Bergfrid Marie Skaara
Peter Keung

eZ Publish
Advanced Content Management

eZ Publish Advanced Content Management
by Bergfrid Marie Skaara & Peter Keung

Printed in the United Kingdom / United States.

Publisher: Jennifer Zickerman
Editor and layout: Peter Keung
Contributors: Svitlana Shatokhina, Julia Shymova, Balazs Halasy
Illustrators: Thomas Hellstrøm, Vivien Anayian, Peter Keung
Cover Design: Hansi Von Grenzfeldt
Printing: Lightning Source

International Standard Book Number (ISBN):
978-82-92797-05-1

First printing: April 2008

eZ Press, a division of eZ Systems AS
Klostergata 33, N-3732, Skien, Norway

Phone: + 47 35 58 70 20
Fax: + 47 35 58 70 21
Email: info@ez.no
http://www.ez.no

Table of Contents

List of Figures

List of Tables

Preface

eZ Publish is an open source enterprise content management system that is used to build powerful, flexible web solutions, enabling people and organizations to share their information. With more than 40,000 downloads per month (over 2.3 million in total), eZ Publish is trusted by enterprises and organizations around the world.

This book was written to address the needs of content managers and webmasters. It provides a comprehensive advanced guide to managing content with eZ Publish.

This book builds upon the concepts from the *eZ Publish Content Management Basics* book. For readers who have not read that book, we include a review of the most important concepts and interfaces where relevant.

A considerable amount of material in this book was drawn from or based upon the eZ Publish technical and user manuals (see http://ez.no/doc), which were written by Balazs Halasy, Svitlana Shatokhina and Julia Shymova. Material from the Website Interface, eZ Find, eZ Flow, and eZ OpenDocument Format manuals (see http://ez.no/doc/extensions) was also included.

1. About eZ Publish

eZ Publish is a highly flexible and customizable content management system (CMS). It can be used to build everything from personal home pages to multi-national corporate web solutions with role-based multi-user access, online shopping, discussion forums and other advanced functions. Based on open source technologies and principles, it can be easily extended and customized to interact with other solutions. The following list presents a brief overview of the most important features and benefits of eZ Publish:

- Dynamic and customizable content structure
- Separation of content and design
- Powerful web-based **Administration Interface**
- Easy-to-use **Website Interface** for creating and updating content
- Powerful and flexible template language
- WYSIWYG Online Editor for text editing
- Platform independence (UNIX/Linux, Windows, OS X and so on)
- Flexible licensing (open source and professional license options)
- Built-in content versioning

- Support for multiple languages, translations and locales
- Built-in search engine
- E-commerce / Webshop functionality
- Built-in plugin system
- Role-based permission system
- Built-in workflow system
- Based on open and widely used standards (XHTML, XML, PDF, RSS, PHP, SQL, LDAP, WebDAV, SOAP and more)

The **Website Interface** is an extension to eZ Publish that makes it simple for users to create and maintain website content. It integrates into the front-end of the website, making content management intuitive and easy to learn. Most of the basic content management tasks can be performed through the **Website Interface**.

The **Website Interface** is implemented as a site package (called "ezwebin") that can be installed by the eZ Publish Setup Wizard.

Three other extensions are also featured in this book. eZ Flow is targeted at media sites and enables you to build complex page layouts and pre-plan the publication schedule to ensure a constant flow of rich content. eZ Find is an enterprise-ready search plugin, with more features than the built-in search engine. Among other things, eZ Find provides more search term options and relevancy ranking of results. The eZ Open Document Format extension enables you to import and export Microsoft Word and OpenDocument Text documents to and from your site.

2. Target audience and usage

This book will take you along the path to becoming an eZ Publish *expert content manager*. It is mainly written for content managers and webmasters, and strives to be easily comprehensible for non-technical readers. It can be used in a self-study environment or as a supporting resource for eZ Publish courses and certifications. On the technical scale, it lies somewhere between the *eZ Publish Content Management Basics* and *eZ Publish Basics* books.

For this book, it is an advantage to have some content editing experience, but it is not required. Beginners who have little or no experience with eZ Publish are encouraged to spend some extra time studying the first two chapters. The book attempts to ease people through the learning curve by providing a clear and well-structured overview of the system from a content manager's point of view. It includes detailed descriptions, examples and how-to procedures of the various editing and management tasks. It is not assumed that the reader has prior knowledge related to web development, programming or object-ori-

ented concepts. Where relevant, the necessary background information is provided. In those cases, the information is kept as simple as possible.

Throughout the book it is assumed that you are working with an existing eZ Publish installation. Installation, setup and technical material in general is covered in the book *eZ Publish Basics* and in the technical documentation at http://ez.no/doc.

This book is compatible with eZ Publish version 4.0 and **Website Interface** version 1.3. At the time of publication, the latest versions were eZ Publish 4.0.0 with the Online Editor version 4.2.4. In addition, eZ Flow version 1.0 and eZ Find version 1.0 for eZ Publish 4.0 are used.

In order to perform many of the procedures in this book, you need to log in to the **Administration Interface** with an Administrator user account. This provides the access permissions necessary for not only site administrators, but also for advanced content managers.

2.1. Special in-depth material

We have isolated some of the more technical material and presented it in *In Depth* blocks. These blocks provide more background information for interested readers, but can be skipped without interrupting the flow of the book or missing out on important concepts.

2.2. Example strategies

This book makes extensive use of examples, screenshots and illustrations. It includes tips and tricks, real-life examples and an example website. The examples assume a standard, default installation, and the examples within a topic build on each other, so that sometimes one example uses data entered in a previous example. However, the site is reset at the beginning of each chapter, using a fresh install. Therefore, the examples do not rely on any specific site data that was added or modified from previous chapters.

