.ibm.portal.state.exceptions.StateException;
import com.ibm.portal.state.exceptions.UnknownAccessorTypeException;
import com.ibm.portal.state.service.PortalStateManagerServiceHome;
import com.ibm.workplace.wcm.api.exceptions.DocumentIdCreationException;
```
public class PortalWCMContextHelper extends WCMContextHelper {

    /** QName of the shared render parameter that holds the WCM public context */
    static final QName PUBLIC_WCM_CONTEXT_PARAM_QNAME =
        new QName("http://www.ibm.com/xmlns/prod/datatype/content", "context");

    /** QName of the shared render parameter that holds the path info */
    static final QName PUBLIC_PATH_INFO_QNAME =
        new QName("http://www.ibm.com/xmlns/prod/websphere/portal/publicparams", "path-info");

    /**
     * Page metadata key that controls the sharing scope for public render
     * parameter of portlets on this page
     */
    static final String PARAM_SHARING_SCOPE_KEY = "param.sharing.scope";

    private final PortalStateManagerServiceHome stateManagerServiceHome;
    private final ContentModelHome contentModelHome;
    private final ContentMetaDataModelHome contentMetaDataModelHome;

    /**
     * Initializes the PortalWCMContextHelper.
     * @throws NamingException
     */
    public PortalWCMContextHelper() throws NamingException {
        // this initialization needs to be done only once.
        final Context ctx = new InitialContext();
        stateManagerServiceHome =
            (PortalStateManagerServiceHome) ctx.lookup(PortalStateManagerServiceHome.JNDI_NAME);
        contentModelHome = (ContentModelHome) ctx.lookup(ContentModelHome.JNDI_NAME);
        contentMetaDataModelHome = (ContentMetaDataModelHome) ctx.lookup(ContentMetaDataModelHome.JNDI_NAME);
    }

    /**
     * Gets the WCM context of the current page. It checks path info, public WCM context render parameter
     * and content mapping in this order. A WCM context defined as private render parameter or
     * preference of the Web Content Viewer portlet is NOT returned.
     * @param request HttpServletRequest
     * @param response HttpServletResponse
     * @return String representation of a content path.
     * @throws StateException
     * @throws IllegalDocumentTypeException
     * @throws DocumentRetrievalException
     * @throws DocumentIdCreationException
     * @throws OperationFailedException
     * @throws ServiceNotAvailableException
     * @throws ContentMappingDataBackendException
     * @throws ModelException
     */
    public String getCurrentWCMContext(final HttpServletRequest request, final HttpServletResponse response)
        throws StateException, ContentMappingDataBackendException, ServiceNotAvailableException,
        OperationFailedException, DocumentIdCreationException, DocumentRetrievalException,
        IllegalDocumentTypeException, ModelException {
        String contentPath = null;
    }
final StateManagerService stateManagerService = 
    stateManagerServiceHome.getPortalStateManagerService(request, response);
final StateHolder currentState = getCurrentPortalState(request, stateManagerService);
final ContentModel<ContentNode> contentModel = 
    contentModelHome.getContentModelProvider().getContentModel(request, response);
final ContentMetaDataModel metaDataModel = 
    contentMetaDataModelHome.getContentMetaDataModelProvider().getContentMetaDataModel(request, response);

// find out public render parameter scope
final ObjectID currentPage = getCurrentPageID(stateManagerService, currentState);
final ContentNode page = contentModel.getLocator().findByID(currentPage);
final MetaData metadata = metaDataModel.getMetaData(page);

String publicRenderScope;
if (scope != null) {
    publicRenderScope = scope.toString();
} else {
    publicRenderScope = SharedStateAccessor.KEY_GLOBAL_PUBLIC_RENDER_PARAMETERS;
}

// load public render parameter
final PortletAccessorFactory portletAccFct = 
    stateManagerService.getAccessorFactory(PortletAccessorFactory.class);
final SharedStateAccessor sharedStateAcc = 
    portletAccFct.getSharedStateAccessor(publicRenderScope, currentState);
if (sharedStateAcc != null) {
    try {
        final Map<QName, String[]> sharedRenderParams = sharedStateAcc.getParameters();
        // check path info first.
        if (sharedRenderParams.containsKey(PUBLIC_PATH_INFO_QNAME)) {
            final String[] pathInfo = sharedRenderParams.get(PUBLIC_PATH_INFO_QNAME);
            if (pathInfo != null && pathInfo.length > 0) {
                String contentMapping = getContentMapping(stateManagerService, currentState);
                contentPath = assembleContentPath(contentMapping, pathInfo);
            }
        }
        // if there is no path info check the public WCM context.
        if (contentPath == null && sharedRenderParams.containsKey(PUBLIC_WCM_CONTEXT_PARAM_QNAME)) {
            final String[] values = sharedRenderParams.get(PUBLIC_WCM_CONTEXT_PARAM_QNAME);
            if (values != null && values.length > 0) {
                contentPath = values[0];
            }
        }
    } finally {
        sharedStateAcc.dispose();
    }
}

if (contentPath == null) {
    // check for a content mapping
    contentPath = getContentMapping(stateManagerService, currentState);
}
return contentPath;

/**
 * Gets the current portal state.
 * @param request HttpServletRequest
 * @param stateManagerService StateManagerService, entry point to the portal state API
 * @return the current portal state
 * @throws UnknownAccessorTypeException
 * @throws CannotInstantiateAccessorException
 * @throws StateNotInRequestException
 */
private StateHolder getCurrentPortalState(final HttpServletRequest request,
    final StateManagerService stateManagerService) throws UnknownAccessorTypeException,
CannotInstantiateAccessorException, StateNotInRequestException {
    StateHolder result = null;
    final StateAccessorFactory stateAccFac =
        (StateAccessorFactory) stateManagerService.getAccessorFactory(StateAccessorFactory.class);
    final StateAccessor stateAcc = stateAccFac.getStateAccessor();
    try {
        result = stateAcc.getStateHolder(request);
    } finally {
        stateAcc.dispose();
    }
    return result;
}

WCMContextHelper
The WCMContextHelper helper class is an abstract class that is used by the PortalWCMContextHelper class and the PortletWCMContextHelper class.

The WCMContextHelper class provides three methods:

get_CurrentPageID
   This method uses the State API to determine the currently selected page and returns the object ID of the page.

getContentMapping
   This method uses the Content Mapping Service to retrieve the content item that is associated with the current page. The association always contains the ID of the mapped content. After determining the content ID, the method uses the IBM Web Content Manager API to convert the ID to a content path. The method returns the content path of the mapped item.

assembleContentPath
   This method takes the path-info parameter, which has multiple values, and the content association of the current page. The method creates the web content context by appending all elements of the path-info parameter to the content mapping. The method then returns the result, which is the content path that defines the web content context.

Source of WCMContextHelper
import com.ibm.workplace.wcm.api.exceptions.IllegalArgumentException;
import com.ibm.workplace.wcm.api.exceptions.OperationFailedException;
import com.ibm.workplace.wcm.api.exceptions.ServiceNotAvailableException;

/**
 * Abstract class containing helper methods to determine the current WCM context.
 * This class can not be used directly. When in portal code please use PortalWCMContextHelper,
 * when in portlet code please use PortletWCMContextHelper.
 */
public abstract class WCMContextHelper {
    private final ContentMappingInfoHome contentMappingInfoHome;
    private final WebContentService webContentService;

    /**
     * Initializes the WCMContextHelper.
     *
     * @throws NullPointerException
     */
    public WCMContextHelper() throws NullPointerException {
        final Context ctx = new InitialContext();
        contentMappingInfoHome = (ContentMappingInfoHome) ctx.lookup(ContentMappingInfoHome.JNDI_NAME);
        webContentService = (WebContentService) ctx.lookup("portal:service/wcm/WebContentService");
    }

    /**
     * Gets the default content mapping of the currently selected page.
     *
     * @param stateManagerService StateManagerService, entry point to the portal state API
     * @param state the current portal state
     *
     * @return String representation of the content path that is mapped to the page.
     *
     * @throws NullPointerException
     * @throws IllegalArgumentException
     * @throws IllegalStateException
     * @throws OperationFailedException
     * @throws DocumentIdCreationException
     * @throws IllegalDocumentTypeException
     * @throws DocumentRetrievalException
     */
    protected String getContentMapping(final StateManagerService stateManagerService, final State state) throws Exception {
        final ObjectID currentPage = getCurrentPageID(stateManagerService, state);
        final ContentMappingInfo cmInfo = contentMappingInfoHome.getContentMappingInfo(currentPage);
        final ContentMapping cm = cmInfo.getDefaultContentMapping();
        String contentPath = null;
        if (cm != null) {
            final String contentID = cm.getContentID();
            final Workspace workspace = webContentService.getRepository().getWorkspace();
            final DocumentId docID = workspace.createDocumentId(contentID);
            contentPath = workspace.getPathById(docID, true, true);
        }
        return contentPath;
    }

    /**
     * Gets the ID of the currently selected page.
     *
     */
    public ObjectID getCurrentPageID(StateManagerService stateManagerService, final State state) throws Exception {
        final StateHolder stateHolder = stateManagerService.getStateHolder(state);
        final ObjectID currentPageID = stateHolder.getCurrentPageID();
        final ContentMappingInfo currentCMInfo = currentCMInfoHome.getContentMappingInfo(currentPageID);
        final ContentMapping currentCM = currentCMInfo.getDefaultContentMapping();
        final String currentContentPath = null;
        if (currentCM != null) {
            final String currentContentID = currentCM.getContentID();
            final Workspace workspace = webContentService.getRepository().getWorkspace();
            final DocumentId docID = workspace.createDocumentId(contentID);
            contentPath = workspace.getPathById(docID, true, true);
        }
        return contentPath;
    }
}
protected ObjectID getCurrentPageID(final StateManagerService stateManagerService, final StateHolder state) throws UnknownAccessorTypeException, CannotInstantiateAccessorException, InvalidSelectionNodeIdException {
    ObjectID result = null;
    final SelectionAccessorFactory selectionAccFct = stateManagerService.getAccessorFactory(SelectionAccessorFactory.class);
    final SelectionAccessor selectionAcc = selectionAccFct.getSelectionAccessor(state);
    try {
        result = selectionAcc.getSelection();
    } finally {
        selectionAcc.dispose();
    }
    return result;
}

/**
 * Assembles a proper content path that resembles 'library/site/sitearea/content'. The method
 * takes care of placing path separators where necessary. The returned content path does not
 * contain path separators as first or as last character.
 *
 * @param contentMapping
 * The content mapping of the current page. Must not be <code>null</code>.
 * @param pathInfo
 * The value of the path-info public render parameter for the current page. Must not be
 * <code>null</code>.
 * @return A fully-qualified content path.
 */
protected String assembleContentPath(final String contentMapping, final String[] pathInfo) {
    final StringBuilder result = new StringBuilder();
    // add the context mapping of the page w/o trailing forward slash
    if (contentMapping.charAt(contentMapping.length() - 1) == '/') {
        result.append(contentMapping, 0, contentMapping.length() - 1);
    } else {
        result.append(contentMapping);
    }
    // add all parts of path-info separated by a forward slashes
    for (final String pathInfoFragment : pathInfo) {
        if (pathInfoFragment != null && pathInfoFragment.length() > 0) {
            // add leading forward slash before each fragment
            result.append('/');
            // add the path-info fragment
            result.append(pathInfoFragment);
        }
    } return result.toString();
}

REST services for Web Content Manager

Application developers can use Representational State Transfer (REST) services to
work with Web Content Manager. The REST services for Web Content Manager
provide authoring access to content items and elements. The service follows the
Atom Publication Protocol, and atom feeds, and entries are accessible in XML
(application/atom+xml) and JSON (application/json) format.
Getting started with the REST service for Web Content Manager

Before getting started with the REST service for Web Content Manager you should become familiar with how it works and how to use it.

The REST service for Web Content Manager is a collection of Web services that are compliant with the Atom Publishing Protocol. They provide access to Web content, including versions and workflow states, via HTTP. The service is designed according to the REST (REpresentational State Transfer) architectural style.

REST services make it easy to build interactive content which can be modified directly by your site users. Responsive, integrated editing tools can be created by embedding HTML and JavaScript in web content components which bind to the REST service to display or update content asynchronously. (AJAX)

HTTP makes integration with remote clients easier than with a traditional API. Web Content Manager functions can be exposed to remote systems without adding additional server side components, such as JSP, to access Java APIs. HTTP allows these services to work seamlessly with your infrastructure including firewall, proxy servers and caches.

Note: Any examples in this section that contain incomplete XML, or XML without namespace declarations, use the following declarations:
- xmlns:atom="http://www.w3.org/2005/Atom"
- xmlns:app="http://www.w3.org/2007/app"

Service entry points

The URLs which comprise the REST service can change from release to release, or even with minor updates. Therefore it is recommended to never bookmark, or generate a URL unless it is for a defined entry point.

Atom publishing protocol service document

/wps/mycontenthandler/model/service

This service document includes the entry points for all portal REST services. When browsing for content, you should first retrieve the service document. The AtomPub service document describes the top-level collections in an APP service. These collections represent libraries and other types of content accessible through the service.

Web content queries can be stored in certain collections. This allows administrators to limit the scope and structure of queries, and bind them to specific URLs, which all authenticated users can access to retrieve the results as an atom feed.

POC Service

If a specific content item is known, it can be accessed directly through the POC service, or the POC service can be used to lookup an appropriate URL for the content. The identity of a piece of content in the REST service is represented by its POC URI. The POC URI can be found in the ID element of the Atom Entry documents which represents the content item.
Queries
While queries can be stored within the REST service, they can also be executed directly through a single location:
/wcmrest/query

This flexibility will be subject to security controls to prevent users from inadvertently overloading production servers with unnecessarily complex queries.

REST Service Access Levels
In order to use the REST service for Web Content Manager a client user be assigned the "user" role or higher in the WCM REST SERVICE virtual resource. All authenticated users are assigned the "user" role by default.

An administrator can edit the WCM REST SERVICE virtual resource by going to Administration > Resource Permissions > Virtual Resources.

Table 176, REST user roles

<table>
<thead>
<tr>
<th>Header</th>
<th>Users assigned the &quot;user&quot; role can:</th>
</tr>
</thead>
<tbody>
<tr>
<td>User</td>
<td>work with web content items and execute defined queries.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Editor</th>
<th>Users assigned the &quot;editor&quot; role can:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Editor</td>
<td>work with web content items and execute defined queries.</td>
</tr>
<tr>
<td>Editor</td>
<td>execute custom queries through the following path: /wcmrest/query</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Manager</th>
<th>Users assigned the &quot;manager&quot; role can:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manager</td>
<td>work with web content items and execute defined queries.</td>
</tr>
<tr>
<td>Manager</td>
<td>execute custom queries through the following path: /wcmrest/query</td>
</tr>
<tr>
<td>Manager</td>
<td>create, read, update and delete defined queries.</td>
</tr>
</tbody>
</table>

REST Query services for web content
The REST services for Web Content Manager come with a defined set of query parameters. You can also define your own query parameters in a white list. You can also pre-define a query to perform more complex searches, and control the allowable filters on these searches using a "white list".

Defined query Service
Defined queries are stored queries that can be run and updated as required.

The defined queries feature allows administrators to define a query in XML format through the REST service, and bind that query to a URI. Once this query has been defined, a user can execute it and obtain the results by issuing a GET request to the bound URI.
Create

A defined query is created by sending a POST request to the following URI:
/DefinedQueryComponent/

For example:

HTTP/1.1 POST
http://host:port/wps/mycontenthandler/wcmrest/DefinedQueryComponent
Content-Type: application/atom+xml
<atom:entry xmlns:atom="http://www.w3.org/2005/Atom" xmlns:wcm="wcm/namespace">
  <atom:title>defined query title-1712115665</atom:title>
  <wcm:name>query-name</wcm:name>
  <wcm:description>defined query description</wcm:description>
</atom:entry>

201 Created

The URI that is bound to this query has the following format:
/definedquery/component-name

For example, the URI used to obtain the results of the previous query example is:
/definedquery/query-name

The response to a create operation will contain a link relation "query-results" specifying this URI.

Note: The only way to update this URI is by modifying the name of the component.

Read

To retrieve a list of defined queries, you send a GET request to the following URI:
/DefinedQueryComponent/

For example:

HTTP/1.1 GET
http://host:port/wps/mycontenthandler/wcmrest/DefinedQueryComponent
Accept-Type: application/atom+xml

<atom:feed xmlns:atom="http://www.w3.org/2005/Atom" xmlns:wcm="wcm/namespace">
  <atom:title>Custom Queries</atom:title>
  <atom:entry>
    <atom:id>wcmrest:a5d4f72f-a7b7-4576-a7d3-5a4e15c66f01</atom:id>
    <atom:title>Query 1</atom:title>
    <atom:link atom:rel="edit-media" atom:type="application/vnd.ibm.wcm+xml" atom:href="/wps/mycontenthandler/!ut/p/wcmrest/DefinedQueryComponent/a5d4f72f-a7b7-4576-a7d3-5a4e15c66f01"/>
    <atom:link atom:rel="edit" atom:href="/wps/mycontenthandler/!ut/p/wcmrest/DefinedQueryComponent/a5d4f72f-a7b7-4576-a7d3-5a4e15c66f01"/>
  </atom:entry>
  <atom:entry>
    <atom:id>wcmrest:6276ff18-f370-45eb-89c3-053d335aba88</atom:id>
    <atom:title>defined query title-1280236937</atom:title>
  </atom:entry>
</atom:feed>
To retrieve the XML of a specific defined query, you send a GET request to the following URI:
/DefinedQueryComponent/item-uuid

For example:
HTTP/1.1 GET
Accept-Type: application/atom+xml
200 OK

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<definedQuery restrictParameters="true" page="1" pageSize="10" depth="DESCENDANTS">
  <select>
    <typeEquals>
      <type>Content</type>
    </typeEquals>
    <nameLike>
      <name>article%</name>
    </nameLike>
    <titleLike>
      <title>product%</title>
    </titleLike>
  </select>
  <allowParameters>
    <parameter>lastmodifiedbefore</parameter>
    <parameter>workflowid</parameter>
    <parameter>createdbefore</parameter>
    <parameter>authoringtemplateid</parameter>
  </allowParameters>
</definedQuery>

To retrieve the raw data of a specific defined query, send a GET request to the edit-media link relation:
/DefinedQueryComponent/item-id

For example:
HTTP/1.1 GET
http://host:port/wps/mycontenthandler/wcmrest/DefinedQueryComponent/items-id
Accept-Type: application/vnd.ibm.wcm+xml
200 OK

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<definedQuery restrictParameters="true" page="1" pageSize="10" depth="DESCENDANTS">
  <select>
    <typeEquals>
      <type>Content</type>
    </typeEquals>
    <nameLike>
      <name>article%</name>
    </nameLike>
    <titleLike>
      <title>product%</title>
    </titleLike>
  </select>
  <allowParameters>
    <parameter>lastmodifiedbefore</parameter>
    <parameter>workflowid</parameter>
    <parameter>createdbefore</parameter>
    <parameter>authoringtemplateid</parameter>
  </allowParameters>
</definedQuery>
**Update**

To update the meta-data of a query, issue a PUT request containing the new specification to the following URI:

```
/DefinedQueryComponent/item-uuid
```

For example:

```
HTTP/1.1 PUT
http://host:port/wps/mycontenthandler/wcmrest/DefinedQueryComponent/item-uuid
Content-Type: application/atom+xml
{... atom data ...}
```

200 OK

To update the raw data of a query that specifies the parameters used to conduct the query, issue a PUT request containing the new specification to the following URI:

```
/DefinedQueryComponent/item-uuid
```

For example:

```
HTTP/1.1 PUT
Content-Type: application/vnd.ibm.wcm+xml
{... xml data ...}
```

200 OK

**Delete**

To delete a defined query, issue a DELETE request to the following URI:

```
/DefinedQueryComponent/item-id
```

For example:

```
HTTP/1.1 DELETE
http://host:port/wps/mycontenthandler/wcmrest/definedquery/6276ff18-f370-45eb-89c3-053d335aba88
```

200 OK

**White List**

The white list is a list of additional query parameters that can be used to refine the scope of the defined query. These are the only parameters that will have any effect when appended to the bound URI.

For example, if the white list includes a parameter of name, the following request will return the results of the defined query which have the name "hello world":

```
HTTP/1.1 GET
http://host:port/wps/mycontenthandler/wcmrest/query-name?name=hello+world
```

200 OK
**Query parameters**

The following parameters can be used with queries.

*Table 177. Query parameters*

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>approver</td>
<td>This parameter is used to query items with a specific approver. A user ID must be specified when using this parameter. For example: /query?approver=userid</td>
</tr>
<tr>
<td>authoringtemplateid</td>
<td>This parameter is used to query items with a specific authoring template ID. For example: /query?authoringtemplateid=wcmrest:18cfc80c-a490-4d75-9057-fed3db89de53</td>
</tr>
<tr>
<td>author</td>
<td>This parameter is used to query items with a specific author. A user UID must be specified when using this parameter. For example: /query?author=uid=WCMUT_Editor_A,o=defaultWIMFileBasedRealm</td>
</tr>
<tr>
<td>categoryid</td>
<td>This parameter is used to query items with a specific category ID. For example: query?categoryid=wcmrest:18cfc80c-a490-4d75-9057-fed3db89de53</td>
</tr>
<tr>
<td>createdafter</td>
<td>This parameter is used to query items created after a specific date and time. For example: /query?createdafter=2011-01-11T11:43:29.0150Z</td>
</tr>
<tr>
<td>createdbefore</td>
<td>This parameter is used to query items created before a specific date and time. For example: /query?createdbefore=2011-01-11T11:43:29.0150Z</td>
</tr>
<tr>
<td>creator</td>
<td>This parameter is used to query items with a specific creator. A user UID must be specified when using this parameter. For example: /query?creator=uid=WCMUT_Editor_A,o=defaultWIMFileBasedRealm</td>
</tr>
<tr>
<td>dateformat</td>
<td>This parameter is used to define the date format of query parameters. For example: /query?dateformat=mm-dd-yyyy&amp;createdbefore=12-31-2011 If a date format not specified, then the default format yyyy-MM-dd'T'HH:mm:ssz is used.</td>
</tr>
</tbody>
</table>
### Table 177. Query parameters (continued)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>depth</td>
<td>This parameter is used with the parentid and is used to define whether to search for all descendants of a parent, or just the immediate children of a parent item. For example, to query only the immediate children of an item, you add &amp;depth=CHILDREN to the query: <code>/query?parentid=wcmrest:18cfc80c-a490-4d75-9057-fed3db89de53&amp;depth=CHILDREN</code> To query all descendants of an item, you add &amp;depth=DESCENDANTS to the query: <code>/query?parentid=wcmrest:18cfc80c-a490-4d75-9057-fed3db89de53&amp;depth=DESCENDANTS</code></td>
</tr>
<tr>
<td>expireafter</td>
<td>This parameter is used to query items that have expired after a specific date and time. For example: <code>/query?expireafter=2011-01-11T11:43:29.0150Z</code></td>
</tr>
<tr>
<td>expirebefore</td>
<td>This parameter is used to query items that were expired before a specific date and time. For example: <code>/query?expirebefore=2011-01-11T11:43:29.0150Z</code></td>
</tr>
<tr>
<td>id</td>
<td>This parameter is used to query an item with a specific ID. For example: <code>/query?id=wcmrest:18cfc80c-a490-4d75-9057-fed3db89de53</code></td>
</tr>
<tr>
<td>keyword</td>
<td>This parameter is used to query items that are profiled with a specific keyword. For example: <code>/query?keyword=keywordValue</code></td>
</tr>
<tr>
<td>lastmodifiedafter</td>
<td>This parameter is used to query items that were last modified after a specific date and time. For example: <code>/query?lastmodifiedafter=2011-01-11T11:43:29.0150Z</code></td>
</tr>
<tr>
<td>lastmodifiedbefore</td>
<td>This parameter is used to query items that were last modified before a specific date and time. For example: <code>/query?lastmodifiedbefore=2011-01-11T11:43:29.0150Z</code></td>
</tr>
<tr>
<td>lastmodifier</td>
<td>This parameter is used to query items that were last modified by a specific user. A user UID must be specified when using this parameter. For example: <code>/query?lastmodifier=uid=WCMUT_Editor_A,o=defaultWIMFileBasedRealm</code></td>
</tr>
<tr>
<td>libraryid</td>
<td>This parameter is used to query items that are stored in a specific library. For example: <code>/query?libraryid=uid=WCMUT_Editor_A,o=defaultWIMFileBasedRealm</code></td>
</tr>
</tbody>
</table>
### Table 177. Query parameters (continued)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>namelike</td>
<td>This parameter is used as a wildcard query for items with names like the specified namelike parameter. For example: /query?namelike=nameApproxValue%</td>
</tr>
<tr>
<td>name</td>
<td>This parameter is used to query an item with a specific name. For example: /query?name=nameValue</td>
</tr>
<tr>
<td>owner</td>
<td>This parameter is used to query items with a specific owner. A user UID must be specified when using this parameter. For example: /query?owner=uid=WCMUT_Editor_A,o=defaultWIMFileBasedRealm</td>
</tr>
<tr>
<td>pagesize</td>
<td>This parameter is used to restrict the number of items returned by a query to a set number. It can be used with the page parameter to return specific pages of results. For example, to restrict the number of queries to be returned to 5: /query?type=PresentationTemplate&amp;pagesize=5</td>
</tr>
<tr>
<td>page</td>
<td>This parameter is used with the pagesize parameter to define what set of results to display. For example, if pagesize is set to 5, and the page parameter is set to 2, then only results 6 to 10 will be displayed. For example: /query?type=PresentationTemplate&amp;pagesize=5&amp;page=2</td>
</tr>
<tr>
<td>parentid</td>
<td>This parameter is used to query items that are the children of a specific parent item. For example: /query?parentid=wcmrest:18cfc80c-a490-4d75-9057-fed3db89de53</td>
</tr>
<tr>
<td></td>
<td>You can use the depth parameter to define whether to search for all descendants of a parent, or just the immediate children of a parent item.</td>
</tr>
<tr>
<td>projectid</td>
<td>This parameter is used to query items that are linked to a specific project. For example: /query?projectid=wcmrest:18cfc80c-a490-4d75-9057-fed3db89de53</td>
</tr>
</tbody>
</table>
### Table 177. Query parameters (continued)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Details</th>
</tr>
</thead>
</table>
| projectstate | This parameter is used to query items that are linked to a project with a specific state. The following values can be used with this parameter:  
  - ACTIVE  
  - SYNDICATING  
  - PENDING  
  - PUBLISHING  
  - PUBLISHED  
  - PUBLISHED_FAILED  
  
  For example, to query items linked to projects with a state of "active" you would use the following query:  
  /query?projectstate=ACTIVE |
| publishafter | This parameter is used to query items that have been published after a specific date and time.  
  
  For example:  
| publishbefore | This parameter is used to query items that were published before a specific date and time.  
  
  For example:  
| sort | The sort parameter is appended to queries to determine how query results are sorted. The following values can be used with the sort parameter:  
  - author  
  - created  
  - modified  
  - name  
  - title  
  - parents  
  
  The values _ascending or _descending are appended to the query to determine sort order.  
  
  For example, to sort a presentation template query in ascending order of creation, you would use:  
  /query?type=PresentationTemplate&sort=created_ascending  
  
  To sort a presentation template query in descending order of creation, you would use:  
  /query?type=PresentationTemplate&sort=created_descending  
  
  If _ascending or _descending are not specified, the results as displayed in ascending order. |
Table 177. Query parameters (continued)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Details</th>
</tr>
</thead>
</table>
| state       | This parameter is used to query items that are in a specific state. The following values can be used with this parameter:  
  • DRAFT  
  • PUBLISHED  
  • EXPIRED  
  For example:  
  /query?state=PUBLISHED |
| titlelike   | This parameter is used as a wildcard query for items with titles like the specified titlelike parameter.  
  For example:  
  /query?titlelike=nameApproxValue% |
| title       | This parameter is used to query an item with a specific title.  
  For example:  
  /query?title=titleValue |
| type        | This parameter is used to query items of a specific item type.  
  For example, to query a list of components::  
  /query?type=LibraryHTMLComponent |
| workflowid  | This parameter is used to query items that use a specific workflow.  
  For example:  
  /query?workflowid=wcmrest:8d25860b-7a5c-4015-9cd5-bdcc60ce14bb |
| workflowstageid | This parameter is used to query items that are currently active within a specific workflow stage.  
  For example:  
  /query?workflowstageid=wcmrest:18cfc80c-a490-4d75-9057-fed3db9de53 |

Using multiple parameters

- Multiple instances of the same parameter type in a query can only be specified as 'OR' queries, with the following exceptions:
  
  **Only one value allowed**
  
  Only one instance of the following parameters can be used in a single query. If multiple instances are used, only the first instance is used by the query:
  
  - dateformat  
  - depth  
  - page  
  - pagesize  

  **"AND" queries allowed**
  
  The following queries can be used as "AND" queries:
  
  - createdafter  
  - createdbefore  
  - expireafter  
  - expirebefore
Working with web content items using REST

You can use the Web Content Manager REST Service to create, read, update and delete the following item types.

Using REST with Web Content Manager items

This topic provides an overview of the options available when working with Web Content Manager items, including the types of items that can be created and updated, and other important information related to the process of creating and updating web content items using REST.

Item types

The following item types can be created, read, updated and deleted using the Web Content Manager REST service:

- Site areas
- Content items
- Presentation templates
- Projects
- The following component types:
  - Text
  - Short text
  - Rich text
  - HTML
  - Number
  - Date and Time
  - Image
  - File resource
  - Style Sheet

The following element types can also be created, read, updated and deleted in site areas and content items using the Web Content Manager REST service:

- Text
- Short text
- Rich text
- HTML
- Number
- Date and Time
- Image
- File resource
Content representations

Atom entry documents

Each web content item has an associated atom entry document. The entry document provides access to the common metadata properties of the item, such as title and description, along with links to related items, and access control information.

The default media type of an entry document is application/atom+xml, however application/json representations can also be obtained. Entry documents can be retrieved by executing an HTTP GET to the entry resource. Entry resource URLs can be obtained from:

- the service document
- links in other entry documents
- links in feed documents
- using POC to resolve an ID

Example links to entry documents:

- <link rel="edit" href="/wps/mycontenthandler/lut/p/wcmrest/SiteArea/c6b00ee6-d628-4cbd-9e65-15c90f2093a6"/>
- <link rel="parent" href="/wps/mycontenthandler/lut/p/wcmrest/item/ae6a3632-a1b5-456a-866e-e9baab84fe29"/>
- <link rel="library" href="/wps/mycontenthandler/lut/p/wcmrest/item/54a68ca2-c550-43b5-966f-b0b61247547"/>

Media resources

In addition to the entry document, most items will also have a media resource associated with them. The media resource exists to store the content of the item, for example, HTML or an image. The media resource location is found in the edit-media link of an items entry document. Media resource URLs support HTTP GET and PUT.

Example links to media resources:

- <link rel="edit-media" type="text/html" href="/wps/mycontenthandler/lut/p/wcmrest/PresentationTemplate/02da6e9d-20ca-4c54-ae4a-f114fa8e948"/>

Content Negotiation

When a resource supports multiple representations with different content types, clients can use content negotiation to request a specific representation. For example, entry documents are available in application/atom+xml; type=entry and application/json. There are two ways a client can specify the media type that the client can accept:

HTTP Accept header

Accept: application/atom+xml

Request parameter mime-type

GET /wps/mycontenthandler/lut/p/wcmrest/SiteArea/c6b00ee6-d628-4cbd-9e65-15c90f2093a6?mime-type=application%2Fjson HTTP/1.0

Example entry

This is an example in application/atom+xml format:

```
<entry xmlns="http://www.w3.org/2005/Atom" xmlns:wcm="wcm/namespace">
  <id>3aabfe14-cc9e-4eb5-ad06-d4fc8fd2f1df</id>
  <title>SampleHTMLComponent</title>
  <link rel="edit" href="/wps/mycontenthandler/lut/p/wcmrest/LibraryHTMLComponent/3aabfe14-cc9e-4eb5-ad06-d4fc8fd2f1df"/>
  <link rel="edit-media" type="text/html" href="/wps/mycontenthandler/lut/p/wcmrest/LibraryHTMLComponent/3aabfe14-cc9e-4eb5-ad06-d4fc8fd2f1df"/>
  <link rel="library" href="/wps/mycontenthandler/lut/p/wcmrest/item/c98d11e1-7f2a-480e-9aac-40eb1949cda"/>
  <link rel="create-draft" href="/wps/mycontenthandler/lut/p/wcmrest/item/3aabfe14-cc9e-4eb5-ad06-d4fc8fd2f1df/create-draft"/>
  <link rel="change-to-draft" href="/wps/mycontenthandler/lut/p/wcmrest/item/3aabfe14-cc9e-4eb5-ad06-d4fc8fd2f1df/change-to-draft"/>
</entry>
```
Using REST with components
You can use the Web Content Manager REST service to create, read, update and delete some types of components.

The following component types can be used with the Web Content Manager REST service.

Table 178. Component types

<table>
<thead>
<tr>
<th>Component</th>
<th>API type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Text component</td>
<td>LibraryTextComponent</td>
</tr>
<tr>
<td>Short text component</td>
<td>LibraryShortTextComponent</td>
</tr>
<tr>
<td>Rich text component</td>
<td>LibraryRichTextComponent</td>
</tr>
<tr>
<td>HTML Component</td>
<td>LibraryHTMLComponent</td>
</tr>
<tr>
<td>Number component</td>
<td>LibraryNumericComponent</td>
</tr>
<tr>
<td>Date and time component</td>
<td>LibraryDateComponent</td>
</tr>
<tr>
<td>Image component</td>
<td>LibraryImageComponent</td>
</tr>
<tr>
<td>File resource component</td>
<td>LibraryFileComponent</td>
</tr>
<tr>
<td>Stylesheet component</td>
<td>LibraryStyleSheetComponent</td>
</tr>
</tbody>
</table>

Create

A component can be created by sending a POST request to the following URI with an atom entry representing the component:

/<library-component-api-type>

For example:

POST /wps/mycontenthandler/wcmrest/LibraryNumericComponent HTTP/1.0
Content-Type: application/atom+xml

<entry xmlns="http://www.w3.org/2005/Atom" xmlns:wcm="wcm/namespace"

<title>SampleNumericComponentTitle</title>
<link rel="library" href="/wps/mycontenthandler/lut/p/wcmrest/item/c98d11e1-7f2a-480e-9a4c-40eb1949cbda"/>
<wcm:name>SampleNumericComponentName</wcm:name>
<wcm:description>SampleNumericComponentDescription</wcm:description>
</entry>

HTTP/1.0 201 Created
Content-type: application/atom+xml; type=entry
Content-location: /wps/mycontenthandler/lut/p/wcmrest/LibraryNumericComponent/0d678334-69ae-4d3a-a525-91bb551e5a18
Update

A component can be updated by sending a PUT request to the following URI with an atom entry including the fields of the item that need to be changed.

For example:

PUT /wps/mycontenthandler/wcmrest/LibraryNumericComponent/c98d11e1-7f2a-480e-9aac-40eb1949cbda HTTP/1.0
Content-type: application/atom+xml

HTTP/1.0 200 OK
Content-type: application/atom+xml; type=entry
Read

A component can be read by sending a GET request to the following URI:

/library-component-api-type/</itemuid>

For example:

GET /wps/mycontenthandler/wcmrest/LibraryNumericComponent/0d678334-69ae-4d3a-a525-91bb551e5a18 HTTP/1.0

HTTP/1.0 200 OK
Content-type: application/atom+xml; type=entry

<entry xmlns="http://www.w3.org/2005/Atom" xmlns:wcm="wcm/namespace">
  <id>0d678334-69ae-4d3a-a525-91bb551e5a18</id>
  <title>SampleNumericComponentTitleUpdated</title>
  <link rel="edit" href="/wps/mycontenthandler/!ut/p/wcmrest/LibraryNumericComponent/0d678334-69ae-4d3a-a525-91bb551e5a18"/>
  <link rel="edit-media" type="text/plain" href="/wps/mycontenthandler/!ut/p/wcmrest/LibraryNumericComponent/0d678334-69ae-4d3a-a525-91bb551e5a18"/>
  <link rel="library" href="/wps/mycontenthandler/!ut/p/wcmrest/item/c98d11e1-7f2a-480e-9acc-40eb1994cbda"/>
  <link rel="create-draft" href="/wps/mycontenthandler/!ut/p/wcmrest/item/0d678334-69ae-4d3a-a525-91bb551e5a18/create-draft"/>
  <link rel="change-to-draft" href="/wps/mycontenthandler/!ut/p/wcmrest/item/0d678334-69ae-4d3a-a525-91bb551e5a18/change-to-draft"/>
  <link rel="versions" href="/wps/mycontenthandler/!ut/p/wcmrest/item/0d678334-69ae-4d3a-a525-91bb551e5a18/versions"/>
  <updated>2011-05-30T04:38:49.522Z</updated>
  <author>
    <wcm:distinguishedName>uid=wpsadmin,o=defaultWIMFileBasedRealm</wcm:distinguishedName>
    <uri>/wps/mycontenthandler/!ut/p/digest16Vkh5U17SN7dEgVHm_g/um/users/profiles/Z9eAeH1C2JG561RC6JM47H9E4MMG6PH06JM4C5JD0JMOC6EEJS464JDOG3I
    <name>wpsadmin</name>
  </author>
  </wcm:owner>
  </wcm:name>SampleNumericComponentNameUpdated</wcm:name>
  <wcm:description>SampleNumericComponentDescriptionUpdated</wcm:description>
  <wcm:type>LibraryNumericComponent</wcm:type>
  <wcm:state>PUBLISHED</wcm:state>
</entry>

Delete

A component can be deleted by sending a DELETE request to the following URI:

/library-component-api-type/</itemuid>

For example:

DELETE HTTP/1.1 DELETE
http://host:port/wps/mycontenthandler/wcmrest/LibraryNumericComponent/</itemuid>

Response
Status Code :200
Status Message : OK
Working with raw data

The content of a component is accessed from the location specified in the HREF attribute of the edit-media link. The link will also contain a TYPE attribute which contains the accepted media type for the content. For example:

```
<link rel="edit-media" type="text/plain" href="/wps/mycontenthandler/!ut/p/wcmrest/LibraryTextComponent/0d678334-69ae-4d3a-a525-91bb551e5a18"/>
```

The media types for supported components are:

<table>
<thead>
<tr>
<th>Media type</th>
<th>Component types</th>
</tr>
</thead>
<tbody>
<tr>
<td>text/plain</td>
<td>• Text</td>
</tr>
<tr>
<td>application/vnd.ibm.wcm+xml</td>
<td>• Number</td>
</tr>
<tr>
<td></td>
<td>• Date and Time</td>
</tr>
<tr>
<td></td>
<td>• Image</td>
</tr>
<tr>
<td></td>
<td>• Stylesheet</td>
</tr>
<tr>
<td>text/html</td>
<td>• HTML</td>
</tr>
<tr>
<td></td>
<td>• Rich text</td>
</tr>
<tr>
<td></td>
<td>• Presentation template</td>
</tr>
<tr>
<td>text/css</td>
<td>• Stylesheet</td>
</tr>
<tr>
<td>image/*</td>
<td>• Image</td>
</tr>
<tr>
<td>application/octet-stream</td>
<td>• File</td>
</tr>
</tbody>
</table>

To update content of a library component PUT content in an accepted media type to the edit-media URL. For example:

```
PUT /wps/mycontenthandler/!ut/p/wcmrest/LibraryTextComponent/0d678334-69ae-4d3a-a525-91bb551e5a18 HTTP/1.0
Content-type: text/plain

This is some text to add to the component.
```

HTTP/1.0 200 OK

To retrieve content from a library component GET content from the edit-media URL. For example:

```
GET /wps/mycontenthandler/!ut/p/wcmrest/LibraryTextComponent/0d678334-69ae-4d3a-a525-91bb551e5a18 HTTP/1.0
Accept: text/plain

HTTP/1.0 200 OK
Content-type: text/plain

This is some text to add to the component.

An alternative to specifying the media type in the HTTP accept header is to use the request parameter mime-type. You must URL encode the value. For example:

```
GET /wps/mycontenthandler/!ut/p/wcmrest/LibraryTextComponent/0d678334-69ae-4d3a-a525-91bb551e5a18?mime-type=text%2Fplain HTTP/1.0

HTTP/1.0 200 OK
```
Using REST with elements
You can use the Web Content Manager REST service to create, read, update and delete some types of elements stored in site areas and content items.

The following element types can be used with the Web Content Manager REST service.

<table>
<thead>
<tr>
<th>Element</th>
<th>API type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Text element</td>
<td>TextComponent</td>
</tr>
<tr>
<td>Short text element</td>
<td>ShortTextComponent</td>
</tr>
<tr>
<td>Rich text element</td>
<td>RichTextComponent</td>
</tr>
<tr>
<td>HTML element</td>
<td>HTMLComponent</td>
</tr>
<tr>
<td>Number element</td>
<td>NumericComponent</td>
</tr>
<tr>
<td>Date and time element</td>
<td>DateComponent</td>
</tr>
<tr>
<td>Image element</td>
<td>ImageComponent</td>
</tr>
<tr>
<td>File resource element</td>
<td>FileComponent</td>
</tr>
<tr>
<td>Style sheet element</td>
<td>StyleSheetComponent</td>
</tr>
</tbody>
</table>

Create

An element can be created by sending a POST request to the following URI with an atom entry representing the title of the element:

```
/[[Content|SiteArea]/<parent-uuid>/elements
```

Note: The type of the element to be created must be specified in the type field of the entry that is posted.

For example:

```
POST /wps/mycontenthandler/wcmrest/SiteArea/c6b00ee6-d628-4cbd-9e65-15c90f2093a6/elements HTTP/1.0
Content-type: application/atom+xml

<entry xmlns="http://www.w3.org/2005/Atom" xmlns:wcm="wcm/namespace">
  <title>numericComponentTitle</title>
  <wcm:name>numericComponentName</wcm:name>
  <wcm:type>NumericComponent</wcm:type>
</entry>
```

HTTP/1.0 201 Created
Content-type: application/atom+xml; type=entry
Content-location: /wps/mycontenthandler/!ut/p/wcmrest/SiteArea/c6b00ee6-d628-4cbd-9e65-15c90f2093a6/elements/numericComponentName

```
<entry xmlns="http://www.w3.org/2005/Atom" xmlns:wcm="wcm/namespace">
  <link rel="edit-media" type="text/plain" href="/wps/mycontenthandler/!ut/p/wcmrest/SiteArea/c6b00ee6-d628-4cbd-9e65-15c90f2093a6/elements/numericComponentName"/>
  <wcm:name>numericComponentName</wcm:name>
  <wcm:type>NumericComponent</wcm:type>
</entry>
```
Update

An element can be updated by sending a PUT request to the following URI with an atom entry including the name and title of the element.

/([Content|SiteArea]/<parentuuid>/elements/<element-name-encoded>

For example:

PUT /wps/mycontenthandler/wcmrest/SiteArea/c6b00ee6-d628-4cbe-96e5-15c90f2093a6/elements/numericComponentName HTTP/1.0
Content-type: application/atom+xml

<atom:entry xmlns:atom="http://www.w3.org/2005/Atom" xmlns:wcm="wcm/namespace">
  <title>numericComponentTitleUpdated</title>
  <wcm:name>numericComponentNameUpdated</wcm:name>
</atom:entry>

HTTP/1.0 200 OK
Content-type: application/atom+xml; type=entry

<entry xmlns="http://www.w3.org/2005/Atom" xmlns:wcm="wcm/namespace">
  <title>numericComponentTitleUpdated</title>
  <link rel="edit-media" type="text/plain" href="/wps/mycontenthandler/!ut/p/wcmrest/SiteArea/c6b00ee6-d628-4cbe-96e5-15c90f2093a6/elements/numericComponentName"/>
  <link rel="edit" href="/wps/mycontenthandler/!ut/p/wcmrest/SiteArea/c6b00ee6-d628-4cbe-96e5-15c90f2093a6/elements/numericComponentName"/>
  <wcm:name>numericComponentNameUpdated</wcm:name>
  <wcm:type>NumericComponent</wcm:type>
</entry>

Read

An element can be read by sending a GET request to the following URI:

/([Content|SiteArea]/<parentuuid>/elements/<element-name-encoded>

For example:

GET /wps/mycontenthandler/wcmrest/Content/c6b00ee6-d628-4cbe-96e5-15c90f2093a6/elements/numericComponentName HTTP/1.0

HTTP/1.0 200 OK
Content-type: application/atom+xml; type=entry

<entry xmlns="http://www.w3.org/2005/Atom" xmlns:wcm="wcm/namespace">
  <title>numericComponentTitleUpdated</title>
  <link rel="edit-media" type="application/vnd.ibm.wcm+xml" href="/wps/mycontenthandler/!ut/p/wcmrest/SiteArea/c6b00ee6-d628-4cbe-96e5-15c90f2093a6/elements/numericComponentName"/>
  <link rel="edit" href="/wps/mycontenthandler/!ut/p/wcmrest/SiteArea/c6b00ee6-d628-4cbe-96e5-15c90f2093a6/elements/numericComponentName"/>
  <wcm:name>numericComponentNameUpdated</wcm:name>
  <wcm:type>NumericComponent</wcm:type>
</entry>

Delete

An element can be deleted by sending a DELETE request to the following URI:

/([Content|SiteArea]/<parentuuid>/elements/<element-name-encoded>

For example:

DELETE HTTP/1.1 DELETE

Response

Status Code : 200
Status Message : OK
Working with raw data

The content of an element is accessed from the media resource specified in the HREF attribute of the edit-media link. The link will also contain a TYPE attribute which contains the accepted media type of the content. For example:

```html
<link rel="edit-media" type="text/html" href="/wps/mycontenthandler/ut/p/wcmrest/SiteArea/c6b00ee6-d628-4cbd-9e65-15c90f2093a6/elements/heading"/>
```

The media types for supported elements are:

Table 181. Media types for elements

<table>
<thead>
<tr>
<th>Media type</th>
<th>Component types</th>
</tr>
</thead>
<tbody>
<tr>
<td>text/plain</td>
<td>• Text</td>
</tr>
<tr>
<td>application/vnd.ibm.wcm+xml</td>
<td>• Number</td>
</tr>
<tr>
<td></td>
<td>• Date and Time</td>
</tr>
<tr>
<td></td>
<td>• Image</td>
</tr>
<tr>
<td></td>
<td>• Stylesheet</td>
</tr>
<tr>
<td>text/html</td>
<td>• HTML</td>
</tr>
<tr>
<td></td>
<td>• Rich text</td>
</tr>
<tr>
<td>text/css</td>
<td>• Stylesheet</td>
</tr>
<tr>
<td>image/*</td>
<td>• Image</td>
</tr>
<tr>
<td>application/octet-stream</td>
<td>• File</td>
</tr>
</tbody>
</table>

To update content of an element PUT content in an accepted media type to the edit-media URL. For example:

```
PUT /wps/mycontenthandler/ut/p/wcmrest/SiteArea/c6b00ee6-d628-4cbd-9e65-15c90f2093a6/elements/heading HTTP/1.0
Content-type: text/html

<h1>Heading Text</h1>
```

HTTP/1.0 200 OK

To retrieve content from a library component GET content from the edit-media URL. For example:

```
GET /wps/mycontenthandler/ut/p/wcmrest/SiteArea/c6b00ee6-d628-4cbd-9e65-15c90f2093a6/elements/heading HTTP/1.0
Accept: text/html

HTTP/1.0 200 OK
Content-type: text/plain

<h1>Heading Text</h1>
```

An alternative to specifying the media type in the HTTP accept header, is to use the request parameter mime-type. You must URL encode the value. For example:

```
GET /wps/mycontenthandler/ut/p/wcmrest/SiteArea/c6b00ee6-d628-4cbd-9e65-15c90f2093a6/elements/heading?mime-type=text%2Fhtml HTTP/1.0

HTTP/1.0 200 OK
Content-type: text/plain

<h1>Heading Text</h1>
```
Using REST with presentation templates
You can use the Web Content Manager REST service to create, read, update and delete presentation templates.

Create
A presentation template can be created by sending a POST request to the following URI with an atom entry representing the presentation template:
/PresenterTemplate
For example:
POST /wps/mycontenthandler/wcmrest/PresentationTemplate HTTP/1.0 POST
Content-Type : application/atom+xml

<entry xmlns="http://www.w3.org/2005/Atom" xmlns:wcm="wcm/namespace">
  <title>SamplePresentationTemplateTitle</title>
  <link rel="library" href="/wps/mycontenthandler/!ut/p/wcmrest/item/c98d11e1-7f2a-480e-9aac-40eb1949cbda"/>
  <wcm:name>SamplePresentationTemplateTitleName</wcm:name>
  <wcm:description>SamplePresentationTemplateDescription</wcm:description>
</entry>

HTTP/1.0 201 Created
Content-type: application/atom+xml; type=entry
Content-location: /wps/mycontenthandler/!ut/p/wcmrest/PresentationTemplate/02da6e9d-20ca-4c54-ae4a-f1114fa8e948

Update
A presentation template can be updated by sending a PUT request to the following URI with an atom entry including the fields on the item that need to be changed.
/PresenterTemplate/<itemuuid>
For example:
PUT /wps/mycontenthandler/wcmrest/PresentationTemplate/02da6e9d-20ca-4c54-ae4a-f1114fa8e948 HTTP/1.0
Content-Type: application/atom+xml

<entry xmlns="http://www.w3.org/2005/Atom" xmlns:wcm="wcm/namespace">
<title>SamplePresentationTemplateTitleUpdated</title>
<link rel="library" href="/wps/mycontenthandler/!ut/p/wcmrest/item/c98d11e1-7f2a-480e-9aac-40eb1949cbda"/>
<wcm:name>SamplePresentationTemplateTitleNameUpdated</wcm:name>
<wcm:description>SamplePresentationTemplateDescriptionUpdated</wcm:description>
</entry>
HTTP/1.0 200 OK
Content-type: application/atom+xml; type=entry

<entry xmlns="http://www.w3.org/2005/Atom" xmlns:wcm="wcm/namespace">
<id>02da6e9d-20ca-4c54-ae4a-f1114fa8e948</id>
<title>SamplePresentationTemplateTitleUpdated</title>
<link rel="edit" href="/wps/mycontenthandler/!ut/p/wcmrest/PresentationTemplate/02da6e9d-20ca-4c54-ae4a-f1114fa8e948"/>
<link rel="edit-media" type="text/html" href="/wps/mycontenthandler/!ut/p/wcmrest/PresentationTemplate/02da6e9d-20ca-4c54-ae4a-f1114fa8e948"/>
<link rel="library" href="/wps/mycontenthandler/!ut/p/wcmrest/item/c98d11e1-7f2a-480e-9aac-40eb1949cbda"/>
<link rel="create-draft" href="/wps/mycontenthandler/!ut/p/wcmrest/item/02da6e9d-20ca-4c54-ae4a-f1114fa8e948/create-draft"/>
<link rel="change-to-draft" href="/wps/mycontenthandler/!ut/p/wcmrest/item/02da6e9d-20ca-4c54-ae4a-f1114fa8e948/change-to-draft"/>
<link rel="versions" href="/wps/mycontenthandler/!ut/p/wcmrest/item/02da6e9d-20ca-4c54-ae4a-f1114fa8e948/versions"/>
<link rel="add-attachment" href="/wps/mycontenthandler/!ut/p/wcmrest/PresentationTemplate/02da6e9d-20ca-4c54-ae4a-f1114fa8e948/attachments"/>
<author>
<wcm:distinguishedName>uid=wpsadmin,o=defaultWIMFileBasedRealm</wcm:distinguishedName>
<uri>/wps/mycontenthandler/!ut/p/digest!*GVkh5Ul75Ln7DdEvHm_g/um/users/profiles/Z9eAeH1C2JG561RC6JM47H9E4MMG6PH06JMC45JD0JMC6BEEJS46A4DGD315</uri>
</author>
<wcm:owner>
<wcm:distinguishedName>uid=wpsadmin,o=defaultWIMFileBasedRealm</wcm:distinguishedName>
<uri>/wps/mycontenthandler/!ut/p/digest!*GVkh5Ul75Ln7DdEvHm_g/um/users/profiles/Z9eAeH1C2JG561RC6JM47H9E4MMG6PH06JMC45JD0JMC6BEEJS46A4DGD315</uri>
</wcm:owner>
<wcm:name>SamplePresentationTemplateTitleNameUpdated</wcm:name>
<wcm:description>SamplePresentationTemplateDescriptionUpdated</wcm:description>
<wcm:type>PresentationTemplate</wcm:type>
<wcm:state>PUBLISHED</wcm:state>
</entry>

Read

A presentation template can be read by sending a GET request to the following URI:
/PresentationTemplate/<itemuuid>

For example:
GET /wps/mycontenthandler/wcmrest/PresentationTemplate/02da6e9d-20ca-4c54-ae4a-f1114fa8e948 HTTP/1.0

HTTP/1.0 200 OK
Content-type: application/atom+xml; type=entry

<entry xmlns="http://www.w3.org/2005/Atom" xmlns:wcm="wcm/namespace">
<id>02da6e9d-20ca-4c54-ae4a-f1114fa8e948</id>
<title>SamplePresentationTemplateTitleUpdated</title>
<link rel="edit" href="/wps/mycontenthandler/!ut/p/wcmrest/PresentationTemplate/02da6e9d-20ca-4c54-ae4a-f1114fa8e948"/>
<link rel="edit-media" type="text/html" href="/wps/mycontenthandler/!ut/p/wcmrest/PresentationTemplate/02da6e9d-20ca-4c54-ae4a-f1114fa8e948"/>
<link rel="library" href="/wps/mycontenthandler/!ut/p/wcmrest/item/c98d11e1-7f2a-480e-9aac-40eb1949cbda"/>
<link rel="create-draft" href="/wps/mycontenthandler/!ut/p/wcmrest/item/02da6e9d-20ca-4c54-ae4a-f1114fa8e948/create-draft"/>
<link rel="change-to-draft" href="/wps/mycontenthandler/!ut/p/wcmrest/item/02da6e9d-20ca-4c54-ae4a-f1114fa8e948/change-to-draft"/>
<link rel="versions" href="/wps/mycontenthandler/!ut/p/wcmrest/item/02da6e9d-20ca-4c54-ae4a-f1114fa8e948/versions"/>
<link rel="add-attachment" href="/wps/mycontenthandler/!ut/p/wcmrest/PresentationTemplate/02da6e9d-20ca-4c54-ae4a-f1114fa8e948/attachments"/>
<author>
<wcm:distinguishedName>uid=wpsadmin,o=defaultWIMFileBasedRealm</wcm:distinguishedName>
<uri>/wps/mycontenthandler/!ut/p/digest!*GVkh5Ul75Ln7DdEvHm_g/um/users/profiles/Z9eAeH1C2JG561RC6JM47H9E4MMG6PH06JMC45JD0JMC6BEEJS46A4DGD315</uri>
</author>
...
Delete

A presentation template can be deleted by sending a DELETE request to the following URI:

/PresentationTemplate/<itemuuid>

For example:

DELETE

HTTP/1.1 DELETE
http://host:port/wps/mycontenthandler/wcmrest/PresentationTemplate/<itemuuid>

Response:
Status Code :200
Status Message : OK

Working with raw data

The content of a presentation template is accessed from the media resource specified in the HREF attribute of the edit-media link. The link will also contain a TYPE attribute which contains the accepted media type for presentation templates, which is text/html. For example:

```
<link rel="edit-media" type="text/html"
href="/wps/mycontenthandler/!ut/p/wcmrest/PresentationTemplate/02da6e9d-20ca-4c54-ae4a-f1114fa8e948"/>
```

To update content of a presentation template PUT content of type text/html type to the edit-media URL. For example:

```
PUT /wps/mycontenthandler/!ut/p/wcmrest/PresentationTemplate/02da6e9d-20ca-4c54-ae4a-f1114fa8e948 HTTP/1.0
Content-type: text/html

<html>
<body>
    [Component name="story"]
</body>
</html>
```

HTTP/1.0 200 OK

To retrieve content from a library component GET content from the edit-media URL. For example:

```
GET /wps/mycontenthandler/!ut/p/wcmrest/PresentationTemplate/02da6e9d-20ca-4c54-ae4a-f1114fa8e948 HTTP/1.0
Accept: text/html
```

HTTP/1.0 200 OK
Using REST with content items

You can use the Web Content Manager REST service to create, read, update and delete content items.