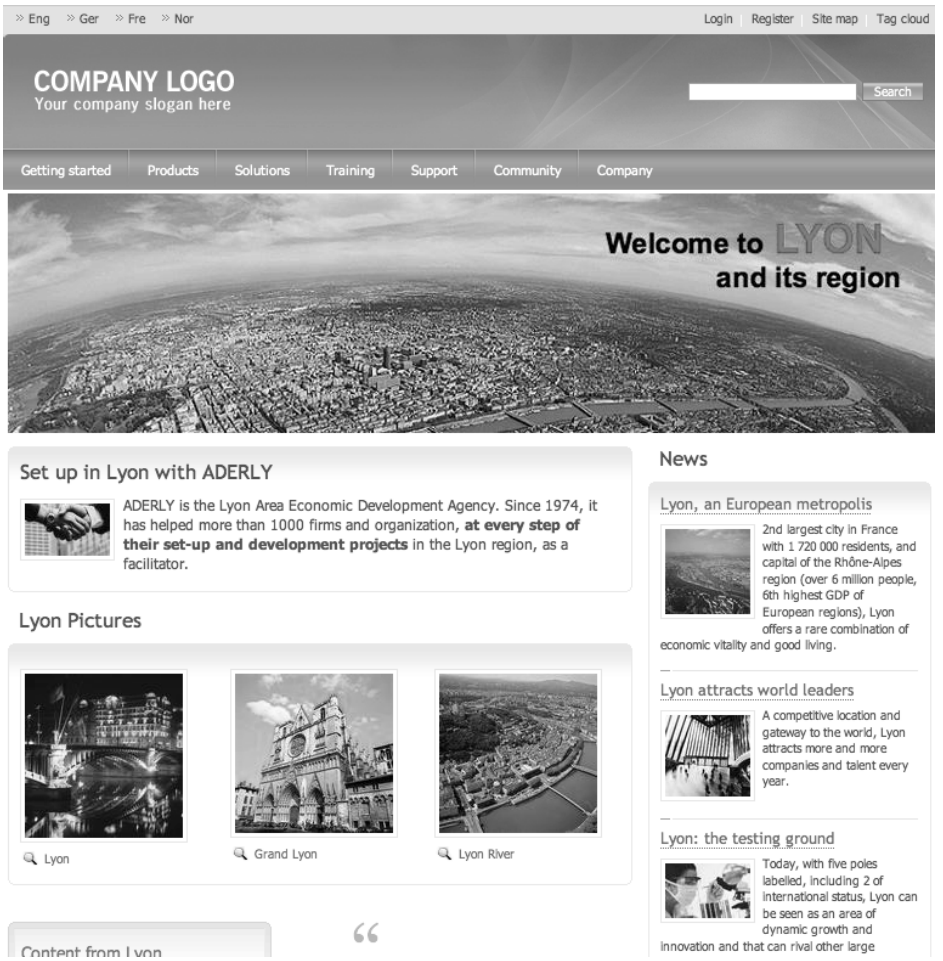


Figure 1. Example website

The example site is a multilingual site with four languages, a front-end interface to site visitors (the *public siteaccess*), a front-end interface for content management (the **Website Interface**) and a back-end interface for both content editing and site administration (the **Administration Interface**). The site has a Webshop, community forum, news article repository, image gallery and many other built-in eZ Publish features.

You can follow the examples by installing eZ Publish and the "ezwebin" site package (which includes the **Website Interface**). For the sections about eZ Flow, eZ Publish must be installed with the "ezflow" site package. The eZ Find and eZ OpenDocument Format extensions must also be installed for the appropriate chapters. Keep in mind that the buttons and interfaces, and the resulting screenshots and procedures, will vary from those found in this book if you use a version other than eZ Publish 4.0 and the **Website Inter-**

face 1.3. In most cases these variations will be minor and we encourage you to work with the latest version in spite of the variations.

For clarity of presentation, we have made some minor changes, such as ensuring that all content types are shown in the secondary / left menu of the **Administration Interface**. (By default, the content types that are shown are restricted, in order to enhance performance).

3. Contents

The chapters are organized as follows:

Chapter 1, *Introduction to Content Management Concepts*: background about eZ Publish features and concepts, including the separation of content and design, siteaccesses, templates, content objects and the node tree, user accounts, modules and views, notifications and requirements.

Chapter 2, *Graphical Interfaces and Tools*: explores the main interfaces and tools for editing and managing content: the **Website Interface**, the **Administration Interface**, the Online Editor, the context-sensitive pop-up menu and eZ Flow.

Chapter 3, *The Setup tab*: describes the layout and contents of the **Setup** tab, including eZ Publish URLs, cache management, the configuration model, the workflow system and extension management.

Chapter 4, *Sections*: outlines the features related to *sections* and shows how sections are used and managed.

Chapter 5, *User management and the permission system*: explores the layout and interfaces of the **Administration Interface User accounts** tab, identifies and describes the four parts of access control (users, user groups, roles and policies), and illustrates how these are managed in daily tasks.

Chapter 6, *Node visibility*: introduces the concept of *node visibility* and explains how to work with the *hide / show feature*.

Chapter 7, *Delayed publishing*: shows how to postpone publishing to a specific point of time.

Chapter 8, *Collaboration system*: describes the feature that makes it possible to force certain content to be approved by an editor before it is published. It explains the special-purpose interfaces and the approval process.

Chapter 9, *Web 2.0 - users as contributors*: introduces the concepts of Web 2.0, tag clouds and blogging. It explains the utility of blogs as well as the related content classes, attributes, and datatypes.

Chapter 10, *Structuring content*: shows how sitemaps can be used to get an overview of your content structure, how the node swapping feature can be used to change the type of a node while preserving the structure below it, and how to create and maintain a good content structure.

Chapter 11, *Search engines and finding content*: discusses the principles of the built-in search engine and the eZ Find search engine.

Chapter 12, *Working with multilingual sites*: describes language concepts and features, translation mechanisms, object availability and the language fallback system, and how to manage language settings for your site.

Chapter 13, *WebDAV*: provides the definition of WebDAV, its benefits and requirements, and demonstrates how to connect to WebDAV. It also shows how to browse, import, export, create, move, copy, rename and delete content via WebDAV.

Chapter 14, *OpenDocument Text and Microsoft Word documents*: describes how to create and edit content objects by importing and exporting OpenDocument Text and Microsoft Word files via the **Website Interface**, the **Administration Interface** and WebDAV clients.

Chapter 15, *Webshop*: explains the components of the e-commerce sub-system, how they relate, and how to manage them. It goes through the layout and interfaces of the **Webshop** tab, and explores some Webshop features in more depth, such as how to use multiple currencies on your site and how the Value Added Taxes (VAT) charging system works.

4. Typographical conventions

- Code samples, class names and HTML tags and their parameters are printed in `monospace font`.
- File names are printed in *`monospace italic font`*.
- Graphical user interfaces and their elements (such as button labels) are printed in **bold font**.

5. About eZ Systems

eZ Systems is the creator of eZ Publish, an open source enterprise content management system. eZ Systems was founded in 1999 with one simple and pragmatic idea: people working together, sharing experiences and ideas, can accomplish great things. With this simple idea, eZ has grown into a vibrant corporation with a thriving ecosystem of users, partners, customers and the “eZ crew” team members. As a company, we number 80 people with headquarters in Norway and offices in Germany, Ukraine, Denmark, France, Belgium, and North America. The eZ Ecosystem has thousands of members all over the globe.

We support our vision with a business model that combines innovations from the open source software development world with proven strategies from traditional software enterprises. Working together with our community and our partners, we serve ventures worldwide, ranging from single-person webshops to multinational enterprises, from regional governments to global humanitarian organizations.

5.1. Contact eZ Systems

Please visit our website at http://ez.no. Send questions, suggestions and comments to info@ez.no.

All books in the eZ Press series are available online at http://ez.no/store/books/ez_press.

6. Acknowledgments

Two groups of people have been instrumental in producing this book and its predecessor.

The first is the eZ crew, who exemplify sharing, encouragement, and patience in all that they do. From the developers to the marketers, they are all intelligent and willing to go above and beyond for their colleagues.

The second is the eZ Ecosystem and the open source community as a whole. An interest in open source is what brought me into the internet field, as the people in the community help to make technology accessible, personable, and easy to learn. It is a wonderful model for human interaction.

I have had a lot of fun working on these books, as Bergfrid and Jen are great at what they do, and the two groups mentioned above create an inspiring and healthy working environment. I hope that this book can be the useful contribution that the eZ crew and community deserve.

Peter Keung

When I was growing up, struggling to master something new, my father Bergfinn Vistnes always said "Aller Anfang ist schwer" (all beginnings are hard), and urged me to get my thoughts and ideas down on the paper without censoring myself. Drafts can be re-worked, but you have to overcome the fear that staring at a blank page infuses. Thank you dad for giving me the confidence to believe in my skills and ideas.

The beginning of this book was also the beginning of *eZ Publish Content Management Basics*. Thanks to Aleksander Farstad, Bård Farstad and Balazs Halasy (also known as Allman) for coming up with the idea initially. The contents have evolved over time. What we initially believed to be a single book of a few hundred pages turned out to be two separate full-bodied volumes. I've been working on this material for almost 2 years now,

and I could not have seen it through without the help and support of my co-workers at eZ Systems, my family and friends. Thank you all!

A huge thank you to the eZ Knowledge Products team for your support, contributions and motivation. I had fun spending time with you all during the eZ Conference 2007. Particular thanks to Balazs Halasy, Julia Shymova and Svitlana Shatokhina for their excellent work on the online documentation at http://ez.no/doc, from which this book pulls technical facts and inspiration.

Thanks to the eZ Press team for professional reviews, feedback, edits and layout of this book as well as the previous one. I know few people that work as hard as Jennifer Zickerman and Peter Keung, always willing to put in the extra hours to finish a task on time, even when the delay was not their doing in the first place. Thank you Jen, for letting me develop my writing skills, for making me understand and appreciate the process of creating technical books, and last but not least for your great personality and friendship. Big hugs, I'm gonna miss you! Thanks Peter, for your solid edits, your patience during our countless formatting discussions, your flexible working hours and for teaching me the meaning of the word "pertain". I will remember our "Friday morning at 7 - your time or mine?" meetings for a long time :)

Thomas Hellstrøm provided some of the illustrations in this book, which were re-used from Balazs Halasy's book *eZ Publish Basics*. In addition, Vivien Anayian and Peter Keung re-purposed and created additional illustrations specifically for this book.

I've had a lot of support from the eZ Ecosystem and I am grateful for your help and contributions. Thanks to André Rømcke, Bertrand Dunogier, Bjørn Reiten, Derick Rethans, Kristof Coomans, Kåre Køhler Høvik, Łukasz Serwatka, Ole Marius Smestad, Paul Borgermans, Thomas Johnsen, Doug Plant and Vidar Langseid for your technical support. Thanks to my co-workers at eZ Systems Nordic for sharing your knowledge and experience on eZ Publish. I would in particular like to thank Yelitza Jaramillo, who is both a skilled consultant and trainer, for all our sharing moments, helping me to re-think concepts and explanations, and for taking a true personal interest in your crew-mates. Thank you Nancy Waldock for your encouraging emails when times were rough. And, thank you so much all of you who sent me greetings when I was hospitalized last summer.

Thanks to my family for being patient when I was too busy or too tired to call or visit. I hope to see you more when things settle.

Many thanks to my good friend Anne-Helen Rasmussen for our cafe moments, dog walks and for listening when I need someone to talk to.

I would also like to thank Pedro Dias for putting up with my odd working hours and occasional bad moods after a rough day in the office. Thank you for always being at my side, supporting me no matter what and encouraging me to take on new challenges. I love you with all my heart!

Bergfrid Marie Skaara

Chapter 1. Introduction to Content Management Concepts

This chapter introduces the eZ Publish content management system and defines some key concepts that are necessary for the subsequent chapters in this book. Most (but not all) of the material in this chapter will be familiar to readers of the *eZ Publish Content Management Basics* book.

eZ Publish sites have several user interfaces for content editing and management in addition to simply displaying content. The Administration Interface is the advanced back-end interface set used mostly by advanced content managers, webmasters, site administrators and developers. It includes the **Object Edit Interface** for content editing operations. The Website Interface is accessible through the front-end of the website and is a simplified interface set mostly used by content editors. It includes the **Content Editing Interface**, which has the same purpose as the **Object Edit Interface**. (Note that the Website Interface is part of an optional add-on package.) In addition to the editing interfaces, each set has some special-purpose sub-interfaces, such as those for managing content versions and translations. These interfaces are described in Chapter 2.

To support various explanations in this book that build upon basic concepts from *eZ Publish Content Management Basics*, the appendix to this book contains a table that lists how to perform common content-related tasks. These tasks include copying, previewing and creating content in the Administration Interface and Website Interface.

This chapter is written for everybody working with an eZ Publish website, as well as those who want an introduction to content management systems (CMS) in general and eZ Publish in particular.

In this chapter, you will find information about the following:

- eZ Publish access requirements and user accounts
- Content management systems, and the separation between content and design
- Siteaccesses and their relationship to the Website Interface and the Administration Interface
- The content model: content objects, datatypes, the node tree, versions and translations
- Modules and views and the template system
- The email notification system

We recommend that you read the entire chapter in the order given. Readers already familiar with the contents of *eZ Publish Content Management Basics* can skip most of this chapter or use it as a quick review. The one exception is Section 1.3, "Modules and views", to which all readers should pay close attention. Chapters 1, 2 and 3 are prerequisites for the subsequent chapters in this book. If you are completely new to eZ Publish, we recommend that you skim through Section 2.2, "Website Interface" and Section 2.3, "Administration Interface" before reading this chapter, in order to familiarize yourself with the interfaces.