Create

A content item can be created by sending a POST request to the following URI with an atom entry representing the content item:

/Content

For example:

Post

HTTP/1.1 POST
http://host:port/wps/mycontenthandler/wcmrest/Content/
Content-Type: application/atom+xml

<atom:entry xmlns:atom="http://www.w3.org/2005/Atom" xmlns:wcm="wcm/namespace">
  <atom:name>Content Name</atom:name>
  <atom:title>Content Title</atom:title>
</atom:entry>

Response:
201 Created

Create from a skeleton

A "skeleton" representation of a content item created from a content template can be obtained to aid in the creation of content items. This can be obtained by issuing a GET request to the following URI. Once the skeleton is obtained and completed a POST request can be made using this data to create the item.

/ContentTemplate/template-uuid/new-content

For example:

HTTP/1.1 GET http://host:port/wps/mycontenthandler/wcmrest/ContentTemplate/b7b8b3fb-8fa1-4eb3-915e-ce7514f706f7/new-content

Response
200 OK

<?xml version="1.0" encoding="UTF-8"?>
  <id>wcmrest:6bab48c6-1f24-454e-9bab-ae1be4cf3a0a</id>
  <title lang="en"></title>
  <summary lang="en"></summary>
  <wcm:name></wcm:name>
  <wcm:type>Content</wcm:type>
  <updated>2012-01-31T03:28:08.118Z</updated>
</entry>
Update

A content item can be updated by sending a PUT request to the following URI with an atom entry including the fields on the item that need to be changed.

/Content/item-uuid

For example:

PUT
HTTP/1.1 PUT
http://host:port/wps/mycontenthandler/wcmrest/Content/abae799b-4cca-47ae-aad8-b3d8204deefb
Content-Type: application/atom+xml

Response:
200 OK

Read

A content item can be read by sending a GET request to the following URI:

/Content/item-uuid

For example:

GET
HTTP/1.1 GET
http://host:port/wps/mycontenthandler/wcmrest/Content/fa2bfd32-7b2f-4394-a5ab-2e150c5ed8aa
Delete

A content item can be deleted by sending a DELETE request to the following URI:
/Content/item-uuid

For example:

DELETE
HTTP/1.1 DELETE
http://host:port/wps/mycontenthandler/wcmrest/Content/fa2bfd32-7b2f-4394-a5ab-2e150c5ed8aa/

Response:
200 OK

Using REST with site areas
You can use the Web Content Manager REST service to create, read, update and delete site areas.

Create

A site area can be created by sending a POST request to the following URI with an atom entry representing the site area:
/SiteArea

- A library or parent link relation must be specified. This tells the REST service the location of the hierarchical item being created.
- An authoring template must be specified. This tells the REST service what authoring template to use when creating the item.
- Template mappings can also be specified.

For example:

POST /wps/mycontenthandler/wcmrest/SiteArea HTTP/1.0
Content-type : application/atom+xml

<entry xmlns="http://www.w3.org/2005/Atom" xmlns:wcm="wcm/namespace">
  <title>SampleSiteAreaTitle</title>
  <link rel="parent" href="/wps/mycontenthandler/!ut/p/wcmrest/item/ae6a3632-a1b5-456a-866e-e9baab84fe29"/>
  <wcm:name>SampleSiteAreaName</wcm:name>
  <wcm:description>SampleSiteAreaDescription</wcm:description>
</entry>

HTTP/1.0 201 Created
Content-type : application/atom+xml; type=entry
Content-location: /wps/mycontenthandler/!ut/p/wcmrest/SiteArea/18001a8c-2117-45d2-be1c-baea28a41769

<entry xmlns="http://www.w3.org/2005/Atom" xmlns:wcm="wcm/namespace">
  <id>18001a8c-2117-45d2-be1c-baea28a41769</id>
  <title>SampleSiteAreaTitle</title>
  <link rel="edit" href="/wps/mycontenthandler/!ut/p/wcmrest/SiteArea/18001a8c-2117-45d2-be1c-baea28a41769"/>
  <link rel="create-draft" href="/wps/mycontenthandler/!ut/p/wcmrest/item/18001a8c-2117-45d2-be1c-baea28a41769/create-draft"/>
  <link rel="change-to-draft" href="/wps/mycontenthandler/!ut/p/wcmrest/item/18001a8c-2117-45d2-be1c-baea28a41769/change-to-draft"/>
</entry>
Create from a skeleton

A "skeleton" representation of a site area created from a content template can be obtained to aid in the creation of content items. This can be obtained by issuing a GET request to the following URI. Once the skeleton is obtained and completed a POST request can be made using this data to create the item.

/SiteAreaTemplate/template-uuid/new-content

For example:

```xml
<?xml version="1.0" encoding="UTF-8"?>
  <id>wcmrest:38188c20-44e4-4447-8a54-91d47ecfcc13</id>
  <wcm:name></wcm:name>
  <wcm:type>SiteArea</wcm:type>
  <updated>2012-01-31T03:33:17.826Z</updated>
  <author>
    <wcm:distinguishedName>uid=wpsadmin,o=defaultWIMFileBasedRealm</wcm:distinguishedName>
    <uri>/wps/mycontenthandler/lut/p/digest!7K1PhYjxBw0jzCDqHCwg2w/um/users/profiles/Z9eAeHPCAJG963RD2MMG6P9O6MMG66BD6MM47IHP4MMS6M1DAJQ4C1BCAM1D653</uri>
    <name>wpsadmin</name>
  </author>
  <wcm:owner>
    <wcm:distinguishedName>uid=wpsadmin,o=defaultWIMFileBasedRealm</wcm:distinguishedName>
    <uri>/wps/mycontenthandler/lut/p/digest!7K1PhYjxBw0jzCDqHCwg2w/um/users/profiles/Z9eAeHPCAJG963RD2MMG6P9O6MMG66BD6MM47IHP4MMS6M1DAJQ4C1BCAM1D653</uri>
    <name>wpsadmin</name>
  </wcm:owner>
  <link rel="source" href="/wps/mycontenthandler/lut/p/digest!PQo5Yhy68oepWWcEz2sdda/wcmrest/SiteArea/72456e6a-1975-481f-a015-98e75f73c7af">Sitearea Template</link>
  <content type="application/vnd.ibm.wcm+xml">
    <siteArea xmlns="http://www.ibm.com/xmlns/wcm/8.0">
      <elements>
        <element name="summary">
          <title lang="en-US">Summary</title>
          <type>TextComponent</type>
          <data type="text/plain">[[CDATA[Text inside the element]]]&gt;\</data>
        </element>
        <element name="body">
          <title lang="en-US">Body</title>
          <type>HTMLComponent</type>
          <data type="text/html">[[CDATA[&lt;p&gt;
Lorem ipsum dolor sit amet, consectetur adipiscing elit. 
]]&gt;\</data>
        </element>
      </elements>
    </siteArea>
  </content>
</entry>
```
Update

A site area can be updated by sending a PUT request to the following URI with an atom entry including the fields on the item that need to be changed.

/SiteArea/item-uuid

For example:

PUT /wps/mycontenthandler/wcmrest/SiteArea/18001a8c-2117-45d2-be1c-baea28a41769 HTTP/1.0
Content-Type : application/atom+xml

<entry xmlns="http://www.w3.org/2005/Atom" xmlns:wcm="wcm/namespace">
  <title>SampleSiteAreaTitleUpdated</title>
  <wcm:name>SampleSiteAreaNameUpdated</wcm:name>
  <wcm:description>SampleSiteAreaDescriptionUpdated</wcm:description>
</entry>

HTTP/1.0 200 OK
Content-type: application/atom+xml; type=entry

<entry xmlns="http://www.w3.org/2005/Atom" xmlns:wcm="wcm/namespace">
  <id>18001a8c-2117-45d2-be1c-baea28a41769</id>
  <title>SampleSiteAreaTitleUpdated</title>
  <link rel="edit" href="/wps/mycontenthandler/!ut/p/wcmrest/SiteArea/18001a8c-2117-45d2-be1c-baea28a41769"/>
  <link rel="library" href="/wps/mycontenthandler/!ut/p/wcmrest/item/54a68ca2-c550-4385-966f-b0b612174754"/>
  <link rel="parent" href="/wps/mycontenthandler/!ut/p/wcmrest/item/18001a8c-2117-45d2-be1c-baea28a41769/change-to-draft"/>
  <link rel="versions" href="/wps/mycontenthandler/!ut/p/wcmrest/item/18001a8c-2117-45d2-be1c-baea28a41769/versions"/>
  <link rel="default-content" href="/wps/mycontenthandler/!ut/p/wcmrest/Content/null"/>
  <link rel="elements" href="/wps/mycontenthandler/!ut/p/wcmrest/SiteArea/18001a8c-2117-45d2-be1c-baea28a41769/elements"/>
  <updated>2011-05-30T06:44:25.741Z</updated>
  <author>
    <wcm:distinguishedName>uid=wpsadmin,o=defaultWIMFileBasedRealm</wcm:distinguishedName>
    <uri>/wps/mycontenthandler/!ut/p/digest!6GVkh5Ul75Ln7DdEgvHm_g/um/users/profiles/Z9eAEHIC2JG651RC6JM7H9E4MMG6PH06JM45JD0JMO6BEJEJS464JDG31575</uri>
  </author>
  <wcm:owner>
    <wcm:distinguishedName>uid=wpsadmin,o=defaultWIMFileBasedRealm</wcm:distinguishedName>
    <uri>/wps/mycontenthandler/!ut/p/digest!6GVkh5Ul75Ln7DdEgvHm_g/um/users/profiles/Z9eAEHIC2JG651RC6JM7H9E4MMG6PH06JM45JD0JMO6BEJEJS464JDG31575</uri>
  </wcm:owner>
  <wcm:name>SampleSiteAreaNameUpdated</wcm:name>
  <wcm:description>SampleSiteAreaDescriptionUpdated</wcm:description>
  <wcm:type>SiteArea</wcm:type>
  <wcm:state>PUBLISHED</wcm:state>
</entry>

Read

A site area can be read by sending a GET request to the following URI:

/SiteArea/item-uuid

For example:

GET /wps/mycontenthandler/wcmrest/SiteArea/18001a8c-2117-45d2-be1c-baea28a41769 HTTP/1.0

HTTP/1.0 200 OK
Content-type: application/atom+xml; type=entry

<entry xmlns="http://www.w3.org/2005/Atom" xmlns:wcm="wcm/namespace">
  <id>18001a8c-2117-45d2-be1c-baea28a41769</id>
  <title>SampleSiteAreaTitleUpdated</title>
</entry>
Delete

A site area can be deleted by sending a DELETE request to the following URI:
/SiteArea/item-uuid

For example:

DELETE /wps/mycontenthandler/wcmrest/SiteArea/18001a8c-2117-45d2-be1c-baea28a41769 HTTP/1.0

HTTP/1.0 200 OK

Using REST with managed pages

You can use the Web Content Manager REST service to read managed pages.

Read

A managed pages can be read by sending a GET request to the following URI:
/PortalPage/item-uuid

For example:

GET /wps/mycontenthandler/wcmrest/PortalPage/6c523449-e919-41a8-a8d0-f8b1ea207af1 HTTP/1.0

HTTP/1.0 200 OK

Content-type: application/atom+xml; type=entry

<entry xmlns="http://www.w3.org/2005/Atom" xmlns:wcm="wcm/namespace">
  <atom:id>urn:uuid:6c523449-e919-41a8-a8d0-f8b1ea207af1</atom:id>
  <atom:title>RestPortalPage1</atom:title>

  <atom:link atom:rel="edit" atom:href="/wps/mycontenthandler/wcmrest/PortalPage/6c523449-e919-41a8-a8d0-f8b1ea207af1"/>
  <atom:link atom:rel="library" atom:href="/wps/mycontenthandler/wcmrest/item/d07f0a12-3801-465e-bc20-eaec2cecf5cb"/>
</entry>
Chapter 8. Developing
Using REST with drafts and workflows

You can use the REST services for Web Content Manager to create drafts, approve items in a workflow, and move items through different stages of a workflow.

The workflow features of the REST services for Web Content Manager can be used with the following item types:

<table>
<thead>
<tr>
<th>Item type</th>
<th>API type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Text component</td>
<td>LibraryTextComponent</td>
</tr>
<tr>
<td>Short text component</td>
<td>LibraryShortTextComponent</td>
</tr>
<tr>
<td>Rich text component</td>
<td>LibraryRichTextComponent</td>
</tr>
<tr>
<td>HTML Component</td>
<td>LibraryHTMLComponent</td>
</tr>
<tr>
<td>Number component</td>
<td>LibraryNumericComponent</td>
</tr>
<tr>
<td>Date and time component</td>
<td>LibraryDateComponent</td>
</tr>
<tr>
<td>Image component</td>
<td>LibraryImageComponent</td>
</tr>
<tr>
<td>File resource component</td>
<td>LibraryFileComponent</td>
</tr>
<tr>
<td>Content item</td>
<td>Content</td>
</tr>
<tr>
<td>Site Area</td>
<td>SiteArea</td>
</tr>
<tr>
<td>Managed Page</td>
<td>PortalPage</td>
</tr>
</tbody>
</table>

Creating a draft of an item that does not use a workflow

You can create a draft of items that do not use workflows.

To do that, specify explicitly the draft workflow state in the request entry data. For example, to create a link component as draft:

```
HTTP/1.1 POST
http://host:port/wps/mycontenthandler/wcmrest/LibraryLinkComponent

<atom:entry xmlns:atom="http://www.w3.org/2005/Atom" xmlns:wcm="wcm/namespace">
  <atom:name>linkDraft_E</atom:name>
  <wcm:name>linkDraft_E</wcm:name>
  <category label="Draft" scheme="wcmrest:workflowState" term="DRAFT"/>
</atom:entry>
```

201 Created

Creating a draft in a workflow

You can use the REST service to create a new draft item. This is equivalent to using the Create Draft button in the authoring portlet.
A new draft of an item can be created by sending a POST request to the following URI:
/item/item-uuid/create-draft

For example:
HTTP/1.1 POST
http://host:port/wps/mycontenthandler/wcmrest/item/item-uuid/create-draft
201 Created

Moving an item to the next stage

You can use the REST service to move an item to the next stage of a workflow. This is equivalent to using the Next Stage button in the authoring portlet.

An item can be moved to the next stage by sending a POST request to the following URI:
/item/item-uuid/next-stage

For example:
HTTP/1.1 POST
http://host:port/wps/mycontenthandler/wcmrest/item/<item-uuid>/next-stage
201 Created

Moving an item to the previous stage

You can use the REST service to move an item to the previous stage of a workflow. This is equivalent to using the Previous Stage button in the authoring portlet.

An item can be moved to the previous stage by sending a POST request to the following URI:
/item/item-uuid/previous-stage

For example:
HTTP/1.1 POST
http://host:port/wps/mycontenthandler/wcmrest/item/item-uuid/previous-stage
201 Created

Approving an item within a workflow

You can use the REST service to move an item to the next stage of a workflow by approving it. This is equivalent to using the Approve button in the authoring portlet.

An item can be approved by sending a POST request to the following URI:
/item/item-uuid/approve

For example:
HTTP/1.1 POST
http://host:port/wps/mycontenthandler/wcmrest/item/item-uuid/approve
201 Created
Rejecting an item within a workflow

You can use the REST service to reject an item in a workflow. This is equivalent to using the **Reject** button in the authoring portlet.

An item can be rejected by sending a POST request to the following URI:

```
/item/item-uuid/reject
```

For example:

HTTP/1.1 POST
http://host:port/wps/mycontenthandler/wcmrest/item/item-uuid/reject

201 Created

Restarting a workflow

You can use the REST service to restart a workflow. This is equivalent to using the **Restart** button in the authoring portlet.

An item can be restarted by sending a POST request to the following URI:

```
/item/item-uuid/restart
```

For example:

HTTP/1.1 POST
http://host:port/wps/mycontenthandler/wcmrest/item/item-uuid/restart

201 Created

Using REST with projects

You can use the REST services for Web Content Manager to create and work with projects.

Creating a project

To create a project, issue a POST request containing the appropriate data to the following URI.

```
/project/
```

For example:

POST /wps/mycontenthandler/wcmrest/Project HTTP/1.0
Content-type: application/atom+xml

```
<entry xmlns="http://www.w3.org/2005/Atom" xmlns:wcm="wcm/namespace">
  <title>SampleProjectTitle</title>
  <wcm:name>SampleProjectName</wcm:name>
  <wcm:description>SampleProjectDescription</wcm:description>
</entry>
```

HTTP/1.0 Created

```
Content-type: application/atom+xml; type=entry
Content-location: /wps/mycontenthandler/!ut/p/wcmrest/Project/80d503aa-fec5-477c-a8b2-372897982afe
```

```
<entry xmlns="http://www.w3.org/2005/Atom" xmlns:wcm="wcm/namespace">
  <id>80d503aa-fec5-477c-a8b2-372897982afe</id>
  <title>SampleProjectTitle</title>
  <link rel="edit" href="/wps/mycontenthandler/!ut/p/wcmrest/Project/80d503aa-fec5-477c-a8b2-372897982afe"/>
  <link rel="add-item" href="/wps/mycontenthandler/!ut/p/wcmrest/Project/80d503aa-fec5-477c-a8b2-372897982afe/additem"/>
  <link rel="remove-item" href="/wps/mycontenthandler/!ut/p/wcmrest/Project/80d503aa-fec5-477c-a8b2-372897982afe/removeitem"/>
```
Reading a project

To read a project, issue a GET request containing the appropriate data to the following URL.

```
/project/{item-uuid}
```

For example:

```
GET /wps/mycontenthandler/wcmrest/Project/80d503aa-fec5-477c-a8b2-372897982afe HTTP/1.1
HTTP/1.0 200 OK
Content-type: application/atom+xml; type=entry
```

Deleting a project

To delete a project, issue a DELETE request containing the appropriate data to the following URL.

```
/project/{item-uuid}
```

For example:
HTTP/1.1 DELETE
http://host:port/wps/mycontenthandler/wcmrest/Project/35b9120a-17d0-4dcb-b0ba-b034e34b50a6
Accept-Type: application/atom+xml

200 OK

**Updating a project**

To update a project, issue a PUT request containing the appropriate data to the following URI:
/project/{item-uuid}

For example:

HTTP/1.1 PUT
http://host:port/wps/mycontenthandler/wcmrest/Project/35b9120a-17d0-4dcb-b0ba-b034e34b50a6
Accept-Type: application/atom+xml

200 OK

**Adding an item to a project**

To add an item to a project, update the item using a PUT request specifying a link with relation "project" specifying the project to add the item to. For example:

HTTP/1.1 PUT
http://host:port/wps/mycontenthandler/!ut/p/digest!MetYLHV_M5sJbvslx18twA/
wcmrest/LibraryTextComponent/fd34a8bf-7ca1-499c-80ab-acdc2f33cf3e</a>
Accept-Type: application/atom+xml

<?xml version="1.0" encoding="UTF-8"?>
  ... data ...
  <link rel="project" href="/wps/mycontenthandler/wcmrest/Project/77d08cf6-88f6-4577-a929-34d43a8e150e"/>
  ... data ...
</entry>

200 OK

**Note:** If using REST inside a Portal project context, posting to the create-draft link relation will create the draft in a project.

**Removing an item from a project**

To remove an item from a project, update the item using a PUT request without specifying the project link relation. For example:

HTTP/1.1 PUT
http://host:port/wps/mycontenthandler/!ut/p/digest!MetYLHV_M5sJbvslx18twA/
wcmrest/LibraryTextComponent/fd34a8bf-7ca1-499c-80ab-acdc2f33cf3e</a>
Accept-Type: application/atom+xml

200 OK

**Submitting a project for review**

 Editors of a project can submit a project for review by issuing a POST request containing the appropriate data to the following URI:
/project/{project-uuid}/submit-for-review

For example:
Withdrawing a project from review

Editors of a project can withdraw a project from review by issuing a POST request containing the appropriate data to the following URI:
/project/{project-uid}/withdraw-from-review

For example:
HTTP/1.1 POST
http://host:port/wps/mycontenthandler/wcmrest/Project/PROJECT-UUID/withdraw-from-review
Accept-Type: application/atom+xml
201 Created

When withdrawn, the project returns to an active state.

Approving a project

Approvers of a project can approve a project by issuing a POST request containing the appropriate data to the following URI:
/project/{project-uid}/approve

For example:
HTTP/1.1 POST
http://host:port/wps/mycontenthandler/wcmrest/Project/PROJECT-UUID/approve
Accept-Type: application/atom+xml
201 Created

When approved, the project is ready to be published.

Rejecting a project

Approvers of a project can reject project approval by issuing a POST request containing the appropriate data to the following URI:
/project/{project-uid}/reject

For example:
HTTP/1.1 POST
http://host:port/wps/mycontenthandler/wcmrest/Project/PROJECT-UUID/reject
Accept-Type: application/atom+xml
201 Created

When rejected, the project returns to an active state.

Withdrawing approval for a project

Approvers of a project can withdraw approval for a project by issuing a POST request containing the appropriate data to the following URI:
/project/{project-uid}/withdraw-approval

For example:
Withdrawing approval does not change the state of the project, which remains in review.

**Using REST to work with item identity controls**
The way identity controls are used with REST depend on whether a text provider has been enabled or not for an item.

**Creating, updating or reading identity controls with no text provider enabled**

When no text provider has been enabled on an item, identity controls work this way:

*Table 183. REST tags with no text provider enabled*

<table>
<thead>
<tr>
<th>REST Tag</th>
<th>Item field</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;wcm:name&gt;text&lt;/wcm:name&gt;</code></td>
<td>Name</td>
</tr>
<tr>
<td><code>&lt;atom:title&gt;text&lt;/atom:title&gt;</code></td>
<td>Display title</td>
</tr>
<tr>
<td><code>&lt;atom:summary&gt;text&lt;/atom:summary&gt;</code></td>
<td>Description</td>
</tr>
</tbody>
</table>

**Enabling updating or reading text provider controls**

To either enable, update or read text provider controls, you add the following parameters to the title tag:

*Table 184. REST tags for text providers*

<table>
<thead>
<tr>
<th>REST Tag</th>
<th>Text provider field</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;wcm:title TextProviderName&gt;text&lt;/wcm:title TextProviderName&gt;</code></td>
<td>Title name</td>
</tr>
<tr>
<td><code>&lt;wcm:title TextProviderKey&gt;text&lt;/wcm:title TextProviderKey&gt;</code></td>
<td>Title key</td>
</tr>
<tr>
<td><code>&lt;wcm:desc TextProviderName&gt;text&lt;/wcm:desc TextProviderName&gt;</code></td>
<td>Description name</td>
</tr>
<tr>
<td><code>&lt;wcm:desc TextProviderKey&gt;text&lt;/wcm:desc TextProviderKey&gt;</code></td>
<td>Description key</td>
</tr>
</tbody>
</table>

**Creating, updating or reading identity controls with a text provider enabled**

When a text provider has been enabled on an item, identity controls work this way:

*Table 185. REST tags with a text provider enabled*

<table>
<thead>
<tr>
<th>REST Tag</th>
<th>Item field</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;wcm:name&gt;text&lt;/wcm:name&gt;</code></td>
<td>Name</td>
</tr>
<tr>
<td><code>&lt;wcm:displayTitle&gt;text&lt;/wcm:displayTitle&gt;</code></td>
<td>Display title</td>
</tr>
</tbody>
</table>
Table 185. REST tags with a text provider enabled (continued)

<table>
<thead>
<tr>
<th>REST Tag</th>
<th>Item field</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;atom:title lang=&quot;language code&quot;&gt;text&lt;/atom:title&gt;</code></td>
<td>Localized title</td>
</tr>
<tr>
<td>For example:<code>&lt;atom:title lang=&quot;fr&quot;&gt;text&lt;/atom:title&gt;</code></td>
<td><strong>Note:</strong> Once a text provider is enabled, the localized title can only be read by the REST service. You will need to update the text provider plugin directly to change the localized title.</td>
</tr>
<tr>
<td><code>&lt;wcm:description&gt;text&lt;/wcm:description&gt;</code></td>
<td>Description</td>
</tr>
<tr>
<td><code>&lt;atom:summary lang=&quot;language code&quot;&gt;text&lt;/atom:summary&gt;</code></td>
<td>Localized description</td>
</tr>
<tr>
<td>For example:<code>&lt;atom:summary lang=&quot;fr&quot;&gt;text&lt;/atom:summary&gt;</code></td>
<td><strong>Note:</strong> Once a text provider is enabled, the localized description can only be read by the REST service. You will need to update the text provider plugin directly to change the localized description.</td>
</tr>
</tbody>
</table>

**Note:** The language code required for the localized tags must be an IETF BCP47 compliant language code.

**Using REST to work with access controls**

The access control section in an Atom entry or JSON entry generated by the REST service for Web Content Manager links to resources from the Portal Access Control REST API.

This is an example of how the access controls are displayed:

```xml
<link rel="access-control" href="/wps/mycontenthandler/lut/p/digest1ZqvndlckP1B8g0HkycCA/ac/access:oid:Z6QReDeOHP0J62BO4JM462R06JM664JC2JM07K1D6JM07MHCE3QKCH1D8619633"/>
```

You can issue a GET request to the URI and obtain the access control information of an item generated by the Portal Access Control REST API. See Portal Access Control REST API for further information.

**Using REST to work with author and owner parameters**

Information for the author and owner of an item can be specified when using the REST service.

The following attributes are used:

- name
- distinguishedName
- A URI to fetch detailed information of the author or owner using the PUMA REST SPI

You can also specify such information in the entry in the same format for a POST or PUT operation to Create or Update a WCM item with these information you specified.

**Example**

This example is in application/atom+xml format:

```xml
<entry xmlns="http://www.w3.org/2005/Atom" xmlns:wcm="wcm/namespace">
  ...
  <author>
```

Chapter 8. Developing  729
Using REST to work with versions

Item versions can be listed and read using the REST service.

Retrieving a list of versions

To retrieve a list of versions you issue a GET request to the following URI:

/itm/item-uuid/versions

A feed is returned containing the identifying information of each version, along with a with relation "versioned-item" that links to the specified version.

Version link relations

Versions feed

Viewing the details of a version

To view the details of a specified version, you issue a GET request in the following format:

/itm/item-uuid/version/version-name

For example:

GET HTTP/1.0
/wps/mycontenthandler/wcmrest/item/8f055ba2-1bc3-4d21-8443-86274e14dd2c/version/1
Host: www.example.com
Accept: application/atom+xml
Using REST to work with recent items

You can use REST service to display a list of recently accessed items. This is the equivalent of the Recent Items view in the library explorer.

**Note:** By default, items accessed through the REST service are added to the recent items list. To stop items accessed through the REST service appearing in the list of recent items, change the `rest.default.add-recent` parameter to false in the WCM WCMConfigService service using the WebSphere Integrated Solutions Console. The `rest.default.add-recent` parameter can be overridden on a per-request basis by specifying `recent=true` or `recent=false` in the query string.

**URI:** /recent-items

**Example:**

```
HTTP 1.1 GET /wps/mycontenthandler/wcmrest/recent-items
```

```
   <id>wcmrest:recent-items</id>
   <title>wcmrest:recent-items</title>
   <updated>2011-09-28T02:51:47.228Z</updated>
   <link rel="next-page" href="/wps/mycontenthandler/lut/p/wcmrest/recent-items?page=2"/>
   <entry>
      <id>wcmrest:8b629b12-e16a-4afa-bb5f-a37bee5a5b8</id>
      <title xml:lang="en">Articles List</title>
      <summary xml:lang="en"></summary>
      <wcm:name>Articles List</wcm:name>
      <wcm:description xml:lang="en"></wcm:description>
      <wcm:type>LibraryMenuComponent</wcm:type>
      <updated>2011-09-28T02:51:47.228Z</updated>
   </entry>
</feed>
```
There is one additional field present in each of the entries when a recent items query is performed. This field is the last accessed date of the item. This indicates the date and time at which the item was last viewed or edited through the authoring portlet. For example:


**Parameters**

The following parameters, along with mime-type, are the only parameters that will work with the returned feed. All other parameters will be ignored.
Table 186. Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Details</th>
</tr>
</thead>
</table>
| sort      | The sort parameter is appended to queries to determine how query results are sorted. The following values can be used with the sort parameter.  
  • accessed  
  • author  
  • created  
  • modified  
  • name  
  • title  

  The values _ascending or _descending are appended to the query to determine sort order.  

  For example, to sort a presentation template query in ascending order of creation, you would use:  
  /recent-items?type=PresentationTemplate&sort=created_ascending  

  To sort a presentation template query in descending order of creation, you would use:  
  /recent-items?type=PresentationTemplate&sort=created_descending  

  If _ascending or _descending are not specified, the results as displayed in ascending order. |
| type      | This parameter is used to query items of a specific item type.  