1.1. Content management system

The role of a *content management system* (CMS) is to organize and store content, regardless of type and complexity. Pieces of information in an eZ Publish installation are defined both by their *content class* (that is, the type of content, such as an image or an article) and their location in the *content hierarchy*. The main goal of such a system is to provide a well-structured, automated yet flexible solution, which allows information to be freely distributed and instantly updated across various communication channels (such as the internet, intranets and miscellaneous front- and back-end systems). Content can be formatted differently for computers, mobile devices, and more.

1.1.1. Separation of content and design

One of the keys to building a site that can be easily managed and maintained is to clearly separate data from presentation. *Content* is information that is organized and stored in a structured manner by eZ Publish. For example, content may be the components of a news article (such as the title, introduction, body and images). The presentation of content is determined by the design of a site. *Design* refers to the way the data is visually presented and includes the things that make up a web interface, namely layout and data markup: HTML, stylesheets (CSS), images that are not part of the content, typography, fonts, and so on. Refer to the Glossary for information on the different web markup standards. The separation of content and design provides the following benefits:

- Content editors and designers can work separately without conflicts
- Content can be published easily in multiple formats
- Content can easily be transferred and re-purposed
- Global redesigns and changes can be applied with simple modifications

Content and design elements combine to form a complete interface, called a *webpage*, as illustrated in the following diagram.

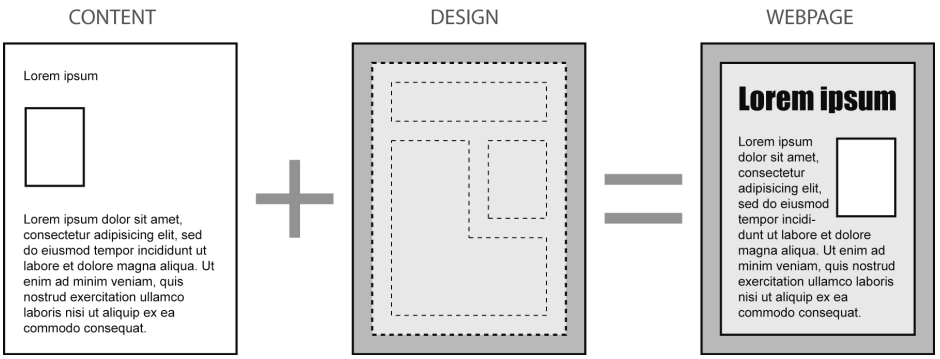


Figure 1.1. Content + design = webpage

An eZ Publish *website* encapsulates and includes everything that belongs to a particular area on the web: configuration settings, a database containing the content structure and actual content, content-related files and design-related files. In contrast, a webpage only refers to a specific part of the site.

In Depth: The eZ Publish directory structure

An eZ Publish installation has multiple sub-directories, each dedicated to a specific part of the system and containing a collection of logically-related files:

Figure 1.2. Directories below the eZ Publish root directory

When we refer to file locations in the subsequent chapters, the files can be found within one of these directories. Most commonly, we are referring to the `settings` directory for configuration files. We do not recommend that content managers access the file system of the web server to modify files or directories. In this book, some of the procedures will outline how to modify configuration files through the Administration Interface.

1.1.2. Siteaccesses

The content of a site can be displayed and modified in various ways. This is possible through multiple *site interfaces*. At a minimum, each site has two site interfaces: the Administration Interface and a public front-end interface for visitors. The latter can include the Website Interface for content management. An eZ Publish installation can have additional site interfaces, for example for viewing the content in a specific language, for an intranet, or even for hosting separate sites within a single installation.

A site interface refers to the visual presentation of a site, and is recognized through the address (also called the *URL*) in the browser. There is a one-to-one relationship between

a site interface and a *siteaccess*. A siteaccess is a collection of (override) configuration settings (see Section 3.5, “The configuration model”), of which the design and database are the most significant. The Administration Interface corresponds to the *admin* siteaccess, and the public front-end interface (possibly containing the Website Interface) corresponds to the *public* siteaccess. By default, these siteaccesses are named "ezwebin_site" and "ezwebin_site_admin" in sites using the Website Interface extension.

The following illustration shows how siteaccesses relate to other parts of the system.

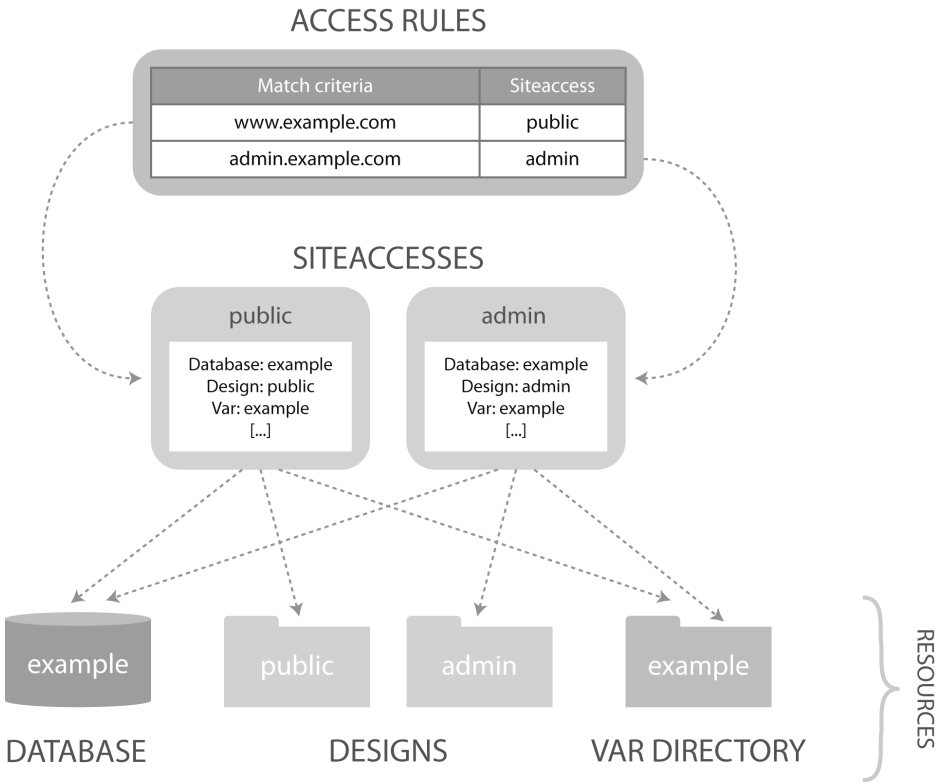


Figure 1.3. Siteaccesses related to access rules and resources

Multilingual eZ Publish sites (see Chapter 12, *Working with multilingual sites*) usually also have different site interfaces for some or all of the available languages. Thus, they have multiple public siteaccesses, one for each available language. In this case, the term "public siteaccess" is used to denote any of the language-specific siteaccesses.