  For example, to query a list of components:  
  /recent-items?type=LibraryHTMLComponent |
| page      | This parameter is used with the pagesize parameter to define what set of results to display. For example, if pagesize is set to 5, and the page parameter is set to 2, then only results 6 to 10 will be displayed.  

  For example:  
  /recent-items?type=PresentationTemplate&page=2 |
| pagesize  | This parameter is used to restrict the number of items returned by a query to a set number. It can be used with the page parameter to return specific pages of results.  

  For example, to restrict the number of queries to be returned to 5:  
  /recent-items?type=PresentationTemplate&pagesize=5 |

Using REST to work with favorite items

You can use REST service to display a list of favorite items. This is the equivalent of the Favorite Items view in the library explorer.
URI: /favorite-items

Example:

```xml
<?xml version="1.0" encoding="UTF-8"?>
  <id>wcmrest:favorite-items</id>
  <title>favorite-items</title>
  <updated>2012-01-31T01:19:00.564Z</updated>
  <entry>
    <id>wcmrest:438dc2be-dbda-40bb-ad78-3c0f4bd11107</id>
    <title lang="en">Sample Article 2</title>
    <wcm:displayTitle lang="en">Sample Article 2</wcm:displayTitle>
    <summary lang="en"></summary>
    <wcm:name>Sample Article 2</wcm:name>
    <wcm:type>Content</wcm:type>
    <updated>2012-01-24T01:13:45.194Z</updated>
    <wcm:lastModifier>
      <wcm:distinguishedName>Replicator</wcm:distinguishedName>
    </wcm:lastModifier>
    <link label="Edit" rel="edit" href="/wps/mycontenthandler/!ut/p/digest!PQo5Yhy68oeppWcEz2sddA/wcmrest/Content/438dc2be-dbda-40bb-ad78-3c0f4bd11107">
      <link label="Library" rel="library" href="/wps/mycontenthandler/!ut/p/digest!PQo5Yhy68oeppWcEz2sddA/wcmrest/item/a423287f-b0ce-4ee3-9c95">
        <category label="Published" scheme="wcmrest:workflowState" term="PUBLISHED" lang="en"/>
      </link>
    </link>
  </entry>
  <entry>
    <id>wcmrest:715cd5e8-ec36-420b-ad1c-fff80f39462b</id>
    <title lang="en">Sample Article</title>
    <wcm:displayTitle lang="en">Sample Article</wcm:displayTitle>
    <summary lang="en"></summary>
    <wcm:name>Sample Article</wcm:name>
    <wcm:type>Content</wcm:type>
    <updated>2012-01-24T01:13:47.981Z</updated>
    <wcm:lastModifier>
      <wcm:distinguishedName>Replicator</wcm:distinguishedName>
    </wcm:lastModifier>
    <link label="Edit" rel="edit" href="/wps/mycontenthandler/!ut/p/digest!PQo5Yhy68oeppWcEz2sddA/wcmrest/Content/715cd5e8-ec36-420b-ad1c-fff80f39462b">
      <link label="Library" rel="library" href="/wps/mycontenthandler/!ut/p/digest!PQo5Yhy68oeppWcEz2sddA/wcmrest/item/a423287f-b0ce-4ee3-9c95">
        <category label="Published" scheme="wcmrest:workflowState" term="PUBLISHED" lang="en"/>
      </link>
    </link>
  </entry>
</feed>
```

Parameters

The following parameters, along with mime-type, are the only parameters that will work with the returned feed. All other parameters will be ignored.
### Table 187. Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Details</th>
</tr>
</thead>
</table>
| **sort**  | The sort parameter is appended to queries to determine how query results are sorted. The following values can be used with the sort parameter.  
  • author  
  • created  
  • modified  
  • name  
  • title  
  
The values _ascending or _descending are appended to the query to determine sort order.  
  
  For example, to sort a presentation template query in ascending order of creation, you would use:  
  `/favorite-items?type=PresentationTemplate&sort=created_ascending`  
  
  To sort a presentation template query in descending order of creation, you would use:  
  `/favorite-items?type=PresentationTemplate&sort=created_descending`  
  
  If _ascending or _descending are not specified, the results as displayed in ascending order. |
| **type**   | This parameter is used to query items of a specific item type.  
  
  For example, to query a list of components:  
  `/favorite-items?type=LibraryHTMLComponent` |
| **page**   | This parameter is used with the pagesize parameter to define what set of results to display. For example, if pagesize is set to 5, and the page parameter is set to 2, then only results 6 to 10 will be displayed.  
  
  For example:  
  `/favorite-items?type=PresentationTemplate&page=2&pagesize=5` |
| **pagesize** | This parameter is used to restrict the number of items returned by a query to a set number. It can be used with the page parameter to return specific pages of results.  
  
  For example, to restrict the number of queries to be returned to 5:  
  `/favorite-items?type=PresentationTemplate&page=2&pagesize=5` |
Adding and removing favorite items

For item types that have an explicit REST URI, such as types that can be created or updated, the item can be added by performing a HTTP PUT to update the item, with the PUT request containing the favorites category.

For example:

HTTP 1.1 PUT /wps/mycontenthandler/wcmrest/LibraryHTMLComponent/47018149-fc6b-46af-a54d-1eab89a6fed7

<?xml version="1.0" encoding="UTF-8"?>
  ...
  <category scheme="wcmrest:favorite" term="true" lang="en"/>
  ...
</entry>

- term="true" will add an item to the list of favorite items.
- term="false" will remove an item from the list of favorite items.
- If the "term" parameter is not present, the item will be removed from the list of favorite items.

For items that do not have an explicit REST URI, a HTTP POST request can be made specifying the generic URI of the item. For this to work the item must be created already.

For example:

Adding:

HTTP 1.1 POST /wps/mycontenthandler/wcmrest/favorite-items/additem?item-uri=/wps/mycontenthandler/lut/p/digestyGlcBv5s09Vb0EY9LLhJyQ/wcmrest/i

Removing:

HTTP 1.1 POST /wps/mycontenthandler/wcmrest/favorite-items/removeitem?item-uri=/wps/mycontenthandler/lut/p/digestyGlcBv5s09Vb0EY9LLhJyQ/wcmrest/i

Note: The item-uri parameter specifies the item to add, and may be in the expanded form, as shown in the previous examples, or the compact form. For example:

wcmrest:item/65a46943-ed1c-4f5d-b497-03c18886ca8e

REST: Attachments

You can use the REST service to attach images to some item types. This is equivalent to using the Insert An Image button in the authoring portlet.

An attachment is an image resource that is associated with another item containing HTML, and can be referenced from within that item. Attachments are a special kind of item in the REST service because they cannot be directly referenced. This is because there is no URI associated with an attachment once it is created. Due to this, an attachment cannot be read or updated, but can be deleted indirectly by performing an update on the parent item.

Attachments can be added to the following item types:

- PresentationTemplate
- LibraryRichTextComponent
- LibraryHTMLComponent

Note: You must create the item containing a rich text or HTML element before creating the attachment.
Creating
/ITEM-TYPE/ITEM-UUID/attachments
Content-Type: image/*

When creating an attachment, the binary data, but not encoding, of an image is sent to the attachments collection of an item. The Content-Type header field is set to the appropriate image type. For example: image/jpg, image/png

Example:
HTTP/1.1 POST
http://host:port/wps/mycontenthandler/wcmrest/LibraryHTMLComponent/ITEM-UUID/attachments
Content-Type: image/jpg
(... binary data ...)
201 Created

Reading
Not supported.

Updating
Not supported.

Deleting
An attachment cannot be directly referenced through the REST service, which means it cannot be directly deleted. However, it can be deleted indirectly by performing an update operation on the parent item.

For example, this is some markup stored in a HTML component:
<h1> Example Delete </h1>
<img src="/wps/wcm/myconnect/65132264-fd8b-461c-b6ec-ccdd2524ea6/image.jpg?MOD=AJPERES" alt='' title='' border='0' />

To remove the image, you would make the following update request:
HTTP/1.1 PUT
http://host:port/wps/mycontenthandler/wcmrest/LibraryHTMLComponent/ITEM-UUID/
Content-Type: text/html

<h1> Example Delete </h1>

200 OK

Generic reading using REST services for Web Content Manager
Although not all item types are handled by the REST service, all item types can be read in a generic fashion using the REST service.

You can issue a GET request to any web content item using the following URI:
/item/{item-uuid}

Example
GET /wps/mycontenthandler/wcmrest/item/c98d11e1-7f2a-480e-9aac-40eb1949cbda HTTP/1.0

HTTP/1.0 200 OK
<entry xmlns="http://www.w3.org/2005/Atom" xmlns:wcm="wcm/namespace">
Reference material for the Web Content Manager REST service

These topics include reference material relating to response codes, links, media types and attachments.

Response codes for the Web Content Manager REST service

These response codes are those generated by the Web Content Manager REST service.

Table 188. Response codes

<table>
<thead>
<tr>
<th>Response code</th>
<th>Message</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>OK</td>
<td>The operation completed successfully.</td>
</tr>
<tr>
<td>201</td>
<td>Created</td>
<td>The resource was successfully created.</td>
</tr>
<tr>
<td>301</td>
<td>Moved Permanently</td>
<td>The resource addressed is known, however, its URI has changed.</td>
</tr>
<tr>
<td>400</td>
<td>Bad Request</td>
<td>Generic client side error. The request data is invalid in some way.</td>
</tr>
<tr>
<td>403</td>
<td>Forbidden</td>
<td>The request is formed correctly, but the server cannot carry out the operation.</td>
</tr>
<tr>
<td>404</td>
<td>Not Found</td>
<td>The URI specified is unknown to the REST service.</td>
</tr>
<tr>
<td>405</td>
<td>Method Not Allowed</td>
<td>The addressed resource does not support the HTTP method used.</td>
</tr>
<tr>
<td>406</td>
<td>Not Acceptable</td>
<td>The client specified an unsupported accept type.</td>
</tr>
<tr>
<td>409</td>
<td>Conflict</td>
<td>The request attempted to put the resource in an impossible or inconsistent state.</td>
</tr>
<tr>
<td>415</td>
<td>Unsupported Media Type</td>
<td>The server did not recognize the specified media type.</td>
</tr>
<tr>
<td>423</td>
<td>Locked</td>
<td>Unable to perform the requested operation, as the resource is locked.</td>
</tr>
<tr>
<td>500</td>
<td>Internal Server Error</td>
<td>An internal error occurred.</td>
</tr>
</tbody>
</table>
**Link relations**

These tables provide information on how different requests can be linked.

*Table 189. Link relations*

<table>
<thead>
<tr>
<th>Relation</th>
<th>Purpose</th>
<th>Supported Operation to the Link HREF</th>
<th>Settable for PUT and POST operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>add-attachment</td>
<td>Add attachment to the item (array component only, such as HTML, RichText component/element and PresentationTemplate).</td>
<td>POST</td>
<td>NO</td>
</tr>
<tr>
<td>approve</td>
<td>Approve the item in current stage and move it into next stage in the workflow.</td>
<td>POST</td>
<td>NO</td>
</tr>
<tr>
<td>content-template</td>
<td>The content template of the item.</td>
<td>GET</td>
<td>YES</td>
</tr>
<tr>
<td>change-to-draft</td>
<td>Go to the draft of the item.</td>
<td>POST</td>
<td>NO</td>
</tr>
<tr>
<td>create-draft</td>
<td>Create a draft of the item.</td>
<td>POST</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>Will also add item to a project if created within a portal project context.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>decline</td>
<td>Decline the item in current stage and move it back to previous stage in the workflow.</td>
<td>POST</td>
<td>NO</td>
</tr>
<tr>
<td>draft-of</td>
<td>The published item link of the draft item.</td>
<td>GET</td>
<td>NO</td>
</tr>
<tr>
<td>draft</td>
<td>The draft link of the published item.</td>
<td>GET</td>
<td>NO</td>
</tr>
<tr>
<td>edit-media</td>
<td>Pointing to the item itself with raw data type specified in the 'type' attribute. Designed to be use for 'Raw Data' Read and Update.</td>
<td>PUT, GET</td>
<td>NO</td>
</tr>
<tr>
<td>edit</td>
<td>Pointing to item itself.</td>
<td>PUT, POST, GET, DELETE</td>
<td>NO</td>
</tr>
<tr>
<td>elements</td>
<td>All the elements in the content/sitearea.</td>
<td>GET</td>
<td>NO</td>
</tr>
<tr>
<td>library</td>
<td>The library which the item resides in.</td>
<td>GET</td>
<td>YES</td>
</tr>
<tr>
<td>next-stage</td>
<td>Move item to next stage in the workflow.</td>
<td>POST</td>
<td>NO</td>
</tr>
<tr>
<td>Relation</td>
<td>Purpose</td>
<td>Supported Operation to the Link HREF</td>
<td>Settable for PUT and POST operation</td>
</tr>
<tr>
<td>-------------</td>
<td>--------------------------------------------------------------------------</td>
<td>--------------------------------------</td>
<td>-------------------------------------</td>
</tr>
<tr>
<td>parent</td>
<td>The parent of the item.</td>
<td>GET</td>
<td>YES, overrides the library if both specified</td>
</tr>
<tr>
<td>previous-stage</td>
<td>Move item to previous stage in the workflow.</td>
<td>POST</td>
<td>NO</td>
</tr>
<tr>
<td>project</td>
<td>The project which the item belongs to, if it is a draft in a project.</td>
<td>GET</td>
<td>YES</td>
</tr>
<tr>
<td>restart</td>
<td>Restart the workflow for the item and move to the draft stage.</td>
<td>POST</td>
<td>NO</td>
</tr>
<tr>
<td>sitearea-template</td>
<td>The site area template of the item.</td>
<td>GET</td>
<td>YES</td>
</tr>
<tr>
<td>versioned-item</td>
<td>The individual version of the item.</td>
<td>GET</td>
<td>NO</td>
</tr>
<tr>
<td>versions</td>
<td>All versions of the item.</td>
<td>GET</td>
<td>NO</td>
</tr>
<tr>
<td>workflow-stage</td>
<td>The workflow stage of the item.</td>
<td>GET</td>
<td>NO</td>
</tr>
<tr>
<td>workflow</td>
<td>The workflow of the item.</td>
<td>GET</td>
<td>YES</td>
</tr>
</tbody>
</table>

### Table 190. Examples

<table>
<thead>
<tr>
<th>Relation</th>
<th>Purpose</th>
<th>Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>add-attachment</td>
<td>&lt;atom:links atom:rel=&quot;add-attachment&quot; atom:href=&quot;wcmrest:LibraryHTMLComponent/c0b72020-10b7-4197-a436-62a1d94ce03f/attachments&quot;/&gt;</td>
<td></td>
</tr>
<tr>
<td>approve</td>
<td>&lt;atom:links atom:rel=&quot;approve&quot; atom:href=&quot;/wps/mycontenthandler/wcmrest/item/68b3bbb5-3b36-4c1f-311b-6c3a037c975a/approve&quot;/&gt;</td>
<td></td>
</tr>
<tr>
<td>authoring-template</td>
<td>&lt;atom:links atom:rel=&quot;authoring-template&quot; atom:href=&quot;/wps/mycontenthandler/wcmrest/item/18cfc80c-a490-4d75-9857-fed3db9de53&quot;/&gt;</td>
<td></td>
</tr>
<tr>
<td>change-to-draft</td>
<td>&lt;atom:links atom:rel=&quot;change-to-draft&quot; atom:href=&quot;/wps/mycontenthandler/wcmrest/item/c0b72020-10b7-4197-a436-62a1d94ce03f/change-to-draft&quot;/&gt;</td>
<td></td>
</tr>
<tr>
<td>create-draft</td>
<td>&lt;atom:links atom:rel=&quot;create-draft&quot; atom:href=&quot;/wps/mycontenthandler/wcmrest/item/e5846504-e4ae-496f-8f33-c06a8bfc31d/create-draft&quot;/&gt;</td>
<td></td>
</tr>
<tr>
<td>decline</td>
<td>&lt;atom:links atom:rel=&quot;decline&quot; atom:href=&quot;/wps/mycontenthandler/wcmrest/item/2cale0ce-3cc9-4810-b4d8-b28738286492/decline&quot;/&gt;</td>
<td></td>
</tr>
<tr>
<td>draft-of</td>
<td>&lt;atom:links atom:rel=&quot;draft-of&quot; atom:href=&quot;/wps/mycontenthandler/wcmrest/LibraryDateComponent/edbc837-e9f0-4766-a4ec-e7fcdef12ca7&quot;/&gt;</td>
<td></td>
</tr>
<tr>
<td>draft</td>
<td>&lt;atom:links atom:rel=&quot;draft&quot; atom:href=&quot;/wps/mycontenthandler/wcmrest/LibraryDateComponent/56937d4-79ac-4963-bb8c-6da57b8dd55&quot;/&gt;</td>
<td></td>
</tr>
<tr>
<td>edit-media</td>
<td>&lt;atom:links atom:rel=&quot;edit-media&quot; atom:type=&quot;text/html&quot; atom:href=&quot;/wps/mycontenthandler/wcmrest/LibraryHTMLComponent/10d5f7ca-f2a2-46b8-6b49-e76a1ee8edee&quot;/&gt;</td>
<td></td>
</tr>
<tr>
<td>edit</td>
<td>&lt;atom:links atom:rel=&quot;edit&quot; atom:href=&quot;/wps/mycontenthandler/wcmrest/LibraryHTMLComponent/10d5f7ca-f2a2-46b8-6b49-e76a1ee8edee&quot;/&gt;</td>
<td></td>
</tr>
</tbody>
</table>
Table 190. Examples (continued)

<table>
<thead>
<tr>
<th>Relation</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>elements</td>
<td>&lt;atom:links atom:rel=&quot;elements&quot; atom:href=&quot;wcmrest:Content/9d9b133b-1bab-40e7-a9bd-5b0ac86cf628/elements&quot;/&gt;</td>
</tr>
<tr>
<td>library</td>
<td>&lt;atom:links atom:rel=&quot;library&quot; atom:href=&quot;/wps/mycontenthandler/wcmrest/item/957a67f2-9d70-469f-9d43-f63f78508e48&quot;/&gt;</td>
</tr>
<tr>
<td>next-stage</td>
<td>&lt;atom:links atom:rel=&quot;next-stage&quot; atom:href=&quot;/wps/mycontenthandler/wcmrest/item/a93ce36a-7a0d-4bda-e1f-e8db09295c8b/next-stage&quot;/&gt;</td>
</tr>
<tr>
<td>parent</td>
<td>&lt;atom:links atom:rel=&quot;parent&quot; atom:href=&quot;/wps/mycontenthandler/wcmrest/item/fbcc2395-ca4c-44f2-5cb7-5f4ca359500f&quot;/&gt;</td>
</tr>
<tr>
<td>previous-stage</td>
<td>&lt;atom:links atom:rel=&quot;previous-stage&quot; atom:href=&quot;/wps/mycontenthandler/wcmrest/item/e5846504-e4ae-496f-e33-c06a8bfc83d/previous-stage&quot;/&gt;</td>
</tr>
<tr>
<td>project</td>
<td>&lt;atom:links atom:rel=&quot;project&quot; atom:href=&quot;/wps/mycontenthandler/wcmrest/Project/35b9120a-17d0-4dbb-b0ba-b034e34b50a6&quot;/&gt;</td>
</tr>
<tr>
<td>restart</td>
<td>&lt;atom:links atom:rel=&quot;restart&quot; atom:href=&quot;/wps/mycontenthandler/wcmrest/item/e5846504-e4ae-496f-e33-c06a8bfc83d/restart&quot;/&gt;</td>
</tr>
<tr>
<td>versioned-item</td>
<td>&lt;atom:links atom:rel=&quot;versioned-item&quot; atom:href=&quot;/wps/mycontenthandler/wcmrest/item/c0b72020-10b7-449e-e436-62a1d94e03f/version/1&quot;/&gt;</td>
</tr>
<tr>
<td>versions</td>
<td>&lt;atom:links atom:rel=&quot;versions&quot; atom:href=&quot;/wps/mycontenthandler/wcmrest/item/a93ce36a-7a0d-4bda-e1f-e8db09295c8b/versions&quot;/&gt;</td>
</tr>
<tr>
<td>workflow-stage</td>
<td>&lt;atom:links atom:rel=&quot;workflow-stage&quot; atom:href=&quot;/wps/mycontenthandler/wcmrest/item/52c43f50-7a4e-4ad2-618a-8975d2362219&quot;/&gt;</td>
</tr>
<tr>
<td>workflow</td>
<td>&lt;atom:links atom:rel=&quot;workflow&quot; atom:href=&quot;/wps/mycontenthandler/wcmrest/item/8d25860b-7a5c-4015-9cd5-bdcd6001e1bb&quot;/&gt;</td>
</tr>
</tbody>
</table>

**Supported media types**
The following media types are supported by the Web Content Manager REST service.

**Meta Data Media Types**

Table 191. Meta Data Media Types

<table>
<thead>
<tr>
<th>Item Description</th>
<th>Item Type</th>
<th>Supported Media Types</th>
<th>Supported Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>Content Template</td>
<td>ContentTemplate</td>
<td>application/atom+xml application/json</td>
<td>GET</td>
</tr>
<tr>
<td>Content</td>
<td>Content</td>
<td>application/atom+xml application/json</td>
<td>GET, PUT, POST</td>
</tr>
<tr>
<td>Date Component</td>
<td>LibraryDateComponent</td>
<td>application/atom+xml application/json</td>
<td>GET, PUT, POST</td>
</tr>
<tr>
<td>Date Element</td>
<td>DateComponent</td>
<td>application/atom+xml application/json</td>
<td>GET, PUT, POST</td>
</tr>
<tr>
<td>File Component</td>
<td>LibraryFileComponent</td>
<td>application/atom+xml application/json</td>
<td>GET, PUT, POST</td>
</tr>
<tr>
<td>File Element</td>
<td>FileComponent</td>
<td>application/atom+xml application/json</td>
<td>GET, PUT, POST</td>
</tr>
<tr>
<td>Item Description</td>
<td>Item Type</td>
<td>Supported Media Types</td>
<td>Supported Methods</td>
</tr>
<tr>
<td>--------------------</td>
<td>-----------------</td>
<td>---------------------------</td>
<td>-------------------</td>
</tr>
<tr>
<td>HTML Component</td>
<td>LibraryHTMLComponent</td>
<td>application/atom+xml, application/json</td>
<td>GET, PUT, POST</td>
</tr>
<tr>
<td>HTML Element</td>
<td>HTMLComponent</td>
<td>application/atom+xml, application/json</td>
<td>GET, PUT, POST</td>
</tr>
<tr>
<td>Image Component</td>
<td>LibraryImageComponent</td>
<td>application/atom+xml, application/json</td>
<td>GET, PUT, POST</td>
</tr>
<tr>
<td>Image Element</td>
<td>ImageComponent</td>
<td>application/atom+xml, application/json</td>
<td>GET, PUT, POST</td>
</tr>
<tr>
<td>Numeric Component</td>
<td>LibraryNumericComponent</td>
<td>application/atom+xml, application/json</td>
<td>GET, PUT, POST</td>
</tr>
<tr>
<td>Numeric Element</td>
<td>NumericComponent</td>
<td>application/atom+xml, application/json</td>
<td>GET, PUT, POST</td>
</tr>
<tr>
<td>Portal Page</td>
<td>PortalPage</td>
<td>application/atom+xml, application/json</td>
<td>GET</td>
</tr>
<tr>
<td>Presentation Template</td>
<td>PresentationTemplate</td>
<td>application/atom+xml, application/json</td>
<td>GET, PUT, POST</td>
</tr>
<tr>
<td>Project Template</td>
<td>ProjectTemplate</td>
<td>application/atom+xml, application/json</td>
<td>GET, PUT, POST</td>
</tr>
<tr>
<td>Project</td>
<td>Project</td>
<td>application/atom+xml, application/json</td>
<td>GET, PUT, POST</td>
</tr>
<tr>
<td>Rich Text Component</td>
<td>LibraryRichTextComponent</td>
<td>application/atom+xml, application/json</td>
<td>GET, PUT, POST</td>
</tr>
<tr>
<td>Rich Text Element</td>
<td>RichTextComponent</td>
<td>application/atom+xml, application/json</td>
<td>GET, PUT, POST</td>
</tr>
<tr>
<td>Short Text Component</td>
<td>LibraryShortTextComponent</td>
<td>application/atom+xml, application/json</td>
<td>GET, PUT, POST</td>
</tr>
<tr>
<td>Short Text Element</td>
<td>ShortTextComponent</td>
<td>application/atom+xml, application/json</td>
<td>GET, PUT, POST</td>
</tr>
<tr>
<td>Site Area Template</td>
<td>SiteAreaTemplate</td>
<td>application/atom+xml, application/json</td>
<td>GET</td>
</tr>
<tr>
<td>Site Area</td>
<td>SiteArea</td>
<td>application/atom+xml, application/json</td>
<td>GET, PUT, POST</td>
</tr>
</tbody>
</table>
### Table 191. Meta Data Media Types (continued)

<table>
<thead>
<tr>
<th>Item Description</th>
<th>Item Type</th>
<th>Supported Media Types</th>
<th>Supported Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stylesheet Component</td>
<td>LibraryStyleSheetComponent</td>
<td>application/atom+xml, application/json</td>
<td>GET, PUT, POST</td>
</tr>
<tr>
<td>Text Component</td>
<td>LibraryTextComponent</td>
<td>application/atom+xml, application/json</td>
<td>GET, PUT, POST</td>
</tr>
<tr>
<td>Text Element</td>
<td>TextComponent</td>
<td>application/atom+xml, application/json</td>
<td>GET, PUT, POST</td>
</tr>
</tbody>
</table>

### Raw Data Media Types

<table>
<thead>
<tr>
<th>Item Description</th>
<th>Item Type</th>
<th>Supported Media Types</th>
<th>Supported Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>Content Template</td>
<td>ContentTemplate</td>
<td>application/vnd.ibm.wcm+xml</td>
<td>GET, PUT</td>
</tr>
<tr>
<td>Content</td>
<td>Content</td>
<td>application/vnd.ibm.wcm+xml</td>
<td>GET, PUT</td>
</tr>
<tr>
<td>Date Component</td>
<td>LibraryDateComponent</td>
<td>application/vnd.ibm.wcm+xml</td>
<td>GET, PUT</td>
</tr>
<tr>
<td>Date Element</td>
<td>DateComponent</td>
<td>application/vnd.ibm.wcm+xml</td>
<td>GET, PUT</td>
</tr>
<tr>
<td>File Component</td>
<td>LibraryFileComponent</td>
<td><em>/</em></td>
<td>GET, PUT</td>
</tr>
<tr>
<td>File Element</td>
<td>FileComponent</td>
<td><em>/</em></td>
<td>GET, PUT</td>
</tr>
<tr>
<td>HTML Component</td>
<td>LibraryHTMLComponent</td>
<td>text/html</td>
<td>GET, PUT</td>
</tr>
<tr>
<td>HTML Element</td>
<td>HTMLComponent</td>
<td>text/html</td>
<td>GET, PUT</td>
</tr>
<tr>
<td>Image Component</td>
<td>LibraryImageComponent</td>
<td>image/* application/vnd.ibm.wcm+xml</td>
<td>GET, PUT</td>
</tr>
<tr>
<td>Image Element</td>
<td>ImageComponent</td>
<td>image/* application/vnd.ibm.wcm+xml</td>
<td>GET, PUT</td>
</tr>
<tr>
<td>Numeric Component</td>
<td>LibraryNumericComponent</td>
<td>application/vnd.ibm.wcm+xml</td>
<td>GET, PUT</td>
</tr>
<tr>
<td>Numeric Element</td>
<td>NumericComponent</td>
<td>application/vnd.ibm.wcm+xml</td>
<td>GET, PUT</td>
</tr>
<tr>
<td>Portal Page</td>
<td>PortalPage</td>
<td>application/vnd.ibm.wcm+xml</td>
<td>GET</td>
</tr>
<tr>
<td>Presentation Template</td>
<td>PresentationTemplate</td>
<td>text/html</td>
<td>GET, PUT</td>
</tr>
<tr>
<td>Project Template</td>
<td>ProjectTemplate</td>
<td>application/vnd.ibm.wcm+xml</td>
<td>GET, PUT</td>
</tr>
<tr>
<td>Project</td>
<td>Project</td>
<td>application/vnd.ibm.wcm+xml</td>
<td>GET, PUT</td>
</tr>
<tr>
<td>Rich Text Component</td>
<td>LibraryRichTextComponent</td>
<td>text/html</td>
<td>GET, PUT</td>
</tr>
<tr>
<td>Item Description</td>
<td>Item Type</td>
<td>Supported Media Types</td>
<td>Supported Methods</td>
</tr>
<tr>
<td>------------------------</td>
<td>----------------------</td>
<td>-----------------------</td>
<td>-------------------</td>
</tr>
<tr>
<td>Rich Text Element</td>
<td>RichTextComponent</td>
<td>text/html</td>
<td>GET, PUT</td>
</tr>
<tr>
<td>Short Text Component</td>
<td>LibraryShortTextComponent</td>
<td>text/html</td>
<td>GET, PUT</td>
</tr>
<tr>
<td>Short Text Element</td>
<td>ShortTextComponent</td>
<td>text/html</td>
<td>GET, PUT</td>
</tr>
<tr>
<td>Site Area Template</td>
<td>SiteAreaTemplate</td>
<td>application/vnd.ibm.wcm+xml</td>
<td>GET, PUT</td>
</tr>
<tr>
<td>Site Area</td>
<td>SiteArea</td>
<td>application/vnd.ibm.wcm+xml</td>
<td>GET</td>
</tr>
<tr>
<td>Stylesheet Component</td>
<td>LibraryStyleSheetComponent</td>
<td>text/css</td>
<td>GET, PUT</td>
</tr>
<tr>
<td>Text Component</td>
<td>LibraryTextComponent</td>
<td>text/html</td>
<td>GET, PUT</td>
</tr>
<tr>
<td>Text Element</td>
<td>TextComponent</td>
<td>text/html</td>
<td>GET, PUT</td>
</tr>
</tbody>
</table>

### Displaying data from external sources

You display data from external sources using the same methods as you would when creating a website.