1.1.3. Templates

Templates are the fundamental unit of site design in eZ Publish. Each template is a custom, extended HTML file that describes how particular types of content (or functionality) should be presented. It is the transforming unit that produces the site layout, either for

an entire webpage or some part of the page. Because templates are part of a design, and designs are one of the settings specified by a siteaccess, the templates used by a specific page is determined by the siteaccess in use. For example, an article might look entirely different in the Administration Interface compared to the public siteaccess. This is because different templates are used to position and style the article elements.

The eZ Publish template system is *component-based*. This refers to the fact that, in most cases, a webpage is created by combining several templates. At a minimum, eZ Publish always renders the main template, which is called the *pagelayout*. The pagelayout contains the `html`, `head` and `body` tags; together with the stylesheets (CSS), these dictate the overall look of a site. Among other things, the pagelayout specifies the title, logo, menu and footer that is presented for each webpage the system generates.

There are several different types of templates. In addition to the main template (the pagelayout), there are standard built-in templates in the core eZ Publish distribution and in extensions. Custom templates are also used. A template override mechanism dictates conditions for using alternative templates, substituting generic, general-purpose templates with custom, special-purpose templates. Conditions include the content class, a specific node, or the context (such as whether a specific piece of content is the main element on a page, an entry of a search listing, or part of a Frontpage). For example, an image may be displayed in different sizes in an article and an image gallery.

In addition to standard HTML syntax, developers can use eZ Publish-specific code in templates to, for example, extract and display content such as the attribute values of the requested object or the top-level containers for building a menu. This code is substituted when the page is rendered (see Section 1.3.3, “Module execution and web requests”). Therefore, when you view the source code in your browser, all you see is the HTML.

1.2. The content model

Unlike many other content management systems, eZ Publish does not use a "one-size-fits-all" approach. Instead of trying to fit data into rigid and predefined structures, the system allows the creation of custom structures using an object-oriented approach. The eZ Publish content structure is based on ideas borrowed from object-oriented programming languages like Java and C++. Superficially, *object-oriented* (OO) means looking at the world in terms of objects. In real life, people are surrounded by many objects, such as furniture, cars, pets, and other people. Each of these objects have traits.

eZ Publish supports various types of content, such as images, articles, multimedia files, forums and feedback forms. A content type is called a *content class* (or "class" for short), while a specific piece of content is called a *content object* (or "object" for short). A data structure is defined by a content class, and a content class is made up of class attributes. An *attribute* can be thought of as a field. The characteristics of the attributes are determined by the *datatypes*. Objects and attributes have names for human-readable reference, and an identifier for the system's internal reference (see the in-depth block below). The

system also stores various information related to each object, including timestamps for when it was created and last modified, as well as the object's status.

In Depth: attribute identifiers

Attribute identifiers are used to refer to attributes in places like template code and configuration settings. The identifier is usually almost identical to the attribute name, except that all letters are in lower case and spaces are replaced by underscores. For example, the name of an attribute of the `Folder` class is "Short description", while its identifier is "short_description". You can view attribute identifiers for a specific content class as described in Section 3.2, “Viewing and editing content classes”.

A *datatype* describes the type of value that can be stored in an attribute and is the smallest possible entity of storage. It determines how a specific type of information should be validated, stored and retrieved. eZ Publish comes with a collection of fundamental datatypes that can be used to build powerful and complex content structures. A complete list of datatypes can be found in the documentation at http://ez.no/doc/ez_publish/technical_manual. Some datatypes support validation in addition to just storing data. For example, attributes of the "XML block" datatype are checked for correct formatting according to the XML formatting standard before they are stored in the system.

Plain text content is text without any formatting, represented by the "Text line" or "Text block" datatype. In contrast, *rich text* content represented by the "XML block" datatype permits formatting. For example, you can mark a word to be shown in bold or italics, or apply more advanced formatting like tables, lists, inline graphics and links. Formatting is achieved by using XML (Extensible Markup Language) - a feature-rich and highly portable format for defining complex documents and data structures.

The following illustration summarizes and shows the relation between datatypes, attributes, content classes and content objects. The content objects are *instances* of the same content class, which means that they are of the same type but contain different data.

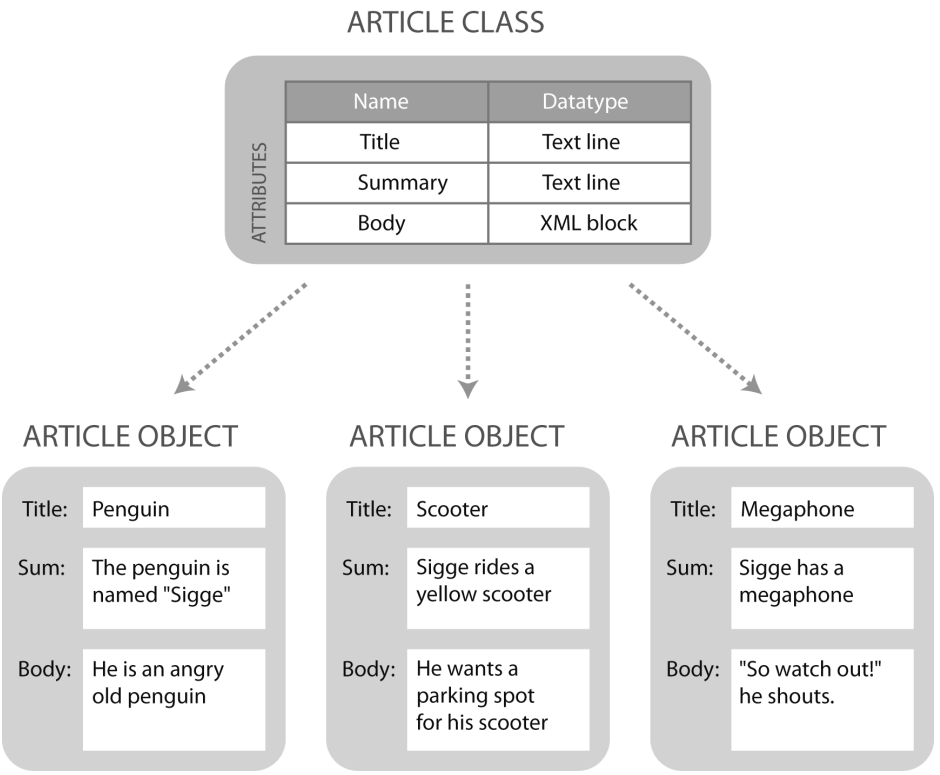


Figure 1.4. Datatypes, attributes, a content class and objects

A class belongs to one or more of the following categories, based on common features or usage. Your site might use some content classes differently than described here.