#### Displaying data

You can display content from external sources using standard Java tag libraries and a JSP element. Java code using standard Java APIs or tag libraries can be used to display and format data from databases, LDAP repositories, or send email.

If using a rendering portlet to display web content on a portal page, you can also use other IBM WebSphere Portal portlets on the same portal page to display data.

#### Web page aggregation

Content from external websites and IBM Web Content Manager can be displayed together on a portal page by using standard WebSphere Portal portlets for displaying content from external websites. Please refer to the web Page portlet and web clipping portlet sections of this Information Center for information about creating and configuring these portlets.

This functionality is only available when displaying content in WebSphere Portal. These portlets are not accessible from the Web Content Manager servlet, though standard Java API or tag libraries may be used with a JSP element to achieve the same result.

### Creating websites for different localities

Although websites that automatically display content for different localities or languages are not directly supported by IBM Web Content Manager, it is possible to maintain separate libraries within Web Content Manager for different localities.
Creating libraries and search collections for different localities

When creating a multi-locale solution, it is recommended that you do the following:
- Create a separate library for each locality.
- Create another library for any content that will be shared across localities.
- Create a separate search collection for each locality.

**Note:** If a language does not exist in the list of languages available when creating a library, you can add that language to the list of supported Web Content Manager languages. See "Language Support" in the Web Content Manager information center for further information.

**Naming consistency**

It is recommended that the same names are given to corresponding items in each library. This will simplify the navigation between localities as only the library name will be different in URLs pointing to the same content in different localities.

For example, only the library name will be different for content displayed in French and German localities:
- `http://host/wps/wcm/connect/frenchlibrary/sitearea/content`
- `http://host/wps/wcm/connect/germanlibrary/sitearea/content`

**Maintaining content in different localities**

At present, Web Content Manager does not manage the synchronization of content between localities.

A workflow can be used to assist with the synchronization of content between localities by using the joint approval feature. You can add a stage to a workflow that would require joint approval from a member in a group representing each locality. You would configure this stage to send an email to notify each joint-approver that an item has been updated. Each joint-approver would not approve the item until they had updated the same item in their locality to match the changes in the primary locality.

**Enabling Java messaging services for web content**

Web Content Manager provides support for the notification of events such as item state changes, or services starting and stopping. These notifications can be delivered as messages to the Java messaging service.

There are three classes of events that can be delivered as messages to the Java messaging service:

**Item events:**
- Item created
- Item updated
- Item moved
- Item deleted

**Syndication events:**
- Starting
To enable Java messaging services for web content:

1. Configure the messaging services parameters in the WCM MessagingService service using the WebSphere Integrated Solutions Console.

2. Run the following command from the \wp_profile_root\ConfigEngine directory:

   **Windows**
   ```
   ConfigEngine.bat create-wcm-jms-resources
   -DPortalAdminId=username -DPortalAdminPwd=password
   -DuseRemoteEndPoints=true/false
   ```

   **UNIX**
   ```
   ./ConfigEngine.sh create-wcm-jms-resources
   -DPortalAdminId=username -DPortalAdminPwd=password
   -DuseRemoteEndPoints=true/false
   ```

   **IBM i**
   ```
   ConfigEngine.sh create-wcm-jms-resources -DPortalAdminId=username
   -DPortalAdminPwd=password -DuseRemoteEndPoints=true/false
   ```

   **z/OS**
   ```
   ./ConfigEngine.sh create-wcm-jms-resources
   -DPortalAdminId=username -DPortalAdminPwd=password
   -DuseRemoteEndPoints=true/false
   ```

   **Note:** An administrator user name and password is not required if you have already specified the portal administrator username and password using the PortalAdminId and PortalAdminPwd settings in the wkplc.properties file.

   **Note:** The "-DuseRemoteEndPoints" parameter is only used on clustered systems. If set to true, the task will use all node end points on the current setup. If set to "false", the task will use the end points of the current node.

3. Restart WebSphere Portal.

**Related tasks:**
- Setting service configuration properties

IBM WebSphere Portal comprises a framework of services to accommodate the different scenarios that portals need to address. Services are available for both WebSphere Portal and IBM Web Content Manager. You can configure some of these services.

**Related reference:**
- Web Content Manager configuration services

Configuration services for IBM Web Content Manager contain settings for the general operation of the web content system, including settings for messaging, pre-rendering, and searching.
Chapter 9. Administering syndication

Learn how to administer syndication relationships.

Syndication overview

Syndication is the method used by IBM Web Content Manager to replicate data from a web content library on one server to a web content library on another server.

To enable syndication, a syndicator and a subscriber must be defined:
• The syndicator defines a connection to the subscriber and indicates which libraries are to be replicated to the subscriber.
• The subscriber defines a connection to the syndicator and receives the data replicated from the libraries specified by the syndicator.

The relationship between a syndicator and a subscriber can be either a one-way or two-way relationship.

Example: One-way syndication
Application 1 syndicates one or more libraries to Application 2, and Application 2 subscribes from Application 1.
Example: Two-way syndication

Both applications syndicate to each other.
Both applications subscribe from each other.

Note:
- When enabling two-way syndication, you must first establish the syndication relationship from Application 1 to Application 2. Once the libraries have been replicated to Application 2, you can set up the syndication relationship between Application 2 and Application 1.
- Although it is possible to set up more than one syndication relationship between the same two applications, there is no reason to do so. The additional syndication relationships are not required because once a syndication relationship has been established between two applications, no further relationships are established.

Syndicators can syndicate libraries to multiple subscribers, and subscribers can subscribe to libraries from multiple syndicators.

Example: Multiple syndication relationships

Both Application 1 and 2 syndicate to Application 3.
Application 3 subscribes from both Application 1 and 2.

Syndication methods

There are three syndication methods available when configuring a syndication relationship:

Live items:
Live item syndication is mostly used when syndicating to a staging or delivery server. The following items are syndicated:
Draft items, projects and items in a project are not syndicated.

**Live and projects:**
The advantage of using "Live and projects" syndication is to gradually syndicate projects to a staging or delivery server rather than waiting to syndicate all the items in a project after they all achieve a published state.
The following items are syndicated:
- Published
- Expired
- Projects
- Draft items in a project
Draft items outside of projects are not syndicated.

**All items:**
All item syndication is mostly used when syndicating between servers within an authoring environment. The following items are syndicated:
- Published
- Expired
- Projects
- Draft items in a project
- Other draft items
- Versions
- Deleted items

**Switching from "all item" syndication to "live item" syndication:** When you switch from "all item" syndication to "live and projects" syndication or "live item" syndication, any drafts previously syndicated to the subscriber are not removed.

**Moving draft items between libraries:** If you move a draft item from a library using "all item" syndication to a library using "live item" syndication, the draft item will also be moved on the subscriber because the action occurred on the library using "all item" syndication. This behavior allows for some draft items to be included in a subscriber library even though "live item" syndication is being used.

**Release and version consistency**
All servers participating in syndication must be of the same release for syndication to work as expected and to be in a supported state. Syndication is supported between different versions of the same release. Syndication is not supported between different releases.

For example:
- Server A with version 8.0.0.0 can be syndicated to Server B with version 8.0.0.1 installed.
- Server A with version 8.0.0.0 can be syndicated to Server B with version 8.0.1 installed.
- Server A with version 8.0.1 can be syndicated to Server B with version 8.0.0.0 installed.
- Syndicating Server A with version 6.1.5 to Server B with version 8.0.0.0 installed is not supported.
- Syndicating Server A with version 8.0.0.0 to Server B with version 7.0.0.1 installed is not supported.

It is also recommended to keep your environments consistent across all servers in your system. For example, the same version of WebSphere Application Server must be used on all servers in your system to ensure successful syndication.

**Web content libraries and syndication relationships**

All the items you work with as part of your Web Content Manager authoring environment (templates, components, content items, and so on) are stored in web content libraries. When you syndicate data between applications, you do so on a library by library basis. As part of the definition of a syndicator or subscriber, you specify which web content libraries are to be included during syndication.

Because syndication is performed by library, it is important to consider how to organize your content between libraries to support your Web Content Manager environment. For example, suppose you are using a single authoring server to develop content for two delivery servers, an intranet site providing Human Resources information intended for internal employees of a company and an external Internet site providing marketing material intended for customers and others outside the company. A basic approach to support this environment would be to use two web content libraries, one for content specific to each site. You would then set up two syndication relationships with each going from the authoring server to the appropriate delivery server.

For easier management, you might divide your content further into three libraries, where one library contains data common to both the intranet and Internet sites and the other two libraries contain site-specific content. The following example demonstrates this configuration, with the addition of two other authoring portlets so that the content of each library is maintained by a different authoring portlet.
In this case you might set up several syndication relationships between the authoring server and the delivery servers:

- The Common Library syndicates to the intranet site (Human Resources Portal).
- The Common Library syndicates to the Internet site (Marketing Portal).
- The HR Library syndicates to the intranet site (Human Resources Portal).
- The Marketing Library syndicates to the Internet site (Marketing Portal).

**Note:** Web Content Manager provides flexibility in how you set up your syndication relationships. If you need to syndicate multiple libraries from one server to another, you can choose to use one syndication relationship that includes all the libraries, or you can choose to use separate syndication relationships for each library, or even a combination of both approaches, depending on how many libraries you are syndicating. The best approach for your situation depends not only on how many libraries are involved but also on how the libraries are related to one another. For example, you use a single syndication relationship for libraries that reference each other, as when one library contains design items like templates that are used by content in the other library. However, if the libraries are independent of one another and you think you might want to suspend syndication of one library but not the other, separate syndication relationships for each library can provide that.

**Important:**

- First-time syndication to an existing library is not supported. If you attempt to syndicate a library to a subscriber that already has a library with the same name, an error results.
- Information about a Library is only syndicated the first time syndication occurs and not on subsequent updates and rebuilds. If a library is renamed or library user access is changed, this information is not syndicated to the Subscriber. If you change the name of a library or change user access to a library, you must manually make the same changes to any subscriber libraries if you want the same settings on all your syndicated libraries.
- If content from one library (Library A) uses an item from another library (Library B), you must include both libraries in the syndicator to ensure that all items are syndicated successfully. If you only include Library A in the syndicator, any items in Library A that reference items in Library B are not syndicated, and syndication errors are generated.
- If you add a new library to a syndicator after the initial syndication, you click Update to force the new library to be syndicated immediately.

**Access control when syndicating**

Although syndication can be used to keep data current between libraries on different servers, access control settings for the libraries are not included as part of syndication. Depending on how your environment is set up and what policies you have in place for library access, there are additional considerations for access control when using syndication.

**User consistency**

For user level access to remain consistent between the syndicator and subscriber, both servers must be configured to use the same user repository. If different user repositories are used, syndication will occur but there will be errors in the subscriber log indicating missing users. If access controls are determined by only using virtual users and groups, such as
"All authenticated" and "Anonymous Users", then there is no need to use the same user repository on the syndicator and subscriber.

First time syndication on a new library
Because library access control settings are not syndicated, you must manually set access permissions on the subscriber’s library when syndicating for the first time. If the library does not exist on the subscriber, it will be created during syndication. By default, no access control settings are specified on the new library, so you must set them manually before users can access content in the new library. The settings on the subscriber library do not have to match those on the syndicator library. This allows you to specify different levels of access for users and groups on the subscriber.

Creating a syndication relationship
To set up syndication between libraries on two IBM Web Content Manager applications, establish a relationship between a syndicator and a subscriber. The syndicating server contains the data to be replicated, and the subscribing server receives the replicated data.

Important:
- To syndicate a library that contains more than 10000 items, update the maximum Java heap size used by the portal application server on the subscriber server:
  1. In the WebSphere Integrated Solutions Console, navigate to the Java virtual machine settings.
  2. Stand-alone server:
     - Servers > Server Types > WebSphere application servers >
     - WebSphere Portal > Java and Process Management > Process definition > Java Virtual Machine
  3. Clustered server:
     - System administration > Deployment manager > Java and Process Management > Process Definition > Java Virtual Machine
  2. Update the value in the Maximum Heap Size field. A value of at least 1024 MB is recommended.
  3. Click OK, and then save your changes.
- You can syndicate only between servers running the same version. You cannot syndicate between versions. For example:
  - You can syndicate between versions 8.0.0.0 and 8.0.0.1.
  - You cannot syndicate between versions 7.0.0.0 and 8.0.0.0.
- Ensure that you have at least as much swap space allocated on the subscriber server as you have physical memory.
- First-time syndication to an existing library is not supported. If you attempt to syndicate a library to a subscriber that already has a library with the same name, an error results.
- Information about a Library is only syndicated the first time syndication occurs and not on subsequent updates and rebuilds. If a library is renamed or library user access is changed, this information is not syndicated to the Subscriber.
- If you change the name of a library or change user access to a library, these changes are not syndicated. If you want the same settings on all your syndicated libraries, you must manually make the same changes to any subscriber libraries.
• If content from one library (Library A) uses an item from another library (Library B), you must include both libraries in the syndicator. Including both libraries ensures that all items are syndicated successfully.
• If you include only Library A in the syndicator, any items in Library A that reference items in Library B are not syndicated. Syndication errors are also generated.
• If you add a library to a syndicator after the initial syndication, you must click Update to force the new library to be syndicated immediately.
• If you are creating a two-way syndication relationship, you must use a consistent syndication strategy. For example, if syndicating "All items", then both syndication relationships must be syndicating "All items".

To set up a syndication relationship, complete the following steps:
1. Ensure both the subscriber and syndicator are running and that they can access each other over a network.
2. On your subscriber server, log in to IBM WebSphere Portal.
3. Go to Administration > Access > Credential Vault and create a shared credential vault slot to allow you to access the syndicator.
4. Go to Administration > Portal Content > Subscribers.
5. Click Subscribe Now.
6. Enter the syndicator URL in the form of http://HostName:HostPort/WcmContextRoot. For example: http://authoring:10039/wps/wcm

Note:
When syndicating to a virtual portal:

Using the URL context of a virtual portal:
http://HostName:HostPort/wps/wcm/url_context

Using the hostname of a virtual portal:
http://VirtualHostName:HostPort/wps/wcm

7. Enter a name for the syndicator item. This name is used for the syndicator item that is created on the syndicator server. Enter a name that helps identify the syndication relationship you are creating. This name must be unique and cannot be the same as an existing syndicator name.

Note: To reuse syndicator names of previously deleted syndication relationships on a subscriber, you must also delete the same relationship on the syndicator.

8. Enter a name for the subscriber item. This name is used for the subscriber item that is created on the subscriber server. Enter a name that helps identify the syndication relationship you are creating. This name must be unique and cannot be the same as an existing subscriber name.

9. Select the credential vault slot you created previously.
10. Click Next
11. Select the libraries you want to subscribe to.
   a. Click Add Libraries to select a library to syndicate and the syndication type:

   Live items:
   Live item syndication is mostly used when syndicating to a staging or delivery server. The following items are syndicated:
   • Published
Live and projects:
Use "Live and projects" syndication to gradually syndicate projects to a staging or delivery server, rather than waiting until all items in a project achieve a published state. The following items are syndicated:
- Published
- Expired
- Projects
- Draft items in a project
Draft items outside of projects are not syndicated.

All items:
All item syndication is mostly used when syndicating between servers within an authoring environment. The following items are syndicated:
- Published
- Expired
- Projects
- Draft items in a project
- Other draft items
- Versions
- Deleted items

b. Select libraries from the current list, and click Remove Libraries to remove a library from the list of syndicated libraries.

12. Click Finish.
13. To begin syndication, click either Update Subscriber or Rebuild Subscriber button.

Creating a syndication relationship from the command line
To set up a syndication relationship from the command line, use the XML configuration interface (xmlaccess command) and the ConfigEngine command to configure the subscriber.

Important:
- To syndicate a library that contains more than 10000 items, update the maximum Java heap size used by the portal application server on the subscriber server:
  1. In the WebSphere Integrated Solutions Console, navigate to the Java virtual machine settings.
  Stand-alone server:
    - Servers > Server Types > WebSphere application servers > WebSphere_Portal > Java and Process Management > Process definition > Java Virtual Machine
  Clustered server:
    - System administration > Deployment manager > Java and Process Management > Process Definition > Java Virtual Machine
  2. Update the value in the Maximum Heap Size field. A value of at least 1024 MB is recommended.
3. Click OK, and then save your changes.

- You can syndicate only between servers running the same version. You cannot syndicate between versions. For example:
  - You can syndicate between versions 8.0.0.0 and 8.0.0.1.
  - You cannot syndicate between versions 7.0.0.0 and 8.0.0.0.
- Ensure that you have at least as much swap space allocated on the subscriber server as you have physical memory.
- First-time syndication to an existing library is not supported. If you attempt to syndicate a library to a subscriber that already has a library with the same name, an error results.
- Information about a library is only syndicated the first time syndication occurs and not on subsequent updates and rebuilds. If a library is renamed or library user access is changed, this information is not syndicated to the subscriber.
- If you change the name of a library or change user access to a library, these changes are not syndicated. If you want the same settings on all your syndicated libraries, you must manually make the same changes to any subscriber libraries.
- If content from one library (Library A) uses an item from another library (Library B), you must include both libraries in the syndicator. Including both libraries ensures that all items are syndicated successfully.
- If you include only Library A in the syndicator, any items in Library A that reference items in Library B are not syndicated. Syndication errors are also generated.
- If you add a library to a syndicator after the initial syndication, you must click Update to force the new library to be syndicated immediately.
- If you are creating a two-way syndication relationship, you must use a consistent syndication strategy. For example, if syndicating "All items", then both syndication relationships must be syndicating "All items".

To set up a syndication relationship from the command line, complete the following steps:

1. Ensure both the subscriber and syndicator are running and that they can access each other over a network.

2. On the subscriber server, create a shared credential vault slot with the XML configuration interface.
   a. Create the CreateVaultSlot.xml file using a text editor.
      This sample file uses the following values that you can change to reflect your environment:

      `<syndication-slot>
      The name of the shared credential vault slot.
      </syndication-slot>

      `<wpsadmin>
      The user ID for the portal administrator.
      </wpsadmin>

      `<password>
      The password for the portal administrator.
      </password>`
b. Run the **xmlaccess** command with the CreateVaultSlot.xml file.

**AIX Linux Solaris z/OS**

```
./xmlaccess.sh -in CreateVaultSlot.xml -out slot-out.xml -url http://localhost:10039/wps/config -user wpsadmin -password passw0rd
```

**IBM i**

```
xmaccess.sh -in CreateVaultSlot.xml -out slot-out.xml -url http://localhost:10039/wps/config -user wpsadmin -password passw0rd
```

**Windows**

```
xmaccess.bat -in CreateVaultSlot.xml -out slot-out.xml -url http://localhost:10039/wps/config -user wpsadmin -password passw0rd
```

3. On the subscriber server, create a virtual portal on the subscriber with the create-virtual-portal task.

This example creates a virtual portal called *sample*, although you can change this value to reflect your environment.

**AIX Linux Solaris z/OS**

```
./ConfigEngine.sh create-virtual-portal -DPortalAdminPwd=passw0rd -DWasPassword=passw0rd -DVirtualPortalTitle=sample -DVirtualPortalContext=sample
```

**IBM i**

```
ConfigEngine.sh create-virtual-portal -DPortalAdminPwd=passw0rd -DWasPassword=passw0rd -DVirtualPortalTitle=sample -DVirtualPortalContext=sample
```

**Windows**

```
ConfigEngine.bat create-virtual-portal -DPortalAdminPwd=passw0rd -DWasPassword=passw0rd -DVirtualPortalTitle=sample -DVirtualPortalContext=sample
```

4. On the subscriber server, set up the syndication relationship with the run-wcm-admin-task-subscribe-now task.

This sample command uses the following additional values that you can change to reflect your environment:

- **syndicator-hostname**
  - The host name of the syndicator server.

- **syndicator1**
  - This name is used for the syndicator item that is created on the syndicator server. Enter a name that helps identify the syndication relationship you are creating. This name must be unique and cannot be the same as an existing syndicator name.

**Note:** To reuse syndicator names of previously deleted syndication relationships on a subscriber, you must also delete the same relationship on the syndicator.

- **subscriber1**
  - This name is used for the subscriber item that is created on the
subscriber server. Enter a name that helps identify the syndication relationship you are creating. This name must be unique and cannot be the same as an existing subscriber name.

In addition, use the following properties to identify the libraries to which you are subscribing and the type of syndication that you want to perform. For each syndication relationship, you can specify only one type of syndication. Separate multiple libraries with commas.

**liveItems="library_name_1,library_name_2"**

Live item syndication is mostly used when syndicating to a staging or delivery server. The following items are syndicated:

- Published
- Expired

Draft items, projects, and items in a project are not syndicated.

**liveProjectsItems="library_name_1,library_name_2"**

Use "Live and projects" syndication to gradually syndicate projects to a staging or delivery server, rather than waiting until all items in a project achieve a published state. The following items are syndicated:

- Published
- Expired
- Projects
- Draft items in a project

Draft items outside of projects are not syndicated.

**allItems="library_name_1,library_name_2"**

All item syndication is mostly used when syndicating between servers within an authoring environment. The following items are syndicated:

- Published
- Expired
- Projects
- Draft items in a project
- Other draft items
- Versions
- Deleted items

Example commands:

**AIX Linux Solaris z/OS**

```
./ConfigEngine.sh run-wcm-admin-task-subscribe-now
-Dsyndicator=http://syndicator-hostname:10039/wps/wcm
-DvaultSlotName=syndication-slot -DsyndicatorName=syndicator1
-DsubscriberName=subscriber1 -DVirtualPortalContext=sample
-DliveItems="Web Content,Portal Site" -DPortalAdminPwd=passw0rd
-DWasPassword=passw0rd
```

**IBM i**

```
ConfigEngine.sh run-wcm-admin-task-subscribe-now
-Dsyndicator=http://syndicator-hostname:10039/wps/wcm
-DvaultSlotName=syndication-slot -DsyndicatorName=syndicator1
-DsubscriberName=subscriber1 -DVirtualPortalContext=sample
-DliveItems="Web Content,Portal Site" -DPortalAdminPwd=passw0rd
-DWasPassword=passw0rd
```

**Windows**

```
ConfigEngine.bat run-wcm-admin-task-subscribe-now
```
Syndication tuning

While syndication is a vital part of keeping your web content current, the same syndication strategy is not appropriate for every environment. Depending on how you have deployed IBM Web Content Manager, you can use different syndication strategies to balance the currency of your content with the performance your environment requires.

There are several different means to manage how and when content is replicated to other servers. It is important to note that, as with any process, syndication entails a performance cost, and this should be taken into account when specifying not only the frequency of syndication but also the number of syndication relationships for any given server.

Syndication interval

The syndication interval controls the frequency of syndication while automatic syndication is enabled and can be used by administrators to put a cap on how often syndication occurs. Because up-to-date information is important to any web content environment, automatic syndication is enabled by default when IBM WebSphere Portal is installed. While the default setting for the syndication interval ensures maximum currency, you can choose to adjust the value accordingly if the currency demands of your environment do not call for the shortest interval.

Here are some general guidelines for setting the syndication interval.

<table>
<thead>
<tr>
<th>Interval setting</th>
<th>Recommended environments</th>
</tr>
</thead>
<tbody>
<tr>
<td>10 minutes – 2 hours</td>
<td>Staging servers to delivery servers.</td>
</tr>
<tr>
<td>30 seconds – 10 minutes</td>
<td>Any environment that requires frequent replication, such as an authoring server to a staging server, a test server, or distributed authoring server.</td>
</tr>
<tr>
<td></td>
<td>When increasing the syndication interval for environments where authoring servers are involved, be mindful that timely replication is often essential, particularly in collaborative authoring environments where multiple authors on different servers might be working on the same content.</td>
</tr>
</tbody>
</table>
Syndication types

Live items:
Live item syndication is mostly used when syndicating to a staging or delivery server. The following items are syndicated:
- Published
- Expired
Draft items, projects and items in a project are not syndicated.

Live and projects:
The advantage of using "Live and projects" syndication is to gradually syndicate projects to a staging or delivery server rather than waiting to syndicate all the items in a project after they all achieve a published state.
The following items are syndicated:
- Published
- Expired
- Projects
- Draft items in a project
Draft items outside of projects are not syndicated.

All items:
All item syndication is mostly used when syndicating between servers within an authoring environment. The following items are syndicated:
- Published
- Expired
- Projects
- Draft items in a project
- Other draft items
- Versions
- Deleted items

Manual syndication
It is not necessary to wait for the syndication interval to elapse in order to perform syndication. Manual syndication can be used at any time to cause syndication to occur immediately. There are different ways to take advantage of manual syndication:
- You can use manual syndication in conjunction with the syndication interval to provide flexibility. For example, if you are using an increased syndication interval to optimize performance between servers, you can still perform manual syndication when you need to update content without waiting for the next syndication.
- If you require complete control of when syndication occurs, you can use manual syndication as the only means of performing syndication. For this approach, you must disable automatic syndication so that the syndication interval setting is ignored.

Publish and expire dates
Because the syndication interval is not based on a scheduled time of the day, it is not suitable for setting up syndication to run during off-peak times. The publish date and expire date are the preferred methods for doing this.
The publish date for each content item specifies the date and time when the content should be published to a website. After a content item is approved and the publish date is reached, the content item is queued for syndication according to the next syndication interval. By using the publish date, you can delay the publishing of content, effectively delaying its syndication. For example, if you are using a short syndication interval to ensure timely replication of most content, you can delay the syndication of less urgent content through the publish date for that content.

The expire date specifies the date and time when the content item should be removed from a website. As with the publish date, you can use the expire date to cause the syndication activity associated with removing the content to occur at a time when other syndication activity is less intensive.

For more information on the publish and expire dates in the workflow process, refer to “Item status” on page 135.

**Syndication troubleshooting**

If you encounter issues when syndicating, there are some common methods available to troubleshoot these issues.