Table 1.1. Content class categories

Category	**Description**	**Examples**
Container	Organizes content within the node tree roughly in the same way as in a directory on a file system. For example, one or more articles are generally stored beneath a folder.	`Folder` class (structures the content hierarchy of your site); `Frontpage` class (displays content from other parts of the site)
Non-container	Objects that are not containers; they cannot have any sub items.	`Article` (single-page or multi-page; see the in-depth block below), `Image`, `Product` and `File` classes

Category	Description	Examples
User interaction	*User interaction* refers to site visitors submitting feedback, posting comments and participating in forum discussions.	`Comment`, `Feedback form` and `Poll` classes, as well as several classes for the forum and blog features
Information sharing	Used to share information on the site.	`Documentation page`, `Infobox`, `Event calendar` and `Gallery` classes
Multimedia	Most containers can hold not only images, but also other media types like Flash, Quicktime, RealVideo and Windows Media files. Each of these multimedia standards have a corresponding content class.	`Flash`, `Quicktime`, `Real video` and `Windows media` classes

In Depth: Single-page and multi-page articles

Articles can be either single-page or multi-page. *Single-page articles* are content objects of the `Article` class, while *multi-page articles* use the `Article (main-page)` and `Article (sub-page)` classes. A main-page object is a container for one or more sub-pages. When displayed, multi-page articles have a sidebar listing all of their sub-pages.

Technically, objects of the `Article` and `Article (main-page)` classes are containers, while objects of the `Article (sub-page)` class are not. Because you can enable comments for an article, and these comments are stored beneath the article, the objects need to be able to have sub items. Related objects such as embedded inline graphics are also usually placed beneath articles as sub items. Still, articles function more like non-containers, in that you would not put subtrees of content beneath them.

1.2.1. The content hierarchy and nodes

eZ Publish adds a layer of abstraction to classic object orientation through the node concept. *Abstraction* here refers to how the system enables you to think of a news article without knowing how it is internally handled. Content objects are wrapped inside (or *encapsulated*) and structured using content nodes (or "nodes" for short). An object has no structural capabilities of its own, but wrapping it inside a node provides a *location* at

which you can view the object's contents. The following illustration shows a simplified example of a node and its encapsulated object as it would have been represented inside the system.

Figure 1.5. Object - node relation

Nodes are organized in a *node tree*, also known as the *content hierarchy*, tying all published content together. The node tree is divided into five branches, of which the Content, Media and Users branches are the major branches. The following illustrations show a simplified example of how objects are referenced by nodes to make up a content node tree. The first shows the tree from a system perspective, whereas the other shows the same scenario from a user perspective.

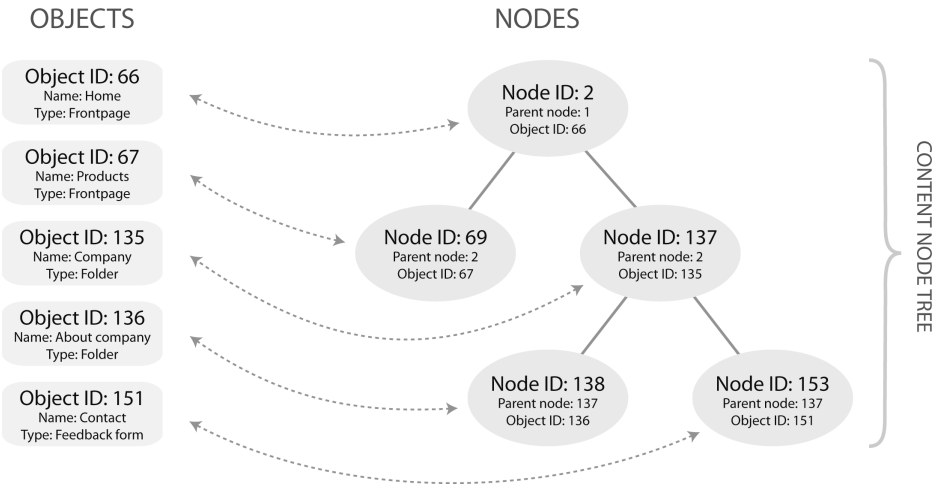


Figure 1.6. Objects, nodes and the content node tree

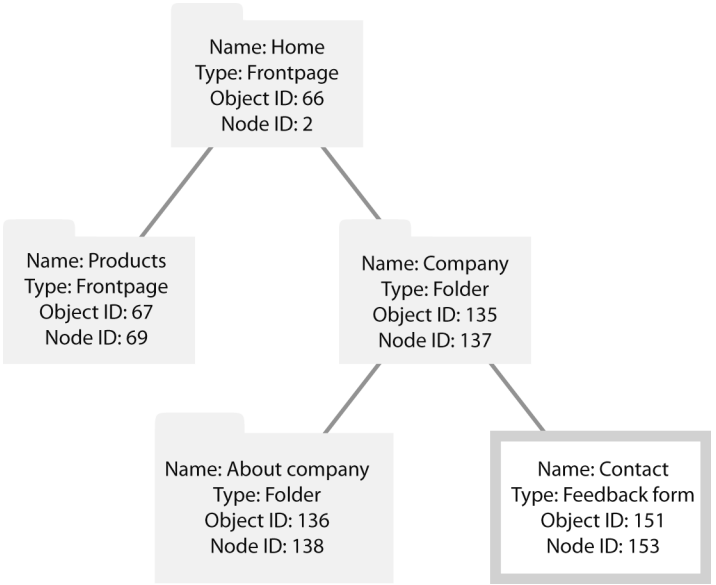


Figure 1.7. Content hierarchy

1.2.2. Content relationships

A *content relationship* implies that there is some form of connection between different pieces of content. In eZ Publish, there are three ways to express content relationships:

- As previously described, the content node tree ties all published content together in a content hierarchy. Content relationships here are between parent and child nodes.
- *Object relations* are relationships between objects in eZ Publish. For example, in addition to being part of an image gallery, images can be *embedded* in the body of an article. Also, frontpages can embed content pulled from any part of the node tree.
- External links connect objects in eZ Publish to other webpages on the internet and are created by using objects of the `Link` class, hyperlinks directly in attributes of the "XML block" datatype or attributes of the "URL" datatype. Internal links connect objects within eZ Publish. They are a type of object relation and can only be used in attributes of the "XML block" datatype.