**Common issues**

*Table 194. Common issues*

<table>
<thead>
<tr>
<th>Issue</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unable to reach host</td>
<td>This is a common reason why syndication does not work. The URL for the syndicator or the subscriber may not be valid. You may need to use the IP address rather than the domain name.</td>
</tr>
<tr>
<td>Syndicator becomes unresponsive during syndication</td>
<td>Syndication can require a large amount of resources to run successfully. Consequently, if your server is performing other tasks at the same time as syndication, the process of syndication may slow or stop altogether. You should schedule your syndication to occur at times when the server load is at its lowest.</td>
</tr>
<tr>
<td>Syndicator status hangs on &quot;Pending&quot;, or &quot;Pending, Active&quot;</td>
<td>If you are attempting to update or rebuild a syndicated library containing large number of items, the syndicator status might hang on 'Pending', or 'Pending, Active'. This can occur because the syndicator keeps retrying to syndicate when some items fail to syndicate to the subscriber, or when a system timeout occurs on the subscriber when saving data. Improving the performance of your database can help avoid these situations. For example, two of the database attributes that DB2 relies upon to perform optimally are the database catalog statistics and the physical organization of the data in the tables. Catalog statistics should be recomputed periodically during the life of the database, particularly after periods of heavy data modifications (inserts, updates, and deletes) such as a population phase. In order to fix this, you should run &quot;Runstats&quot; on the JCR database before and after syndication. The DB2 runstats command is used to count and record the statistical details about tables, indexes and columns. See Database performance for information on using the &quot;Runstats&quot; task. Due to the heavy load of computing these statistics, it is recommend that this maintenance occurs during off hours, periods of low demand, or when the portal is off-line.</td>
</tr>
</tbody>
</table>
### Table 194. Common issues (continued)

<table>
<thead>
<tr>
<th>Issue</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time-outs during syndication</td>
<td>Time-outs during syndication are often caused by the failure of large items to be saved. Increasing the total transaction lifetime timeout setting of your IBM WebSphere Portal server can address this issue. The total transaction lifetime timeout setting of your subscriber should be at least the same as the syndicator. The total transaction lifetime timeout setting is changed using the WebSphere Integrated Solutions Console. Go to Servers &gt; Server Types &gt; WebSphere application servers &gt; portal_server-&gt;Container Services &gt; Transaction Service. See the WebSphere Application Server information center for more information.</td>
</tr>
<tr>
<td>Subscriber becomes unresponsive during syndication</td>
<td>If you are attempting to syndicate a library containing more than 10000 items, the subscriber machine might become unresponsive during the syndication operation. This can occur due to an insufficient Java heap size setting on the subscriber. To update the maximum Java heap size used by the portal application server on the subscriber machine, complete the following steps: 1. In the WebSphere Integrated Solutions Console, click System administration &gt; Deployment manager &gt; Java and Process Management &gt; Process Definition &gt; Java Virtual Machine. 2. Update the value in the Maximum Heap Size field. A value of at least 1024 MB is recommended. 3. Click OK, and then save your changes. 1. In the WebSphere Integrated Solutions Console, select Servers &gt; Application Servers &gt; yourPortalServerName &gt; Java and Process Management &gt; Process Definition &gt; Servant &gt; Java Virtual Machine &gt; Maximum Heap Size to set the JVM heap size. 2. Set the value to a maximum of 768 MB. 3. Click OK, and then save your changes. In addition, ensure that you have at least as much swap space allocated on the subscriber machine as you have physical memory.</td>
</tr>
<tr>
<td>Syndication takes a long time to complete on z/OS systems</td>
<td>When running syndication with a large amount of data, the z/OS database 4k buffer pool, 4k index buffer pool and the 32k buffer pool will need to monitored and increased as necessary. Increase of these buffer pools will make significant impact on syndication performance. See Preparing DB2 for z/OS for further information.</td>
</tr>
</tbody>
</table>

### Other solutions

### Table 195. Other solutions

<table>
<thead>
<tr>
<th>Option</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resetting the web content event log.</td>
<td>To assist in the troubleshooting process you can reset the web content event log. See “Resetting the web content event log” on page 397 for further information.</td>
</tr>
</tbody>
</table>
### Working with failed items

From time to time items will fail to syndicate. You use the failed items view to review a list of failed items and then run syndication again once you have fixed the issue.

1. Logon to your syndicator as an administrator.
2. Go to **Administration>Portal Content>Syndicators**.
3. The number of failed items for a syndicator are displayed in the Failed Items column. Click on the number of failed items to open the Failed Items view.
4. Each failed item for the selected syndicator is displayed in the Failed Items view. The reason for the failure is displayed against each failed item along with a message catalog code. Look up the code in the message catalog and take the appropriate action to fix the issue. You can then try to syndicate the failed item again by rebuilding the syndication relationship. Updating the syndication relationship will not retry failed items.

**Related tasks:**

- [Setting service configuration properties](#)

IBM WebSphere Portal comprises a framework of services to accommodate the different scenarios that portals need to address. Services are available for both WebSphere Portal and IBM Web Content Manager. You can configure some of these services.

---

**Table 195. Other solutions (continued)**

<table>
<thead>
<tr>
<th>Option</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enabling the &quot;deployment.enableReport&quot; setting on the subscriber.</td>
<td>You can update the WCM WCMConfigService service and specify the <code>deployment.enableReport</code> property with a value of true. This enables high level reporting of syndication to the SystemOut log on the subscriber server. It provides a summary of items that were processed, and which items failed syndication to help you troubleshoot syndication issues. You can update the WCM WCMConfigService service and specify the <code>deployment.enableReport</code> property with a value of true. This enables high level reporting of syndication to the portal application server job log on the subscriber server. It provides a summary of items that were processed, and which items failed syndication to help you troubleshoot syndication issues.</td>
</tr>
</tbody>
</table>
Chapter 10. Troubleshooting

To help you resolve problems, use diagnostic tools such as IBM Support Assistant and tracing to capture system errors.

Tools for troubleshooting and diagnostics

A number of tools and resources are available to help you troubleshoot issues and resolve problems that users might encounter while using IBM WebSphere Portal. If you need further assistance, you can use the tools described here to identify and collect information to help IBM Support determine the underlying cause of a problem.

IBM Support Assistant

IBM Support Assistant (ISA) provides quick access to product, education, and support resources that can help you answer questions and resolve problems with IBM software products on your own, without needing to contact IBM Support. Different product-specific plug-ins let you customize IBM Support Assistant for the particular products you have installed. IBM Support Assistant can also collect system data, log files, and other information to help IBM Support determine the cause of a particular problem.

IBM Support Assistant is a utility to be installed on an administrator’s workstation, not directly onto the WebSphere Portal or Web Content Manager server system itself. The memory and resource requirements for the Assistant could negatively impact the server’s performance. The included portable diagnostic components are designed for minimal impact to the normal operation of a server.

You can use IBM Support Assistant to help you in the following ways:

- To search through IBM and non-IBM knowledge and information sources across multiple IBM products to answer a question or solve a problem
- To find additional information through product-specific Web resources; including product and support home pages, customer news groups and forums, skills and training resources and information about troubleshooting and commonly asked questions
- To extend your ability to diagnose product-specific problems with targeted diagnostic tools available via the Support Assistant
- To simplify collection of diagnostic data to help you and IBM resolve your problems (collecting either general or product/symptom-specific data)
- To help in reporting of problem incidents to IBM Support through a customized online interface, including the ability to attach the previous diagnostic data or any other information to new or existing incidents
Finally, you can use the built-in Updater facility to obtain support for additional software products and capabilities as they become available.

More information, the download package, installation instructions and the latest version of the IBM Support Assistant are available from the IBM Support Assistant Web page at www.ibm.com/software/support/isa/.

Data collection and symptom analysis

There are two methods for collecting data and analyzing symptoms for problem determination scenarios. The first method is IBM Support Assistant Lite for WebSphere Portal, which provides automatic data collection and symptom analysis support for problem determination scenarios. This tool collects and analyzes information pertinent to a problem scenario to help identify the origin of the problem being encountered. The second method is running a task that can collect and optionally send the data for you.

IBM Support Assistant Lite reduces the amount of time it takes to reproduce a problem with the proper Reliability, Availability, and Serviceability (RAS) tracing levels set (trace levels are set automatically by the tool) and performs symptom analysis to streamline problem determination. If you need further assistance, IBM Support Assistant Lite also reduces the effort required to send the appropriate log information to IBM Support.

IBM Support Assistant Lite can also create and deploy aspect-enabled JAR files to enable additional troubleshooting. After reproducing the problem, you can use IBM Support Assistant Lite to remove the aspect-enabled JAR files, replace them with the original version, and collect the LOG file. For detailed instructions, see the User Guide that comes with IBM Support Assistant Lite.

If you do not want to install IBM Support Assistant Lite, you can implement an aspect-enabled jar file manually.

Choose one of the following methods to collect data:

IBM Support Assistant Lite

Perform the following steps:

1. Download IBM Support Assistant Lite for WebSphere Portal.
2. Extract the .zip file anywhere on the portal server.
3. Go to the ISALite directory that was created when you completed the previous step.
4. To collect data, run the following script:
   Windows GUI: runISALite.bat
   AIXSolarisLinux GUI: ./runISALite.sh
   Windows command line: runISALiteConsole.bat
   AIXSolarisLinux command line: ./runISALiteConsole.sh
5. Select the Problem Type. For example, for a portal logon problem, select WebSphere Portal > Security and Administration > Portal Login Problem.
6. Specify the collection file. The file name must include the path and .zip extension.
7. Click Collect Data and answer the remaining prompts.
Tip: To have the IBM Support Assistant Lite automatically FTP the logs to IBM so that a manual transfer is not necessary, simply choose the required FTP option when prompted.

8. If you did not automatically FTP your results, locate the collection file specified in step 6. To manually FTP your results, follow the instructions in Exchanging information with IBM Technical Support for problem determination.

For detailed instructions, see the IBM Support Assistant Lite for WebSphere Portal tool User's Guide.

wpcollector tool collect diagnostics task

Perform the following steps:

1. If the support team has requested tracing, enable it now as instructed and then re-create the problem. If no tracing is requested, skip to the next step.

2. Open a command prompt and change to the `wp_profile_root/PoralServer/bin/` directory.

   Attention: You must run the `wpcollector` task from the `wp_profile_root/PoralServer/bin/` directory. If you run the task from a different directory, the task fails.

3. Run the following script to collect data:
   - Windows: `wpcollector.bat`
   - AIXSolarisLinux: `./wpcollector.sh`
   - IBM: `wpcollector.sh`

   Tip: The `wpcollector` script can automatically FTP the logs to IBM so that you don't need to manually transfer them. To begin the collection and FTP the results to IBM correctly, use the `-Dpmr=pmr_number` parameter to identify the collection with your PMR (Problem Management Record) number. Format the number using either periods or commas. For example: `wpcollector.bat -Dpmr=12345.xxx.000`

   To collect files for the Deployment Manager profile, use the `-Ddmgr.root=dmgr_root` parameter either alone or in conjunction with `-Dpmr=pmr_number`.

4. If you did not automatically FTP your results, locate the `wp.mustgather.zip` file or the `pmr-wp.mustgather-timestamp.zip` file in the `wp_profile_root/filesForAutoPD/` directory. Follow the instructions in "Exchanging information with IBM Technical Support for problem determination" to manually FTP your results.

Restriction: If you try to extract the `wp.mustgather.zip` file, some collections might not expand properly if the path name exceeds the 256 character limitation.

Manual creation of aspect-enabled JAR files

IBM Support Assistant Lite is the preferred way to create and deploy aspect-enabled JAR files for troubleshooting because the tool automates the process for you. If you do not want to install IBM Support Assistant Lite, or you are running on a platform that does not support IBM Support Assistant Lite, you can manually create your own aspect-enabled JAR files using either the aspect source files that ship with IBM WebSphere Portal, or aspect source files provided by IBM Support.

1. Set up AspectJ as follows:
a. Download the latest version of AspectJ from the Eclipse Web site.
b. To install AspectJ, run the following command:
   ```
   java -jar aspect_enabled_jar_filename
   ```

2. Make the following changes:
   a. Set the CLASSPATH environment variable to include `Aspect-Home\aspectjrt.jar` where `Aspect-Home` is the directory where you installed AspectJ.
   b. Set the PATH environment variable to include `Aspect-Home\bin`.
   c. Modify the AspectJ compiler batch file `Aspect-Home\bin\ajc.bat` to use `-Xmx1500M` instead of `-Xmx64M`.

3. To create an aspect-enabled JAR file, run the following command on a single line:
   ```
   ajc -inpath full path to jar file
   -sourceroots full path of directory that contains the aspect source file
   -extdirs full path of directory that will contain the built JAR file
   -outjar aspect_enabled_jar_file
   ```
   **Note:** Contact IBM Support for the values that you must specify for `-inpath`, `-sourceroots`, `-extdirs`, and `-outjar`.

4. To implement an aspect-enabled JAR file:
   a. Go to the directory that contains the JAR file that you want to replace.
   b. Copy the JAR file to a backup location.
   c. Replace the specified JAR file with the aspect-enabled version.

5. Restart the server.
6. Set any necessary trace levels.
7. Reproduce the problem that you want to troubleshoot.
8. Collect the LOG files.
9. Replace the aspect-enabled JAR file with the backup copy of the original JAR file.
10. Reset the trace levels.
11. Restart the server.

### Portal version and history information

You can use the IBM WebSphere Portal version and history information tools to gather information about your portal installation. This information can be useful when you need a snapshot of your portal installation specifics, for example when you contact customer support. This information is automatically included in the automated data collection that is available when you use the IBM Support Assistant Lite for WebSphere Portal.

#### Version information

The portal version information tool is located in the following directory:

- **AIX Linux Solaris**: `wp_profile_root/PtoralServer/bin`
- **IBM i**: `wp_profile_root/PtoralServer/bin`
- **Windows**: `wp_profile_root/PtoralServer/bin`
- **z/OS**: `wp_profile_root/PtoralServer/bin`

You invoke the tool by using the following command:

- **AIX Linux Solaris**: `./WPVersionInfo.sh`
• IBM i: WPVersionInfo.sh
• Windows: WPVersionInfo.bat
• z/OS: ./WPVersionInfo.sh

You can also generate a report in html format by executing the genVersionReport tool:
• AIX Linux Solaris: ./genVersionReport.sh
• IBM i: genVersionReport.sh
• Windows: genVersionReport.bat
• z/OS: ./genVersionReport.sh

History information

The History information tool can be used to gather installation history for the WebSphere Portal product. The History information tool is located in the following directory:
• AIX Linux Solaris: wp_profile_root/PortalServer/bin
• IBM i: wp_profile_root/PortalServer/bin
• Windows: wp_profile_root/PortalServer/bin
• z/OS: wp_profile_root/PortalServer/bin

The History information tool can be invoked using the following:
• AIX Linux Solaris: ./WPHistoryInfo.sh
• IBM i: WPHistoryInfo.sh
• Windows: WPHistoryInfo.bat
• z/OS: ./WPHistoryInfo.sh

You can also generate a report in HTML format by executing the genHistoryReport tool:
• AIX Linux Solaris: ./genHistoryReport.sh
• IBM i: genHistoryReport.sh
• Windows: genHistoryReport.bat
• z/OS: ./genHistoryReport.sh

Logging and tracing

If you are experiencing a problem, you might want to enable tracing and then re-create the problem to capture more log information. You can enable logging and tracing for software that is shipped with WebSphere Portal. Enabling tracing makes log output more verbose. For example, you can enable tracing within WebSphere Application Server to obtain information about application servers and other processes.

Refer to the MustGather data collection lists used in troubleshooting various problems in WebSphere Portal and IBM Web Content Manager. Collecting MustGather data early, even before opening a PMR, helps IBM Product Support quickly determine if:
• Symptoms match known problems (rediscovery).
• There is a non-defect problem that can be identified and resolved.
• There is a defect that identifies a workaround to reduce severity.
• Locating the root cause can speed development of a code fix.

Simplify this process even more by using the IBM Support Assistant Lite for WebSphere Portal to automate the collection of the diagnostic data needed to troubleshoot most of these situations. You can use the information gathered to help solve your own problems or to report an issue to IBM Product Support.

**Links to important WebSphere Portal tracing questions**

**How do I turn on WebSphere Portal trace logging?**

See “Trace logging” on page 775 for information.

**What are the different trace settings and where are the logged?**

See “WebSphere Portal run-time logs” for information.

**How do I change the location of my logs?**

See “Changing the log file name and location” on page 776

**WebSphere Portal run-time logs**

If tracing is enabled, IBM WebSphere Portal generates a log file during run time that contains messages and trace information.

The default run-time log file is:

- **Windows:** wp_profile_root/logs/WebSphere_Portal/trace.log
- **AIX Linux Solaris:** wp_profile_root/logs/WebSphere_Portal/trace.log
- **IBM i (UserData):** wp_profile_root/logs/WebSphere_Portal/trace.log
- **z/OS:** wp_profile_root/logs/WebSphere_Portal/trace.log

**Note:** The default run-time log file is at the output location you specified as ras_trace_outputLocation in your IBM WebSphere Application Server for z/OS installation. It can either be in the job log of the servant region, in a CTRACE data set, or in a file in HFS. For details, see [Trace control settings](#).

See the topic on system event logging for details on how to configure logging and for information on the grammar of the “trace string” configuration key.

The following table describes trace loggers for particular situations and problem symptoms. Enabling the trace loggers can slow down WebSphere Portal.

**Notes:**

- The trace strings beginning with `com.ibm.wps.*` are extensions in the IBM portlet API.
- In case of problems with portal administration portlets, the error is usually not caused by the portlet code itself, but by the underlying functionality for which the portlet provides the UI. Therefore the portlet trace strings are not listed here. In case of issues with these portlets, provide the trace strings of the underlying function. If you need traces or logs for portlets, support personnel will tell you how to obtain them.

**Access Control**

**When to use**

Enable this tracer if you want permissions for resources to be explained in detail, need to verify the correctness of a permission, or need to isolate a defect in access control.
Trace String
com.ibm.wps.ac.*=all

Additional comments
The traces are easier to evaluate while WebSphere Portal usage is low.

Important: Enabling this logger creates very large log files.

Authentication
Trace String
com.ibm.wps.services.puma.*=all:
com.ibm.wps.puma.*=all:
com.ibm.wps.auth.*=all:
com.ibm.wps.sso.*=all:
com.ibm.wps.um.*=all:
com.ibm.wps.services.authentication.*=all

Command
When to use
Use to turn on all command trace loggers.

Trace String
com.ibm.wps.commands.*=all

Layout Model
When to use
Enable these messages if you want to get more information on how pages are constructed, need to verify page lists displayed on WebSphere Portal for correctness, or need to isolate an error in the WebSphere Portal aggregation component.

Trace String
com.ibm.wps.model.*=all:
com.ibm.wps.composition.*=all

Additional comments
The traces are easier to evaluate while WebSphere Portal usage is low.

Important: Enabling this logger creates very large log files.

Credential Vault
Trace String
com.ibm.wps.sso.credentialvault.*=all:
com.ibm.wps.command.credentialvault.*=all:
com.ibm.wps.portletservice.credentialvault.*=all:
com.ibm.wps.services.credentialvault.*=all:
com.ibm.portal.portlet.service.credentialvault.*=all

Database
When to use
Deals with generated SQL statements and the internal flow in the WebSphere Portal database layer.

Trace String
com.ibm.wps.datastore.*=all:
com.ibm.wps.services.datastore.*=all

Additional comments
Important: Enabling this logger will create very large log files.

**Engine**

**When to use**
Use to enable all engine trace loggers.

**Trace string**
```java
com.ibm.wps.engine.*=all
```

**General**

**Trace string**
```java
com.ibm.wps.*=all
```

**Note:** If you want to use general tracing but do not want render times to be displayed for such portlets, you must selectively disable tracing using the following trace string:
```java
com.ibm.wps.pe.PortletRenderTimeLoggingHelper=info
```

**Additional comments**
When general tracing is enabled and parallel portlet rendering is turned on, portlets that are configured to be rendered in parallel will display the render time as part of the portlet content.

**Mail Service**

**When to use**
Use to diagnose problems with the Mail Service.

**Trace string**
```java
com.ibm.wps.services.mail.*=all
```

**Mapping URLs**

**When to use**
Use to diagnose problems with the user-defined mappings of URLs

**Trace string**
```java
com.ibm.wps.mappingurl.*=all:
com.ibm.wps.command.mappingurl.*=all
```

**Personalization**

**Trace string**
```java
com.ibm.websphere.personalization.*=all:
com.ibm.dm.pzn.ui.*=all
```

**Additional comments**
When Personalization is installed outside of a WebSphere Portal server, Personalization will log using WebSphere Application Server tracing with the same trace strings.

**Portlet Container**

**Trace string**
```java
com.ibm.wps.pe.pc.*=all:
org.apache.jetspeed.portlet.Portlet=all:
javax.portlet.Portlet=all
```
Portlet Environment

Trace string
com.ibm.wps.pe.ext.*=all:
com.ibm.wps.pe.factory.*=all:
com.ibm.wps.pe.om.*=all:
com.ibm.wps.pe.util.*=all

Portlet Load Monitoring

When to use
Use to diagnose problems with Portlet Load Monitoring (PLM).

Trace string
com.ibm.wps.pe.pc.waspc.plm.*=all:
com.ibm.wps.command.plm.*=all

Deployment

Trace string
com.ibm.wps.pe.mgr.*=all:
com.ibm.wps.services.deployment.*=all:
com.ibm.wps.command.applications.*=all:
com.ibm.wps.command.portlets.*=all

Portlets

When to use
Use to diagnose problems with portlets.

Trace string
com.ibm.wps.portlets.*=all:
org.apache.jetspeed.portlet.PortletLog=all

Additional comments
Enables tracing for all portlets. Therefore, place the suspect portlet on a separate page for testing.

Scripting Interface

When to use
Use this trace string to diagnose problems with the Portal Scripting Interface, or with application interface scripting, and the execution of such scripts.

Trace string
com.ibm.wps.scripting.*=all

Additional comments
The traces are easier to evaluate while portal usage is low.

Note: Enabling this logger can create large log files fast.

Selfcare

When to use
Use to diagnose problems with user registration and profile editing.

Trace string
com.ibm.wps.services.puma.*=all:
com.ibm.wps.puma.*=all:
com.ibm.wps.um.*=all
Additional comments
Use this logger if there are errors in the sign-up, Edit My Profile, and the Manage Users and Groups portlets.

Services: EventBroker
Trace string
com.ibm.wps.services.registry.EventHandlerRegistry=all:
com.ibm.wps.services.events.**=all

Services: Finder
When to use
Use for debugging the resolution of file names.
Trace string
com.ibm.wps.services.finder.**=all

Services: Loader
When to use
Use to trace the dynamic class loading performed by this service.
Trace string
com.ibm.wps.services.ServiceManager=all

ServicesNaming
When to use
Use to debug the lookup of objects by the naming service.
Trace string
com.ibm.wps.services.naming.**=all

ServicesNavigator
When to use
Use to diagnose problems with parts of page aggregation and display.
Trace string
com.ibm.wps.services.navigator.**=all

ServicesRegistry
When to use
Use to view the policies of the internal portlet object caching and watch it reload its content.
Trace string
com.ibm.wps.services.registry.**=all

Services
When to use
Use for switching on tracing for all services.
Trace string
com.ibm.wps.services.**=all

SSO
When to use
Use to turn on all SSO tracer loggers.
Trace string
com.ibm.wps.sso.*=all

Additional comments
Use this logger if errors occur when using the Security Vault task on the Security page of the Administration pages.

**WSRP administration**

When to use
Use to diagnose problems occurring during the administration of Web Services for Remote Portlets (WSRP) with WebSphere Portal.

Trace string
com.ibm.wps.command.wsrp.*=all:
com.ibm.wps.wsrp.cmd.*=all

**WSRP Consumer**

When to use
Use to diagnose problems occurring during the use of WSRP with WebSphere Portal as a Consumer.

Trace string
com.ibm.wps.wsrp.consumer.*=all

**WSRP Producer**

When to use
Use to diagnose problems occurring during the use of WSRP with WebSphere Portal as a Producer.

Trace string
com.ibm.wps.wsrp.producer.*=all

**XML configuration interface**

When to use
Use to diagnose problems with the XML import/export of WebSphere Portal configurations.

Trace string
com.ibm.wps.command.xml.*=all

**Verbose garbage collection in Java Virtual Machine (JVM) logs**

Verbose garbage collection (verbosegc) logging is often required when tuning and debugging many issues, particularly memory problems, and has negligible impact on system performance.

The default WebSphere Portal installation enables verbose garbage collection (verbosegc) logging and configures the following generic JVM argument:

```
-Xverbosegclog:${SERVER_LOG_ROOT}/verbosegc.m%d.5/4/12M%S.%pid.txt,20,10000
```

The verbosegc log file name is `verbosegc.m%d.5/4/12M%S.%pid.txt`. It includes a date/time stamp and the process id (pid) of the WebSphere Portal instance.

The default WebSphere Portal installation redirects the verbosegc output to 20 rotating historical log files, each containing 10000 garbage collection (GC) cycles.
For more information about configuring the JVM through WebSphere Application Server, see the IBM WebSphere Application Server information centers at www.ibm.com/software/webservers/appserv/was/library.

For more information about JVM argument -Xverbosegclog, see Java Diagnostic Guides at www.ibm.com/developerworks/java/jdk/diagnosis/60.html.

**WebSphere Application Server tracing and log files**

Use WebSphere Application Server log files and tracing to troubleshoot problems with WebSphere Portal.

WebSphere Application Server has log files and a tracing function; however, whenever possible use the IBM Installation Manager installation logs to determine whether WebSphere Application Server was successfully configured for WebSphere Portal and whether WebSphere Portal was successfully started on WebSphere Application Server. See [http://www-10.lotus.com/ldd/portalwiki.nsf/dx/Installation_and_migration_logs_wp8](http://www-10.lotus.com/ldd/portalwiki.nsf/dx/Installation_and_migration_logs_wp8).

If you are on the z/OS platform, IBM WebSphere Application Server for z/OS has log files and a tracing function. Some of the WebSphere Application Server for z/OS log files are located in the directory `was_profile_root/logs`. Furthermore, if you enable WebSphere Application Server for z/OS tracing, depending on option ras_trace_outputLocation, the job logs or log streams of the application server control and servant region contain information that might be helpful in detecting and diagnosing problems. Check the information center of WebSphere Application Server for z/OS for details on how to enable tracing and logging. For information, see [Trace controls for IBM service](http://www-10.lotus.com/ldd/portalwiki.nsf/dx/Trace_controls_for_IBM_service).

**System event logging**

The system event logging facility of IBM WebSphere Portal enables the recording of information about the operation of WebSphere Portal.

Event logs provide administrators with information on important or abnormal events, especially errors that occur during the operation of the product. In addition, event logs gather debugging information that helps IBM support to resolve problems.

WebSphere Portal provides two types of logging: logging of messages and logging of debugging messages called traces.

For information about how to use log files and a list of trace logger strings refer to the topic about WebSphere Portal logs.

**Message logging**

WebSphere Portal provides the logging of messages that report errors and status information.

The following types of messages are provided:

**Informational**

A condition worth noting but does not require the user to complete an action.
Warning
An abnormal condition has been detected. The user may have to take action. However, WebSphere Portal code is able to handle the condition without failing.

Error
A serious failure in the execution of the application that requires further action.

Messages for WebSphere Portal are logged in the following files:

SystemOut.log
Contains information that is useful for monitor the health of the WebSphere Portal server and all running processes.

System.err
Contains exception stack trace information that is useful when performing problem analysis.

Locating the log files: Log files for WebSphere Portal, including SystemOut.log and System.err are located in the following directory: \wp_profile_root\logs/ WebSphere_Portal

Trace logging
WebSphere Portal provides the logging of debugging messages called traces. These traces are useful for fixing problems. However, to save system resources, they are switched off by default.

Traces can be set for different durations:

Temporary
Traces can be set for a temporary period by using the administration portlet Enable Tracing or the WebSphere Integrated Solutions Console. To set traces by using the portlet, complete the following steps:
1. Log in as the administrator.
2. Click Administration > Portal Analysis > Enable Tracing. The Enable Tracing portlet appears.
3. Type the required trace string into the field Append these trace settings. For example, this can be com.ibm.wps.command.credentialvault.*=finest
4. Click the Add icon. Enable Tracing updates the field Current trace settings:

Note: Restarting WebSphere Portal will remove traces that were set by using the Enable Tracing Administration portlet.
To disable tracing, do either of the following:
• Select the current trace settings under Current trace settings: and click the Remove icon. By the example, the current setting can be com.ibm.wps.command.credentialvault.*=finest.
• Type the trace string *=info into the field Append these trace settings: and click the Add icon. This trace string overwrites all settings listed under Current trace settings: and resets it to the default.

Extended
To enable trace settings for a longer period of time, that is, for more than one session, switch them on in the WebSphere Application Server configuration. Proceed by the following steps:
1. Access the WebSphere Integrated Solutions Console using this URL:
   http://hostname:port_number/ibm/console
2. Go to Servers > Server Types > WebSphere application servers.
3. Select the application server.
4. Click Troubleshooting > Change Log Detail Levels.
5. Specify the required trace settings. For example, this can be
   com.ibm.wps.command.credentialvault.*=finest
6. Save your updates.
7. Restart the WebSphere_Portal server.
8. To disable tracing, specify tracestring: *=info and restart the
   WebSphere_Portal server.

Changing the log file name and location

You can change the locations of the log files by configuring them in the WebSphere
Integrated Solutions Console. Go to Troubleshooting > Logs and Trace >
server_name and select the logger type that you want to change. In the
configuration dialog change the path for the log file as required.