1.2.3. Version management and content life cycle

A content object has at least one *version*. Every time you save changes to an object, a version is saved. Versions are also known as the *second dimension* of content objects.

The system stores information related to each version, including a version number, timestamps with when it was created and last modified, the user who created the version and the version's status.

The life cycle of a content object consists of several statuses - "Draft", "Published" and "Archived" - as shown in the illustration below. Both content objects and versions have statuses. Since an object can have several versions, there may be a direct correspondence between the status of an object and the status of a version, or they may have different statuses. The version with the "Published" status is the one that is displayed to site visitors and referred to by the node. There can only be one published version for a particular object at a given time. A draft contains content that has been saved but not yet published. Archived versions represent old content that you can revert to if needed.

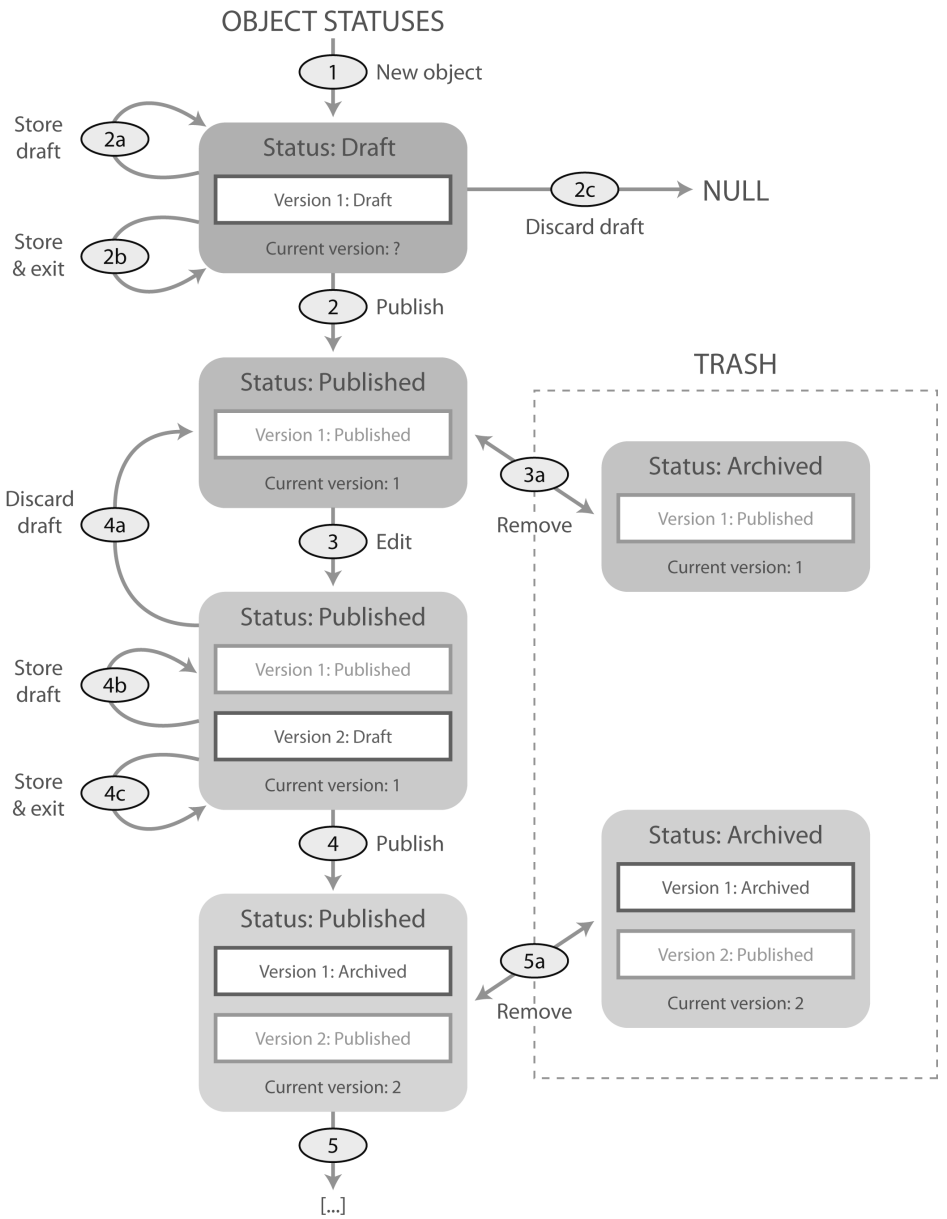


Figure 1.8. Life cycle of a content object

There are two additional version statuses: "Pending" and "Rejected". The former is used by the collaboration system (see Chapter 8) and the delayed publishing feature (see Chapter 7). The latter is currently not used in standard eZ Publish installations.

1.2.3.1. Possible editing actions on versions

Tip

A version can only be edited if it has the status of "Draft" and it can only be edited by the user who initially created it.

The table below shows available and non-available actions for versions, according to their statuses. It is important to keep in mind that these relate to version statuses, not object statuses. You can view and manage the versions of an object in the **Version history** interface (accessed from the **Object Edit Interface**, the pop-up menu, or the Website Interface; see Chapter 2, *Graphical Interfaces and Tools*).

Table 1.2. Actions on version statuses

Status	Available	Not available
Draft	• can be edited (by the same user who initially created it) • can be published (the previous published version will become archived) • can be deleted / discarded • can be stored for later	• cannot be archived (without first being published)
Published	• can be copied, becoming a new draft • can become archived when a more recent version is published	• cannot be edited • cannot become a draft
Archived	• can be copied, becoming a new draft	• cannot be edited

When a published or an archived version is copied, the status of the new version is set to "Draft" and thus it becomes editable. When and if the new draft is published, the system automatically sets the status of the previously published version to "Archived" and the new draft becomes the published version. For more information on versions, refer to *eZ Publish Content Management Basics*.