Changing the language used in the log file

By default, information in the log file is written in the language that was used for
the WebSphere Portal installation. However, because WebSphere Portal supports a
number of languages, you can choose to have the log file information written in a
language other than that used during installation.

To change the language used for the log file, edit the file log.properties. This file
is located in the following WebSphere Portal directory:

- AIX Linux Solaris: wp_profile_root/PortalServer/config/config
- z/OS: wp_profile_root/PortalServer/config/config
- IBM i: wp_profile_root/PortalServer/config/config
- Windows: wp_profile_root/PortalServer\config\config

Add the following line:

locale=xx

where xx is the two-letter abbreviation for the locale. For a list of the locale
abbreviations used with WebSphere Portal, refer to Directories for languages. For
example, to have log information generated in English, you would add the
following line:

locale=en

Reference: Log file format

If the logs are written to the log file of WebSphere Portal and not redirected to the
logging facility WebSphere Application Server, the log file consists of a sequence
log records that are separated by blank lines.

The log records have the following format:

- timestamp classification classname method threadID
- messagecode: logmessage

Where:
• timestamp is the time (to the millisecond) when the log record was created.

• classification is one of the following letters:
  – E for error messages
  – W for warning messages
  – I for informational messages
  – l for traces (low details)
  – m for traces (medium details)
  – h for traces (high details)

• classname is the Java class containing the code that triggered the log event.

• method is the name of the Java method containing the code that triggered the log event.

• messagecode is a unique identifier for this message, to uniquely identify the specific message and refer to it when consulting documentation or support. The message code is only available for error, warning, or informational messages, and not for traces. It consists of:
  – a four-character identifier for the component that defines the message.
  – a four-digit number identifying the message in the component.
  – a one-letter classification code, which can be E, W, or I.

• logmessage is the actual log message describing the logged event. Error, warning, and informational messages are translated into the system locale. Trace messages are not translated.

• threadID is the identification of the thread that triggered the log event.

Note:
1. Traces are written only if the specific tracing facility is enabled; all other messages are written unconditionally.
2. The system locale is part of the general internationalization features of WebSphere Portal and can be configured via LocalizerService. For more information refer to the topics about Setting service configuration properties and about the Portal configuration services.

The following is an example of a log record:
2011.05.16 13:36:14.449 W com.ibm.wps.services.datastore.DataStoreServiceImpl init 0000003a DSTO0063W: The transaction isolation level is not set to READ_COMMITTED.

The current value is TRANSACTION_REPEATABLE_READ.

Web Content Manager tracing files
Enable the use of WebSphere Application Server trace facilities to create trace information for Web Content Manager. This tracing can be enabled either permanently or for just the current WebSphere Portal session.

IBM Web Content Manager uses the IBM WebSphere Application Server trace facilities to create trace information. If you need detailed trace output of Web Content Manager to debug a problem, follow these steps:

Permanently enable tracing
1. Start WebSphere Application Server.
2. Open the WebSphere Integrated Solutions Console.
3. Go to section **Troubleshooting->Logs and Traces->WebSphere_Portal->Diagnostic Trace**.
4. Make sure that the check box **Enable Trace** is selected.
5. Enter any of the following in the **TraceSpecification** field:
   - com.ibm.workplace.wcm.*
   - com.aptrix.*
   - com.presence.*
   For example, to trace all events, enter the following:
   
   ```
   com.ibm.workplace.wcm.*=all:com.aptrix.*=all:com.presence.*=all
   ```
6. Save the changes.
7. Restart **IBM WebSphere Portal**.

**Enable tracing just for the current WebSphere Portal session**

1. Go to **Administration > WebSphere Portal > Portal Analysis > Enable Tracing**
2. Enter any of the following values in the **Append these trace settings** field:
   - com.ibm.workplace.wcm.*
   - com.aptrix.*
   - com.presence.*
   For example, to trace all events, enter the following:
   
   ```
   com.ibm.workplace.wcm.*=all:com.aptrix.*=all:com.presence.*=all
   ```

   Here is a list of advanced trace settings:

   **com.ibm.workplace.wcm.services.content.***
   This enables low level tracing for every item.

   **com.ibm.workplace.wcm.domain.transformers.control.controltype**
   You can enable tracing for any of the following control types:
   - HistoryControlTransformer
   - IdentityControlTransformer
   - ProfileControlTransformer
   - SecurityControlTransformer
   - WorkflowControlTransformer

   **com.ibm.workplace.wcm.domain.transformers.controllable.controllatabletype**
   This enables the tracing for all control types for all items.

   **com.ibm.workplace.wcm.domain.transformers.controllable.controllabletype**
   You can enable tracing for any of the following controllable types:
   - AbstractControllableTransformer
   - AlternateDesignCmpntTransformer
   - AlternateLinkCmpntTransformer
   - ArrayCmpntTransformer
   - AttributeReferenceCmpntTransformer
   - BasePathCmpntTransformer
   - BaseReferenceCmpntTransformer
   - CategoryTransformer
   - CmpntReferenceTransformer
   - CmpntTransformer
   - ConfigParamCmpntTransformer
- ContentLinkTransformer
- ContentSpotCmpntTransformer
- ContentTransformer
- ContextPathCmpntTransformer
- ControllableNodeValueTransformer
- ControllableTransformer
- DateCmpntTransformer
- EmailActionTransformer
- ExpireActionTransformer
- ExternalLinkTransformer
- FEDCmpntReferenceTransformer
- FEDCmpntTransformer
- FileResourceCmpntTransformer
- HistoryCmpntTransformer
- HTMLCmpntTransformer
- IDCmpntTransformer
- ImageResourceCmpntTransformer
- IndentCmpntTransformer
- IndexCmpntTransformer
- InlineEditCmpntTransformer
- InlineEditReferenceCmpntTransformer
- JSPCmpntTransformer
- LinkCmpntTransformer
- MenuCmpntTransformer
- NavigatorCmpntTransformer
- NoPrefixBasePathCmpntTransformer
- NoPrefixServletPathCmpntTransformer
- NumericCmpntTransformer
- ObjectSummaryTransformer
- OptionSelectionCmpntTransformer
- PageInfoCmpntTransformer
- PagingCmpntTransformer
- PDMCmpntReferenceTransformer
- PDMCmpntTransformer
- PlaceholderCmpntTransformer
- PlutoSubscriberTransformer
- PlutoSyndicatorTransformer
- PrefixPathCmpntTransformer
- ProfileCmpntTransformer
- PublishActionTransformer
- ResourceCmpntTransformer
- ScheduledMoveActionTransformer
- SearchCmpntTransformer
- SecurityCmpntTransformer
- ServletPathCmpntTransformer
This enables the tracing for all controllable types for all items.

The resulting traces of Virtual Member Manager are written to:

- **Windows:** wp_profile_root/logs/WebSphere_Portal/trace.log
- **AIX Linux Solaris:** wp_profile_root/logs/WebSphere_Portal/trace.log
- **IBM i:** wp_profile_root/logs/WebSphere_Portal/trace.log
- **z/OS:** The resulting traces of Virtual Member Manager are written to the output location you specified as ras_trace_outputLocation in the WebSphere Integrated Solutions Console. Check the information center of WebSphere Application Server for z/OS for details on what can be specified. For more information, refer to Setting trace controls.

**Contact support**

For contact information, refer to the IBM Software Support site at http://www.ibm.com/support/entry/portal/Overview/Software/WebSphere/WebSphere_Portal.
Chapter 11. Web Content Manager terminology

These common terms are used when describing IBM Web Content Manager.

Web content portlets

The following portlets are used in a Web Content Manager system.

Table 196. Portlet types

<table>
<thead>
<tr>
<th>Portlet type</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Web content authoring portlet</td>
<td>The authoring portlet is used to create and manage web content items.</td>
</tr>
<tr>
<td>Web content viewer portlets</td>
<td>Web content viewer portlets are used to deliver web content within Portal pages.</td>
</tr>
<tr>
<td>Administration portlet</td>
<td>The administration portlet is used to create and manage web content libraries, define and manage syndication relationships and define and manage Web Content Integrator feeds.</td>
</tr>
</tbody>
</table>

Web content libraries

A web content library is used to store a set of web content items. You assign library-level access controls to determine the default level of access to the items in the library and define the default access to the authoring portlet of different users and groups.

Elements

Elements either store web content or generate web content. Elements do not exist as free standing items. You store elements in "container" web content items. For example, menu and navigator elements are used to generate links between web pages. HTML and rich text elements are used to store HTML. Some element types can be stored in site areas, content items and components. Other can only be stored as components.

Item types

Items can be considered as "files" or "documents" and are used to store web content, metadata and access control information.

Table 197. Item types

<table>
<thead>
<tr>
<th>Item type</th>
<th>Examples</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Container items</td>
<td>Site areas, content items and components</td>
<td>Site areas and content items represent different sections of a site framework. You can store more than one element in site areas and content items. Components store a single element-type.</td>
</tr>
<tr>
<td>Folders</td>
<td></td>
<td>Folders are used to group sets of item-types within the different item-type views in the authoring portlet.</td>
</tr>
</tbody>
</table>
Table 197. Item types (continued)

<table>
<thead>
<tr>
<th>Item type</th>
<th>Examples</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Template items</td>
<td>Authoring templates and presentation templates</td>
<td>When creating a new content item or site area, an authoring template must first be selected. An authoring template defines: • which data entry fields are visible on an item form. • the default values for each setting and field on an item form. A presentation template defines: • the layout of elements and components that are displayed on a web page. • the default properties of a web page. When a content item or site area is rendered, the presentation template used to display the item is determined by the current template map. A template map is defined in a site area and consists of a pairing of an authoring template with a presentation template.</td>
</tr>
<tr>
<td>Profile items</td>
<td>Taxonomies and categories</td>
<td>You use categories to profile certain item-types, such as content-items. A category refers to the subject matter of your content. You group categories within taxonomies.</td>
</tr>
<tr>
<td>Change management items</td>
<td>Projects, workflows, workflow stages and workflow actions</td>
<td>You use workflows to control the access to, verification and eventual approval of Web Content Manager items. When creating a workflow, you select a set of workflow stages. Workflow actions are executed upon entering or exiting a workflow stage. Projects are used to manage changes to a set of items. Not until all items in a project have been approved and made ready to publish will all the items in a project be published to the live site.</td>
</tr>
</tbody>
</table>

Site framework

A site framework is a similar concept to the "site map" of a traditional website. Whereas a site map is based on a directory structure or the links between pages in a website, a site framework consists of a set of Web Content Manager items. Each site framework consists of a single site area under which a set of other site areas and content items are grouped.
Chapter 12. Reference

View information that will help you use the information center including directory conventions, terms of use, trademarks, a glossary, and more.

Conventions

Understand the conventions used in this documentation to use it more effectively.

The following section provides conventions that can help you interpret the information that is provided in this documentation:

- File names, directories, and commands appear in Courier font. For example:
  - File name: xmlaccess.bat or xmlaccess.sh
  - Directory: /opt/WebSphere/PortalServer
  - Command: startServer WebSphere_Portal

- Variables are either italicized, enclosed in brackets, or both. For example:
  http://hostname.example.com:10039/wps/portal, where hostname.example.com is the fully qualified host name of the machine where Portal is running and 10039 is the default transport port that is created by WebSphere Application Server; the port number may be different for your environment.

- Variables are used to indicate root installation directories. For more information, see "Directory structure" on page 784.

- Directories are shown with forward slashes (/), unless operating-system specific information is provided. On Windows systems, you should use backward slashes (\) when typing at a command line, unless otherwise noted.

- Operating system-specific information is provided, for example:
  - AIX Linux Solaris: ./ConfigEngine.sh task_name
  - IBM i: ConfigEngine.sh task_name
  - Windows: ConfigEngine.bat task_name
  - z/OS: ./ConfigEngine.sh task_name

- When reading z/OS specific documentation, you may encounter the terms UNIX or Linux. These terms imply z/OS UNIX System Services or z/OS USS when abbreviated. Outside the z/OS specific documentation, and unless otherwise indicated, the UNIX term implies *nix based operating systems such as AIX, Linux and Solaris.

- Tips on how to use specific topics are marked with a green check. For example:
  - How to use this planning section
• Links to reference information and external links are marked with one of these icons:  or 
• Most topics include a Related information section that links to other relevant topics. See the last section for an example.

Directory structure

This topic shows the naming conventions used to denote the location of files on the servers and the types of resources you can find in those directories.

- “WebSphere Portal root directory (PortalServer_root)”
- “WebSphere Portal profile directory (wp_profile_root)” on page 785
- “WebSphere Portal Configuration Engine root directory (ConfigEngine_root)” on page 785
- “WebSphere Portal Configuration Engine profile directory” on page 785
- “WebSphere Application Server directory structure (AppServer_root)” on page 787
- “Lotus Domino directory structure (domino_server_root and domino_data_root)” on page 787
- “Directories for languages” on page 788

WebSphere Portal root directory **PortalServer_root**

Throughout this documentation, the installation location for the portal server component of IBM WebSphere Portal is noted as **PortalServer_root**.

For the IBM i platform, an additional variable is used to indicate the user data directory. The user data directory is noted as **PortalServer_root_user**.

The following information shows the default location if it is not otherwise specified during installation:

**AIX** /usr/IBM/WebSphere/PortalServer

**Linux** /opt/IBM/WebSphere/PortalServer

**Solaris** /opt/IBM/WebSphere/PortalServer

**Windows**

C:\Program Files\IBM\WebSphere\PortalServer

**IBM i**

- portal_server_root (ProdData)
  - /QIBM/ProdData/WebSphere/PortalServer/V8/product offering
  Where **product offering** is Server or Express
- PortalServer_root_user (UserData)
  - WebSphere Application Server Version 8.0 for Network Deployment:
  - /QIBM/UserData/WebSphere/AppServer/V8/ND/profiles/wp_profile/PortalServer

**z/OS** /usr/lpp/zPortalServer/V8ROM0 (install root)

/WEBSphere/V8ROM0/AppServer/PortalServer (config root)
WebSphere Portal profile directory (wp_profile_root)

Throughout this documentation, the profile location is noted as \textit{wp_profile_root}. The following information shows the default profile location if another location it is not specified during installation:

- **AIX**  /usr/IBM/WebSphere/wp_profile
- **Linux**  /opt/IBM/WebSphere/wp_profile
- **Solaris**  /opt/IBM/WebSphere/wp_profile
- **Windows**  C:\Program Files\IBM\WebSphere\wp_profile
- **IBM i**  /QIBM/UserData/WebSphere/AppServer/V8/ND/profiles/wp_profile
- **z/OS**  /WebSphere/V8R0/AppServer/profiles/default

**WebSphere Portal Configuration Engine root directory (ConfigEngine_root)**

Throughout this documentation, the installation location for the Configuration Engine component is noted as \textit{ConfigEngine_root}.

- **AIX**  /usr/IBM/WebSphere/ConfigEngine
- **Linux**  /opt/IBM/WebSphere/ConfigEngine
- **Solaris**  /opt/IBM/WebSphere/ConfigEngine
- **Windows**  C:\Program Files\IBM\WebSphere\ConfigEngine
- **IBM i**  /QIBM/ProdData/WebSphere/PortalServer/V8/ConfigEngine
- **z/OS**  /usr/lpp/zWebSphere/V8R0/ConfigEngine

**WebSphere Portal Configuration Engine profile directory**

The Configuration Engine profile directory is the location of the \textit{ConfigEngine} task.

- **AIX**  /usr/IBM/WebSphere/wp_profile/ConfigEngine
- **Linux**  /opt/IBM/WebSphere/wp_profile/ConfigEngine
- **Solaris**  /opt/IBM/WebSphere/wp_profile/ConfigEngine
- **Windows**  C:\Program Files\IBM\WebSphere\wp_profile\ConfigEngine
- **IBM i**  
  - WebSphere Application Server Version 8.0 for Network Deployment:
    - /QIBM/UserData/WebSphere/AppServer/V8/ND/profiles/wp_profile/ConfigEngine
wp_profile is the default profile name but is used here as an example since there may be multiple profiles with self described or incremental names (for example, wp_profile1, wp_profile2, and so on).

z/OS /WebSphere/V8R0/AppServer/profiles/default/ConfigEngine

**WebSphere Portal directory structure after installation**

WebSphere Portal has the following directory structure after installation:

**Note:** On the Linux and IBM i operating systems, all directories are r/o. On z/OS operating systems, all subdirectories are r/w in the ConfigEngine_root directory. However, the content might be r/w (if files were copied) or r/o (if files are symlinked).

```
<table>
<thead>
<tr>
<th>PortalServer_root</th>
<th>Root directory for WebSphere Portal</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ap</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

786 Web Content Manager Version 8.0
**WebSphere Application Server directory structure**

WebSphere Application Server directory structure (AppServer_root)

Throughout this documentation, the install location for WebSphere Application Server is noted as `AppServer_root`.

Throughout this documentation, the install location for WebSphere Application Server is noted as `AppServer_root`.

The following table shows the WebSphere Application Server installation directory:

<table>
<thead>
<tr>
<th>Platform</th>
<th>Directory Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>z/OS</td>
<td><code>/usr/lpp/zWebSphere/V8R0</code> (install root)</td>
</tr>
<tr>
<td></td>
<td><code>/WebSphere/V8R0/AppServer</code> (config root)</td>
</tr>
<tr>
<td>IBM i</td>
<td>The install location for WebSphere Application Server is noted as <code>app_server_root</code> and refers to the UserData path, unless otherwise specified in the topic where you see it. The <code>profile_root</code> variable refers to the name given to the WebSphere Application Server profile in use.</td>
</tr>
</tbody>
</table>

**WebSphere Application Server profile directory**

Throughout this documentation, the location for the WebSphere Application Server profiles is noted as `was_profile_root`.

**Lotus Domino® directory structure** (domino_server_root and domino_data_root)

Lotus Domino® directory structure (domino_server_root and domino_data_root)

Throughout this documentation, the install location for the IBM Lotus Domino server software is noted as `domino_server_root`.

For IBM i, the install location for the Lotus Domino server software is noted as `domino_server_root` and refers to the ProdData path, unless otherwise specified in the topic where you see it.

Throughout this documentation, the install location for the Lotus Domino server software is noted as `domino_server_root` which is the install root.

The following information shows the default location if it is not otherwise specified during installation:

- **AIX** /opt/IBM/Lotus/
- **Linux** /opt/IBM/Lotus/
- **Solaris** /opt/IBM/Lotus/
- **Windows** C:\Lotus\Domino
- **IBM i**
  - `domino_server_root` (ProdData)
The Lotus Domino server data directory is noted as `domino_data_root`.

For IBM i, the Lotus Domino server data directory is noted as `domino_data_root` and refers to the UserData path, unless otherwise specified in the topic where you see it.

The Lotus Domino server data directory is noted as `domino_data_root`, which is the config root.

The following information shows the default location of the Domino data directory if it is not otherwise specified during installation:

- **AIX** /opt/IBM/Lotus/Domino/data
- **Linux** /opt/IBM/Lotus/Domino/data
- **Solaris** /opt/IBM/Lotus/Domino/data
- **Windows** C:\Lotus\Domino\Data
- **IBM i**
  - `domino_data_root` (UserData)
    - WebSphere Application Server for Base and Express:
      - /QIBM/USERDATA/LOTUS/NOTES
    - WebSphere Application Server for Network Deployment:
      - /QIBM/USERDATA/LOTUS/NOTES
- **z/OS** /usr/lpp/lotus/data (config root)

### Directories for languages

The following shows the languages supported by WebSphere Portal and the directories used for storing locale-specific resources. These directories are used in portlet Web application directories and in the WebSphere Portal enterprise application (themes, skins, and other Web application resources).

*Table 198. Languages supported by WebSphere Portal*

<table>
<thead>
<tr>
<th>Language (locale)</th>
<th>Directory</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arabic</td>
<td>/ar</td>
</tr>
<tr>
<td>Danish</td>
<td>/da</td>
</tr>
<tr>
<td>English</td>
<td>/en</td>
</tr>
<tr>
<td>French</td>
<td>/fr</td>
</tr>
<tr>
<td>Italian</td>
<td>/it</td>
</tr>
<tr>
<td>Kazakh</td>
<td>/kk</td>
</tr>
<tr>
<td>Korean</td>
<td>/ko</td>
</tr>
<tr>
<td>Polish</td>
<td>/pl</td>
</tr>
<tr>
<td>Romanian</td>
<td>/ro</td>
</tr>
<tr>
<td>Slovenian</td>
<td>/sl</td>
</tr>
</tbody>
</table>
Table 198. Languages supported by WebSphere Portal (continued)

<table>
<thead>
<tr>
<th>Language (locale)</th>
<th>Directory</th>
</tr>
</thead>
<tbody>
<tr>
<td>Turkish</td>
<td>/tr</td>
</tr>
<tr>
<td>Traditional Chinese</td>
<td>/zh_TW</td>
</tr>
<tr>
<td>Catalan</td>
<td>/ca</td>
</tr>
<tr>
<td>German</td>
<td>/de</td>
</tr>
<tr>
<td>Spanish</td>
<td>/es</td>
</tr>
<tr>
<td>Croatian</td>
<td>/hr</td>
</tr>
<tr>
<td>Hebrew</td>
<td>/iw</td>
</tr>
<tr>
<td>Dutch</td>
<td>/nl</td>
</tr>
<tr>
<td>Portuguese</td>
<td>/pt</td>
</tr>
<tr>
<td>Russian</td>
<td>/ru</td>
</tr>
<tr>
<td>Swedish</td>
<td>/sv</td>
</tr>
<tr>
<td>Ukrainian</td>
<td>/uk</td>
</tr>
<tr>
<td>Czech</td>
<td>/cs</td>
</tr>
<tr>
<td>Greek</td>
<td>/el</td>
</tr>
<tr>
<td>Finnish</td>
<td>/fi</td>
</tr>
<tr>
<td>Hungarian</td>
<td>/hu</td>
</tr>
<tr>
<td>Japanese</td>
<td>/ja</td>
</tr>
<tr>
<td>Norwegian</td>
<td>/no</td>
</tr>
<tr>
<td>Brazilian Portuguese</td>
<td>/pt_BR</td>
</tr>
<tr>
<td>Slovak</td>
<td>/sk</td>
</tr>
<tr>
<td>Thai</td>
<td>/th</td>
</tr>
<tr>
<td>Simplified Chinese</td>
<td>/zh</td>
</tr>
</tbody>
</table>

Terms of use

Understand the terms and conditions before you use this publication.

Permissions for the use of publications is granted subject to the following terms and conditions.

**Personal Use:** You may reproduce these publications for your personal, non-commercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative work of these publications, or any portion thereof, without the express consent of IBM.

**Commercial Use:** You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of IBM.

Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any information, data, software or other intellectual property contained therein.
IBM reserves the right to withdraw the permissions granted herein whenever, in its
discretion, the use of the publications is detrimental to its interest or, as
determined by IBM, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full
compliance with all applicable laws and regulations, including all United States
export laws and regulations.

IBM MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE
PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT
WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING
BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY AND
FITNESS FOR A PARTICULAR PURPOSE.

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in
other countries. Consult your local IBM representative for information on the
products and services currently available in your area. Any reference to an IBM
product, program, or service is not intended to state or imply that only that IBM
product, program, or service may be used. Any functionally equivalent product,
program, or service that does not infringe any IBM intellectual property right may
be used instead. However, it is the user’s responsibility to evaluate and verify the
operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter
described in this document. The furnishing of this document does not grant you
any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM
Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
1623-14, Shimotsuruma, Yamato-shi
Kanagawa 242-8502 Japan

The following paragraph does not apply to the United Kingdom or any other
country where such provisions are inconsistent with local law: INTERNATIONAL
BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS"
WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED,
INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF
NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR
PURPOSE. Some states do not allow disclaimer of express or implied warranties in
certain transactions, therefore, this statement may not apply to you.
This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

Intellectual Property & Licensing
4205 S Miami Blvd.,
Durham NC 27703
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM’s future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.
If you are viewing this information softcopy, the photographs and color illustrations may not appear.

**Trademarks**

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at www.ibm.com/legal/copytrade.shtml.

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

---

**Glossary**

This glossary includes terms and definitions for IBM WebSphere Portal.

The following cross-references are used in this glossary:

1. **See** refers the reader from a term to a preferred synonym, or from an acronym or abbreviation to the defined full form.
2. **See also** refers the reader to a related or contrasting term.

To view glossaries for other IBM products, go to [www.ibm.com/software/globalization/terminology](http://www.ibm.com/software/globalization/terminology)
access control. In computer security, the process of ensuring that users can access only those resources of a computer system for which they are authorized.

aggregation. The structured collection of data objects for subsequent presentation within a portal.

Ajax. A design approach and a set of techniques for delivering rich Internet applications (RIAs) using open web formats, for example, HTML, CSS and JavaScript; and rendering using a browser engine.

Ajax portlet. A normal server side portlet that uses lots of JavaScript and Ajax technologies and less Java and Java server pages.

anonymous user. A user who does not use a valid user ID and password to log into a site.

applet. A program that performs a specific task and is usually portable between operating systems. Often written in Java, applets can be downloaded from the Internet and run in a web browser.

application server. A server program in a distributed network that provides the execution environment for an application program.

Asynchronous JavaScript and XML. See Ajax.

authenticated user. A portal user who has logged in to the portal with a valid account (user ID and password). Authenticated users have access to all public places.

authentication. A security service that provides proof that a user of a computer system is genuinely who that person claims to be. Common mechanisms for implementing this service are passwords and digital signatures. Authentication is distinct from authorization; authentication is not concerned with granting or denying access to system resources.

authorization. The process of granting a user, system, or process either complete or restricted access to an object, resource, or function.

B

B2B. See business-to-business

B2C. See business-to-consumer

B2E. See business-to-employee

bind. To establish a connection between software components on a network using an agreed-to protocol. In web services, the bind operation occurs when the service requestor invokes or initiates an interaction with the service at run time using the binding details in the service description to locate, contact, and invoke the service.

bookmark. A customizable, graphical link to databases, views, documents, web pages, and newsgroups.

business-to-business (B2B). Refers to Internet applications that exchange information or run transactions between businesses.

business-to-consumer (B2C). Refers to the subset of Internet applications that exchange information or run transactions between businesses and consumers.

CA. See certificate authority

card. WML document that provides user-interface and navigational settings to display content on mobile devices.

cascading style sheet (CSS). A file that defines a hierarchical set of style rules for controlling the rendering of HTML or XML files in browsers, viewers, or in print.

certificate authority (CA). A trusted third-party organization or company that issues the digital certificates. The certificate authority typically verifies the identity of the individuals who are granted the unique certificate.

client side aggregation (CSA). Aggregation based on JavaScript and XSLT transformations that are executed on the client.

cloud application. An application that is extended to be accessible through the Internet. Cloud applications use large data centers and powerful servers that host web applications and web services.

cloud computing. A computing platform where users can have access to applications or compute resources, as services, from anywhere through their connected devices. A simplified user interface and application programming interface (API) makes the infrastructure supporting such services transparent to users.

collaboration. The ability to connect customers, employees, or business partners to the people and processes in a business or organization, in order to facilitate improved decision-making. Collaboration involves two or more individuals with complementary skills interacting together to resolve a business problem.

collaborative components. UI-neutral API methods and tag libraries that allow developers to add IBM Lotus collaborative functionality to their portlets.

collaborative filtering. Personalization technology that calculates the similarity between users based on the behaviors of a number of other people and uses that information to make recommendations for the current user.

collaborative portal. A highly personalized desktop-to-web tool designed for specific audiences and communities of users that organizes information, applications, and services for effective community building at the corporate level and for personal use by individuals.

congrete portlet. A logical representation of a portlet object distinguished by a unique configuration parameter (PortletSettings).

confirmable methods. Interface methods that exist on each modifiable interface of a portal resource that allow users to determine whether a modification can be performed or not.

connector. A servlet that provides a portlet access to external sources of content, for example, a news feed from a web site of a local television station.

content management. Software designed to help businesses manage and distribute content from diverse sources.

content partner. See IBM content partner

currently provider. A source for content that can be incorporated into a portal page as a portlet.

controller. A modifiable instance of a portal model which allows to modify the topology of the model, create and delete resources, and create modifiable instances of existing resources.

cooperative portlets. Two or more portlets on the same web page that interact by sharing information.

creation context. A context that defines immutable properties of a resource that you can create.
CSS. See **cascading style sheet**

**D**

**DB2.** A family of IBM licensed programs for relational database management.

**deck.** An XML document that contains a collection of WML cards.

**default portal page.** The page that displays to a user at initial portal deployment and before the user completes enrollment. Sometimes used as a synonym for home page.

**default public place.** A place whose membership automatically includes all portal users and which appears in the Places selector for every user. A user is always a member of this place.

**derived page.** One or more child pages that have a shared layout that is inherited from the properties of the parent page.

**differential page rendering (DPR).** Renders only those parts of a portal page that were affected by the a user interaction.

**document type definition (DTD).** The rules that specify the structure for a particular class of SGML or XML documents. The DTD defines the structure with elements, attributes, and notations, and it establishes constraints for how each element, attribute, and notation can be used within the particular class of documents.

**DTD.** See **document type definition**

**dynamic layout.** Standard portal layout that consists of rows or columns and is persisted in the database.

**E**

**ECM.** See **Enterprise Content Management**

**embedded static page.** A static page that is rendered in the content area of the portal.

**enrollment.** The process of entering and saving user or user group information in a portal.

**Enterprise Content Management (ECM).** Software and tools designed to enable companies to manage content and documents, optimize business processes, and enable compliance with an integrated infrastructure.

**Enterprise Information Portal.** Software developed by IBM that provides tools for advanced searching, and content customization and summarization.

**Extensible Markup Language (XML).** A standard metalanguage for defining markup languages that is based on Standard Generalized Markup Language (SGML).

**Extensible Stylesheet Language (XSL).** A language for specifying style sheets for XML documents. Extensible Stylesheet Language Transformation (XSLT) is used with XSL to describe how an XML document is transformed into another document.

**F**

**federated search.** A search capability that enables searches across multiple search services and returns a consolidated list of search results.

**G**

**group.**
1. A collection of users who can share access authorities for protected resources.
2. In places, two or more people who are grouped for membership in a place.
**governance.** The decision making processes in the administration of an organization. The rights and responsibilities of these processes are typically shared among the organization's participants, especially the management and stakeholders.

**governance life cycle.** A life cycle that represents the states and transitions that can exist in SOA deployment.

**governance processes.** A process that ensures that compliance and operational polices are enforced, and that change occurs in a controlled fashion and with appropriate authority as envisioned by the business design.

**H**

**helper file.** A properties file that WebSphere Portal provides to ensure that users specify the correct information that is needed to complete different types of configuration tasks such as configuring an LDAP user registry or a database user registry.

**home page.** The top-level web page of a portal.

**HTML.** See [Hypertext Markup Language](#)

**HTTP.** See [Hypertext Transfer Protocol](#)

**HTTP over SSL (HTTPS).** A web protocol for secure transactions that encrypts and decrypts user page requests and pages returned by the web server.

**HTTPS.** See [HTTP over SSL](#)

**Hypertext Markup Language (HTML).** A markup language that conforms to the Standard Generalized Markup Language (SGML) standard and was designed primarily to support the online display of textual and graphical information, including hypertext links.

**Hypertext Transfer Protocol (HTTP).** An Internet protocol that is used to transfer and display hypertext and XML documents on the web.

**I**

**i-mode.** An Internet service for wireless devices.

**IBM content partner (content partner).** IBM partner that provides syndicated content for portals.

**integrity.** In computer security, assurance that the information that arrives at a destination is the same as the information that was sent.

**iwidget.** An open-source specification that allows for seamless interoperability across various platforms and products.

**J**

**JavaScript.** A web scripting language that is used in both browsers and web servers. (Sun)

**JavaScript Object Notation.** A lightweight data-interchange format that is based on the object-literal notation of JavaScript. JSON is programming-language neutral but uses conventions from languages that include C, C++, C#, Java, JavaScript, Perl, Python.

**Jetspeed.** The open-source portal that is part of the Jakarta project by Apache.
**L**

**label.** A node in a portal that cannot contain any content, but can contain other nodes. Labels are used primarily to group nodes in the navigation tree.

**lazy application.** An application whose initialization is deferred until first use.

**LDAP.** See [Lightweight Directory Access Protocol](#).

**LDAP directory.** A type of repository that stores information on people, organizations, and other resources and that is accessed using the LDAP protocol. The entries in the repository are organized into a hierarchical structure, and in some cases the hierarchical structure reflects the structure or geography of an organization.

**light mode.** An operation method that enhances portal performance by improving start-up time and reducing memory consumption in production environments.

**Lightweight Directory Access Protocol (LDAP).** An open protocol that uses TCP/IP to provide access to directories that support an X.500 model and that does not incur the resource requirements of the more complex X.500 Directory Access Protocol (DAP). For example, LDAP can be used to locate people, organizations, and other resources in an Internet or intranet directory.

**load balancing.** The monitoring of application servers and management of the workload on servers. If one server exceeds its workload, requests are forwarded to another server with more capacity.

---

**M**

**mandatory place.** A shared place, either a public place or a restricted place, in which all portal users must be members. Only portal administrators can designate a shared place to be a mandatory place. Because membership is automatic and required, portal users cannot join or leave mandatory places.

**membership.** The state of being a portal user and a place member. Membership in the portal is controlled by the administrator during the installation and set up of portal servers. Membership in places is controlled by a place manager, who determines the level of access for each place member: participant, place designer, or place manager.

**mashup.** A graphical interface that features two or more reusable web applications (widgets) presenting seemingly disparate data in an understandable combination for a specific purpose.

**meta search.** A search across one or more search engines. A meta search engine provides a meaningful subset of search functionality through an abstraction layer that is generic enough to support a wide variety of search services.

**messaging.** A method for communication between programs. Messaging can be synchronous or independent of time.

**middleware.** Software that acts as an intermediate layer between applications or between client and server. It is used most often to support complex, distributed applications in heterogeneous environments.

**model view controller (MVC).** A software architecture that separates the components of the application: the model represents the business logic or data; the view represents the user interface; and the controller manages user input or, in some cases, the application flow.

**modifiable.** An interface for modifying portal resources that exist in the read only model.

**MVC.** See [model view controller](#).
News Industry Text Format (NITF). An XML-based format that defines the structure and content of news articles.


NewsML. See News Markup Language
NITF. See News Industry Text Format

node. A logical group of managed servers.

OCS channel. See open content syndication channel

open content syndication channel (OCS channel). An XML-based format for syndicated content.

page. A node in a portal that can contain content in addition to labels and other pages. Pages can contain child nodes, column containers, row containers, and portlets.

participant. A member of a portal place who can visit and use the place. By default, all portal users are participants in public places.

people awareness. The collaboration feature that provides access to people from various contexts. People awareness lets you see references to people and contact people by name through the Sametime® online status indicator. Throughout the portal, wherever you see the name of a person, you can view the person's online status, send e-mail, initiate a chat, or share an application via an electronic meeting.

people finder. A portlet that enables users to find, view information about, and contact individuals in their organization. Administrators can configure people finder to display information details such as e-mail address, job title, and location.

person. An individual authenticated by the portal and having a person record in one or more corporate directories. Persons can be members of places, public groups within the organization's corporate directory, or personal groups that a user defines.

person card. An interface that displays information about a registered user such as phone number and online status (if Sametime is enabled), and additional details typically found on a business card. Available actions let you view the person's complete profile and, depending on how the portal is configured, send e-mail, chat, and link to Lotus Connections features such as Communities, Activities, and Blogs.

person link. A reference to a person's name or a group name that appears with the Sametime online status indicator. The reference lets you view the person's online status, send an e-mail, start a chat, or share an application using an electronic meeting, among other actions shown on the person link menu.

personal group. In Sametime Connect, a group of people designated by the user as a group. A user can choose individuals from the public Directory (public group) and create personal groups, which are then stored locally. Users can add and remove people from a personal group, whereas the membership of the public group is defined by the owner of the public Directory.

personalization. The process of enabling information to be targeted to specific users based on business rules and user profile information.

pervasive computing. The use of a computing infrastructure that supports information appliances from which users can access a broad range of network-based services, including Internet-based e-commerce services.

place designer. A member of a place who can edit place layout and bookmarks.

place manager. A member of a place who can edit place membership, layout, and bookmarks.
place member. A individual or group who has joined or been granted access to a place. Place members have three levels of access to a place: manager, designer, and participant.

place template. A format for use in creating a place. The portal provides a set of default templates for creating various types of places. Portal administrators may allow users to create, modify, and delete new templates.

policy. A set of rules and actions that are required to be performed when certain events or status conditions occur in an environment. Policies are implemented using the IPL.

port. An end point for communication between applications, generally referring to a logical connection. A port provides queues for sending and receiving data. Each port has a port number for identification.

port type. An element in a Web Services Description Language (WSDL) document that comprises a set of abstract operations, each of which refers to input and output messages that are supported by the web service.

portal. A single, secure point of access to diverse information, applications, and people that can be customized and personalized.

portal administration. The place where portal administrators set and maintain basic collaboration permissions, place records, place membership records, and server settings for companion products for advanced collaboration.

portal artifacts. Stored in the portal file system. All deliverables of software development are usually artifacts (otherwise referred to as software components).

portal configuration. The Portal Configuration is stored in the portal configuration database. It consists of configuration entities. Each portal resource is represented by one portal configuration entity in the portal database.

portal extension artifacts. Artifacts that belong to components that are installed together with the portal but are not core portal components.

portal farm. A series of identically configured, stand-alone portal server instances that offer a way to maintain a highly scalable and highly available server environment.

portal member. An individual or group who has a user record in the portal directory (LDAP or other directory) and can log in to the portal.

portal solution release. The solution that is developed by the customer and is based on WebSphere Portal. The solution consists of portal configurations, portal artifacts, and portal extension artifacts and is shared between multiple users.

portal theme. The style element that gives a place a particular look. The portal provides several themes, similar to virtual wallpaper, which can be chosen when creating a place.

portlet. A reusable web module that runs on a portal server. Portlets have predefined roles such as retrieving news headlines, searching a database, or displaying a calendar.

portlet API. The set of interfaces and methods that are used by Java programs running within the portal server environment to obtain services.

portlet application. A collection of related portlets that can share resources with one another.

portlet container. A column or row that is used to arrange the layout of a portlet or other container on a page.

portlet control. A portlet registry setting that renders the outer frame for a portlet.

portlet framework. The set of classes and interfaces that support Java programs running within the portal server environment.

portlet palette. A web module that enables users to browse available portlets organized by category, search for individual portlets, and add them to a portal page by dragging to the desired location.

pre-rendered site. A complete website saved to disk in HTML that can be used as a live site and displayed to users using either Web Content Manager or a web server.
presentation template. A template that determines the structure of each web page in the site and which elements and components are displayed on each page. HTML defines the default properties and layout of the template.

producer definition. A set of interfaces that are defined for the producer portal. The producer definition can include the producer service description, the producer portal URL, and the security setup.

producer portal. A portal that provides portlets as a service so that other portals, called consumer portals, can use the portlets and make the portlets available to their users.

property extension database. A database that is used to store additional attributes that cannot be stored in the LDAP user registry.

provisioning. The process of setting up and maintaining a user's access to a system.

PSTN. See public switched telephone network

public group. A group of individuals, known to all portal users, that the administrator has created or that exists in the organization's corporate directory. Only administrators can modify and manage public groups.

public place. A shared place that is open to all portal users. The person who creates the place (and who automatically becomes the place manager) designates it as a public place during place creation.

public switched telephone network (PSTN). A communications common carrier network that provides voice and data communications services over switched lines.

pure server side portlet. A normal server side portlet that uses Java and Java server pages, but usually uses no JavaScript.

registered user. A portal user who has a user ID and password for logging in to a portal.

Representational State Transfer (REST). A software architectural style for distributed hypermedia systems like the World Wide Web. The term is also often used to describe any simple interface that uses XML (or YAML, JSON, plain text) over HTTP without an additional messaging layer such as SOAP.

REST. See Representational State Transfer

restricted place. A shared place that is open to only those individuals and groups whom the place creator (or place manager) adds to the place's membership list. The person who creates the place (and who automatically becomes the place manager) designates the place as a restricted place during place creation.

Rich Site Summary (RSS). An XML-based format for syndicated web content that is based on the RSS 0.91 specification. The RSS XML file formats are used by Internet users to subscribe to web sites that have provided RSS feeds.

role. A job function that identifies the tasks that a user can perform and the resources to which a user has access. A user can be assigned one or more roles.

RSS. See Rich Site Summary

rules-based personalization. Personalization technology that enables you to customize web content based on user needs and preferences, and business requirements.

search center. A portlet that enables site users to search for keywords.

search collections. A searchable collection of documents that can span multiple content sources.

search service. A service that is used to define the configuration parameters for a search collection. A search service can be local, remote, inside the product, or outside the product.
SecureWay Directory. An LDAP directory that can store user-related data, such as the user ID, the user name, and passwords.

security. The protection of data, system operations, and devices from accidental or intentional ruin, damage, or exposure.

security manager. A component that is responsible for authenticating user logins.

server side aggregation (SSA). Aggregation based on Java server pages that are executed on the Server.

service. In service-oriented architecture, a unit of work accomplished by an interaction between computing devices.

service description. The description of a web service, which can be defined in any format such as WSDL, UDDI, or HTML.

service provider. A company or program that provides a business function as a service.

service requester. The application that initiates an interaction with a web service. The service requestor binds to the service using the published information and calls the service.

service-oriented architecture (SOA). A conceptual description of the structure of a software system in terms of its components and the services they provide, without regard for the underlying implementation of these components, services and connections between components.

session bean. An enterprise bean that is created by a client and that usually exists only for the duration of a single client/server session. (Sun)

shared place. A place created for a community of people with a common purpose. Shared places can be public or restricted. The place creator (who automatically becomes the place manager) specifies whether a place is public or restricted during place creation.

Short Message Service (SMS). A service that is used to transmit text to and from a mobile phone.

single sign-on (SSO). An authentication process in which a user can access more than one system or application by entering a single user ID and password.

site area. A component contained in a site framework as a way to group similar content items. There can be several site areas within one site framework.

site framework. A structure that consists of a single top-level intelligent page or site area beneath which are stored other intelligent pages, site areas, and content items. The hierarchical set of intelligent pages and site areas define the navigational structure of the website.

site template. A pre-built sample site that can be used to streamline the process of developing a custom portal.

SMS. See Short Message Service

source portlet. The portlet that sends the information to other portlets.

SSO. See single sign-on

staging. The process of moving solution releases from development to production.

stand-alone static page. A static page that renders the complete browser content.

static page. A portal page that references a static layout.

static layout. The layout of a page that is based on a plain HTML page that may contain references to portlets.

subscribe. To register to access data published by another application or system.

subscriber. The consumer of a business service.
TAI. See trust association interceptor.

target portlet. The portlet that receives the information from the source portlet.

template library. The database, known as the Portal Template Catalog, that stores place template specifications and portlets' forms, subforms, and profiles.

theme. The style element that gives a place a particular look. The portal provides several themes, similar to virtual wallpaper, from which you can choose when creating a place.

transcoding technology. Content adaptation to meet the specific capabilities of a client device.

transport. The process or protocol mechanism of transferring an XML message or document between parties as part of a meaningful, reliable exchange. The most common transports for web services are SOAP/HTTP, SOAP/HTTPs, and SOAP/JMS.

trust association interceptor (TAI). The mechanism by which trust is validated in the product environment for every request received by the proxy server. The method of validation is agreed upon by the proxy server and the interceptor.

U

Uniform Resource Identifier (URI). A unique address that is used to identify content on the web, such as a page of text, a video or sound clip, a still or animated image, or a program. The most common form of URI is the web page address, which is a particular form or subset of URI called a Uniform Resource Locator (URL). A URI typically describes how to access the resource, the computer that contains the resource, and the name of the resource (a file name) on the computer.

Uniform Resource Locator (URL). The unique address of an information resource that is accessible in a network such as the Internet. The URL includes the abbreviated name of the protocol used to access the information resource and the information used by the protocol to locate the information resource.

Universal Description, Discovery, and Integration (UDDI). A set of standards-based specifications that enables companies and applications to quickly and easily find and use web services over the Internet. See also Web service.

URI. See Uniform Resource Identifier.

URL. See Uniform Resource Locator.

user group. A group consisting of one or more defined individual users, identified by a single group name.

V

W


WAP. See Wireless Application Protocol.

WAR. See Web archive.

WAR file. See Web archive.

Web archive (WAR). A compressed file format, defined by the Java EE standard, for storing all the resources required to install and run a web application in a single file.

Web content library. A library that stores items required for displaying or creating web content, such as workflow items, an authoring template, a presentation template, site areas, and content items. Most systems have at least two web content libraries, one for design items and one for web content.
Web crawler. A type of crawler that explores the web by retrieving a web document and following the links within that document.

Web service. A self-contained, self-describing modular application that can be published, discovered, and invoked over a network using standard network protocols. Typically, XML is used to tag the data, SOAP is used to transfer the data, WSDL is used for describing the services available, and UDDI is used for listing what services are available. See also SOAP, Web Services Description Language.

Web service endpoint. An entity that is the destination for web service messages. A web service endpoint has a Uniform Resource Identifier (URI) address and is described by a Web Service Definition Language (WSDL) port element.

Web service interface. A group of operations described by the content of a Web Service Definition Language (WSDL) 1.1 port element. These operations can provide access to resource properties and metadata. (OASIS)

Web service semantics (WSDL-S). A technical specification that defines a mechanism to associate semantic annotations with web services that are described using Web Service Description Language (WSDL).

Web Services Description Language (WSDL). An XML-based specification for describing networked services as a set of endpoints operating on messages containing either document-oriented or procedure-oriented information. See also Web service.

Web Services Interoperability Organization (WSI). An open industry organization that promotes web services interoperability across platforms, operating systems, and programming languages.

Web Services Policy Framework (WS-Policy). A model and framework for describing the capabilities, requirements, and general characteristics of a web service as a policy assertion or a collection of policy assertions.

Web Services Resource Framework (WSRF). The set of specifications that define the specific rendering of a Web Services Resource (WS-Resource), the association of that resource with the web service interface, and the messages that define the querying and updating of the properties of that resource.

widget. A graphical interface that features two or more reusable web applications (widgets) presenting seemingly disparate data in an understandable combination for a specific purpose.

wire. To connect two or more components or cooperative portlets so that they work together. In an application, wiring identifies target services; for portlets changes in the source portlet automatically update the target portlets.

Wireless Application Protocol (WAP). An open industry standard for mobile Internet access that allows mobile users with wireless devices to easily and instantly access and interact with information and services.

Wireless Markup Language (WML). A markup language based on XML that is used to present content and user interfaces for wireless devices such as cellular phones, pagers, and personal digital assistants.

workflow. The sequence of activities performed in accordance with the business processes of an enterprise.


WML. See Wireless Markup Language.

WSDL. See Web Services Description Language.

WSDL-S. See Web service semantics.

WSI. See Web Services Interoperability Organization.

WSRF. See Web Services Resource Framework.

XML. See Extensible Markup Language.
XSL. See *Extensible Stylesheet Language*.
Index

A
access
assigning authoring roles 263
assigning blog access 335
assigning user roles 255
assigning wiki access 338
libraries
overview 251
roles 252
setting access 39
roles 252
access control
REST 729
trace string 768
accumulator
configuring 396
when to run 409
accumulator utility
LikeMinds Recommendation Engine 416
Action beans
overview 422
reference 424
ActionEvent class 438
actions
creating do 362
creating select 351
developing remote 627
implement logging 423
set to default actions 365
administrator 514
administrators
in web team 513
always do action 362
APIs
custom authoring interfaces 268
rules engine 515
application object personalization 369
archetypes
enabling 399
overview 398
setting in cache 398
architecture
content acquisition 48
design 44
maintenance 50
security 39
site information 43
attachments
using REST to attach images 736
AttributeResource tag 226
attributes
using start and end 229
authentication
from remote servers 72
trace string 768
authoring
assigning roles 263
building a content system 52
authoring (continued)
customizing interfaces 268
deploying an environment 51
templates 173
tools
content items 269, 273
controlling behavior 271
web content viewers 271
authoring portlet
adding authoring tools 269, 273
creating custom pages 262
customization 263
hidden pages
overview 94
preferences 267
reserved portlet
overview 94
authoring templates
creating 173
customizing with JSP fields 175
overview 173
types 173
automatic syndication
types 758
B
backup and restore strategies 50
bindings
personalization 355
blogs
adding existing blogs to a page 334
adding to a page 333
assigning access to users 335
deleting 336
overview 332
viewing 335
browsers
checking capabilities 354
buildstats utility 415
business user 513
caching
archetypes 398
checking for mentors 398
connect tags 240
personalization in a resource
cache 525
using custom parameters 236
using parameters 237, 238
viewing settings 598
campaigns for personalization 369
cascading style sheets 187
categories
accessing a category count
implicit profiling example 353
rule example 424
beans reference 425
categories (continued)
creating category selection trees 216
within taxonomies 192
Category Count
rule example 425
CategoryEvent class 438
clear history tool 594
clear versions tool 596
Clickstream Engine
default recommendations 408
generating recommendations 390
minimum activities 407
overview 392, 407
recommendations 391
recomputing predictions 408
collaborative filtering 402
command line
creating syndication relationship 754
commands
trace string 768
component reference element 211
components
adding ratings widgets 246
adding tags widgets 245
overview 190
profiling strategies 191
using with REST 705
ConfigEngine tasks
create-page-nodes 101
create-virtual-portal-site-nodes 101
disable-managed-pages 103
enable-managed-pages 101
internalize-content-mappings 104
content
content items
adding existing blogs 334
adding with authoring tools 269, 273
customizing with profiles 191
mapping with presentation
templates 188
overview 189
purpose 181
content templates 173
content types overview 43
resources 348
contextual linking 236
coverage 403
credential vault
trace string 768
CSS 187
Custom listener classes 433
customizing
authoring pages
custom 262
settings 267
authoring portlet 263
authoring systems 262
authoring templates 173
fields with JSP files 173
personalization (continued)
action examples (continued)
select content 351
show page or portlet 359
update action 352
actions 351
API tips 519
application object 369
arithmetic expressions 359
best bets 411
binding examples
  binding 355
  if-then condition 356
  nested (advanced) 357
  nested (simple) 357
  simple 352
bindings 355
campaigns 369
Category beans 425
current
  request and session attributes 361
  request parameters 361
  session attributes 362
CustomLog beans reference
  method signatures 428
  RuleEvent class 437
details 347
element
  examples 213
  overview 212
federated documents 224
generic query framework
  generic query object 521
  overview 520
implementing category logging 425
interface 371
invoking rules programmatically 526
JAR files 520
LikeMinds Recommendation Engine
  configuring 397
LogEvent class 435
Logging beans 422
mentor pool 404
metadata 222
MovieSite sample 411
multiple sifters 405
multivalue properties APIs 518
overview 343, 345
PageView beans 428
PageViewEvent class 440
prediction quality values 410
preparing an application 513
profiler
  examples 332
  overview 364
  profiler examples
    arithmetic operation 355
    browser capability 354
    category count 353
    count of 354
    request and session attributes 355
    profiling examples
      nested profiler 353
      publishing rules 373
      quick profiler 365
resource
  cache 525
personalization (continued)
resource (continued)
  interface 518
  resources 347
  resources XML file
    sample 523
  RuleEvent class 437
rules
  access control visibility 348
  content spot mappings 367
  current date 361
  error conditions 350
  include only 362
  overview 350
  publishing 374
  publishing over SSL 380
  spot mappings 368
rules engine 515
samples
  helper class 684, 687
  installing 458
  LikeMinds 412
  resources XML file 523
sifter
  utility 415
  configuring mentors 401
  performance 403
  running multiple 405
  sleep time 405
trace string 768
tutorials
  changing the content spot
    mapping 504
  content resource classes 464
  copying the personalized list
    portlet 506
  creating a content rule 488
  creating a content spot 491
  creating a portlet 459
  creating a profiler rule 496
  creating rules for select action and
    binding 499
  creating user classes and a content
    spot 473
  creating workspace folders 486
  developing a portlet 455
  dynamic table 483
  enhancing a personalized
    portlet 493
  installing a portlet 485
  installing a sample database 458
  prerequisites 457
  setRequest calls 482
  translator class 495
  uninstalling the sample
    database 512
  user predictability 407
  user resources 348
  workload management 517
planning
  hardware 39
  maintenance 50
  personalization 343
  prototyping
    overview 37
  security
    overview 39
scopes (continued)
synchronization (continued)
manual 119, 120
tagging and rating content 117
scripting interface
trace string 768
search
elements 201
menus
creating queries 194
query examples 202
results
examples 204
Secure Sockets Layer (SSL)
publishing personalization rules
over 380
security
update security task 590
services
REST 693
service entry points 693
servlets
delivery 49
session attributes
current 362
example 355
set to rules 365
sifter
configuring 396
defined 403
sifters
configuring
for mentor selection 401
configuring number of users 403
configuring sift time interval 406
details 415
multiple
overview 405
preventing sifting the same user 405
pausing 403
setting sleep time 405
tuning performance 403
site toolbar
overview 121
sites
buildvisit utility
overview 415
repeated items 407
creating templates 173
defining default authoring settings 173
defining presentation template maps 188
design architecture 44
examples
brochureware 11
e-library 12
e-business 9
framework 181
intranet portal 8
partner 13
framework details 181
grouping content items 181
information architecture 43
launching 55
maintaining 50
sites (continued)
organization 181
overview 181
personalizing 345
planning a project 19
prototyping 37
using REST 717
website maps 43
SSO
trace string 768
start attribute 229
states
projects details 133
workflow stages 146
style
pages 187
style sheets
creating elements 210
relying 210
subscriber
syndication 752
synchronization
scopes
after syndication 119
manually 119, 120
tags and ratings 118
syndication
administering 758
between libraries 752
cloning
data before syndication 614
overview 613
creating relationship from command line 754
details 747
member fixer 589
overview 747
resetting a web event log 597
subscriber 752
troubleshooting 760
tuning 758
types 758
using RSS feeds 273
SystemOut.log
logging system events 774
T
tags
connect
details 240
filtering scopes 117
indenting elements 233
managing 117
PathCmpnt 222
scopes
filtering 117
synchronization 118
URLCmpnt 222
web content
plug-ins 244
taxonomies
categories in 192
element 216
overview 43
planning a hierarchy 192
templates
authoring 173
overview 173
site area 173
blogs and blog libraries 333
content 173
map examples 188
presentation
overview 184
using REST 712
wikis 336
time
element 211
trace string
access control 768
authentication 768
commands 768
credential vault 768
database 768
deployment 768
EventBrother service 768
finder service 768
layout model 768
loader service 768
mail service 768
mapping URLs 768
mapping service 768
navigator service 768
personalization 768
portal engine 768
portlet container 768
portlet environment 768
portlet load monitoring 768
portlets 768
registry service 768
scripting interface 768
self care 768
SSO (single sign-on) 768
WSRP consumer 768
WSRP producer 768
XML configuration interface 768
trace strings
IBM Web Content Manager 777
troubleshooting
crequiring
action 423
custom 428
custom log listeners 431
enabling 418
listeners 429
LogManager 429
PageView 429
personalization 429
resetting web events 597
rules 422
setting up ratings 427
syndication 760
troubleshooting tools
history information tool
gathering installation history 766
IBM Support Assistant
collecting data 764
version information tool
gathering version information 766
wpcollectortoolcollect
collecting data 764
U
URLCmpnt tag 222
URLs
  generating using tags 222
users
  developing content models 514
  member fixer task 583
  resources 348
  roles 255
  selection element 212
utility plug-ins 247

V
version information gathering 766
versions
  clearing history 596
  using REST 730
viewing
  blogs 335
  visibility rules 359
visits
  repeated items using the buildvisit utility 407

W
WCMContextHelper sample code 690
web content
  content spot exits 523
  creating 53
  delivery architecture 49
  developing content models 514
  helper classes for context 684, 687
  importing 53
    overview 48
  linking to 234
  mappings
    rule spot 368
  planning
    architecture 48
  portal pages
    custom pages 262
  resetting the event log 597
  resource collection 383
  road map for building 51
  security
    administrator 590
    architecture 39
    control elements 304
    update task 590
  site framework 181
tags
  plug-in 244
web content (continued)
viewers
  adding 547
  authoring tools 271
  delivery 49
  filtering results 117
  website 19
wikis
  adding
    existing 338
    new 337
  assigning access 338
  deleting 339
  purging 339
  template libraries 336
workflows
  projects 135
workflows
  adding later 593
  approvals 148
  control elements 301
  draft items 136
  examples 148
  expired items 143
  item status 135
  overview 146, 147
  published items 139
  update tool 593
  using REST 722
wpcollectortoolcollect
  collecting data 764
WSRP
  trace string 768

X
XML configuration interface
  trace string 768
  using to create syndication relationship 754