1.2.3.2. Publishing and unpublishing

The terms "publish" and "unpublish" are subject to some misconceptions that we will clarify here, with reference to Figure 1.8, “Life cycle of a content object”. First, to *publish a content object* refers to the transition from the "Draft" to "Published" object status, when you click the **Send for publishing** button for the first time in an object's life cycle. The object is then part of the content node tree and visible at the public site. Second, to *publish a version* means to publicly make available some changes to a content object, unless it is the object's first version. In that case, publishing a version is the same as publishing the object.

Finally, technically there is no such thing as unpublishing. In other words, a published object cannot become a draft object. However, the term "unpublish" is commonly used to describe taking an object off the public site. This is usually done by deleting the content object (clicking the **Remove** button), or hiding the content object (see Chapter 6, *Node visibility*). By default, objects are put in the trash when they are deleted, and their statuses go from "Published" to "Archived". When the trash is emptied, the object is completely erased from the system. In contrast, objects that are hidden still have the "Published" status, and are still part of the content node tree, but are not displayed in public siteaccesses. You can also take content off the public site by moving it to a restricted section (see Chapter 4, *Sections*).

We recommend that you use the terms "delete", "remove", "hide" or "move" rather than "unpublish". Within technical documentation, "unpublish" and "delete" commonly refer to the same action.

1.2.4. Multilingual support: translations

Each version further consists of at least one *translation*. The translation layer makes it possible to represent content in multiple languages. It is this final layer that holds the attributes of an object, such as the title of an article. When you translate content, you provide language-specific text values for the different input fields in edit mode. By toggling translator mode on and off, you can have the source text displayed above the input fields, as explained in Section 2.3.5, “Object Edit Interface”.

Multilingual eZ Publish sites often have links in the top right or left corner, for selecting the language in which to display content. Selecting a language link here corresponds to selecting which language-specific siteaccess you want to use.

1.3. Modules and views

Modules and views are important architectural concepts that describe how eZ Publish produces the output shown in your web browser. Some basic knowledge about modules and views is recommended as background information for subsequent topics related to eZ Publish URLs (see Section 3.3, “URL management”), workflows (see Section 3.7,

"The workflow system") and access permissions (see Section 5.5, "Roles and policies"). Note that the presentation given here has been simplified to meet the needs of content managers. Some of the technical details have been deliberately omitted.

1.3.1. Modules

A *module* provides an interface for web-based interaction, providing access to eZ Publish core functionality. The requested URL tells eZ Publish which module it should execute in order to process the request. eZ Publish comes with a built-in collection of modules, including the "user" module (which provides features such as logging users in and out and changing passwords) (see Chapter 5); the "shop" module (which provides webshop features such as the basket, sales statistics and wish list) (see Chapter 15); and the "collaboration" module (which handles the collaboration system) (see Chapter 8). It is possible to extend the built-in collection of modules by creating or using module extensions.

Tip

As a rule of thumb (although not entirely technically correct), content managers can think of a module as corresponding to a group of tasks or area of management within the Administration Interface, for example for working with content or webshop management.

The most important and most utilized module is the "content" module. This provides access to the eZ Publish content engine and enables features such as displaying, editing and translating content objects, as well as managing the node tree. You might have noticed that when you are editing a content object, the URL contains the string "content/edit". Also, when visitors are browsing your site, they are usually interacting with the "content" module.

1.3.2. Views

A module consists of a set of *views*. It is these views that enable you to reach the functions provided by the module. In other words, a view represents an interface to a module. For example, the "edit" view of the "content" module lets you modify the contents of an object, while the "versionview" view of the same module presents the **Version history** interface (used to manage an object's versions).

In order to pass information to a view, such as which object to edit, *view parameters* are often used. View parameters are simply appended to the URL when a page is requested, as explained in Section 3.3.1, "eZ Publish URLs".

For most of the content management tasks in this book, you do not have to type in the specific URL for the module and view - you simply click on the correct links and buttons in the interfaces. However, it is quite useful to have some understanding about the URLs.

1.3.3. Module execution and web requests

Every time a web browser accesses an eZ Publish site, it indirectly interacts with a module:

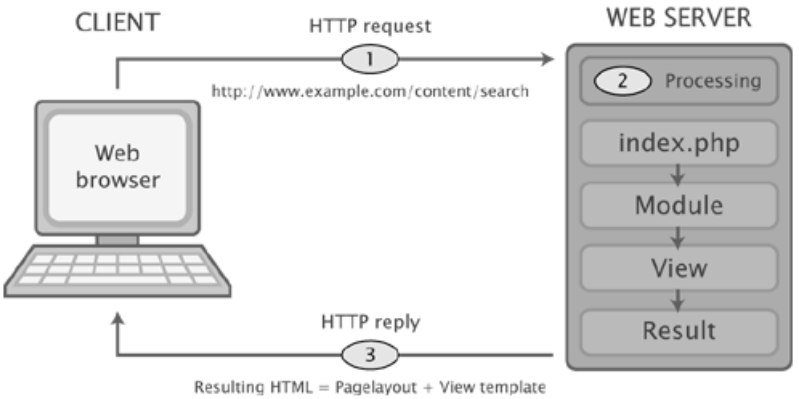


Figure 1.9. Handling web requests

In a default request, eZ Publish needs to know which module to run and which view to execute. The requested URL tells the system this information. For example, "http://www.example.com/index.php/user/login" executes the "login" view of the "user" module. (You will notice that the URL does not always follow this pattern; this will be explained in Section 3.3.1, “eZ Publish URLs”.)

Upon every access, a module is run and one of its views is executed. The view returns a result to the module. The result of module execution can then be used by the template system:

Figure 1.10. Result of web request

Processing and presenting this result is done by the template system. The final outcome is HTML code sent back to your browser, where it will be handled according to web standards. See the technical manual at http://ez.no/doc/ez_publish/technical_manual or the *eZ Publish Basics* book for more detailed information about templates.

1.4. Notifications

The built-in eZ Publish *notification system* enables users to be informed via email about certain events, such as content being updated or added to the site (these are subtree notifications), or collaboration actions, such as when an article is waiting for editorial approval. This section primarily addresses subtree notifications; collaboration notifications, which are email notifications about new collaboration messages, are described in Section 8.4, “Collaboration notifications”.

Subtree notifications can be sent by the *email notification service* in response to changes to a part of the content node tree. For example, email notifications can be sent when a new node is published, or when the contents of an existing node are modified. It is currently not possible to distinguish between these two cases for triggering notifications.

Notifications are personal in the sense that a particular notification email is sent to only those users that have signed up for it. However, the message will be identical for all the users that are subscribed to the same update.

A useful application of the notification system is to subscribe to notifications about changes to content for which you are responsible. In other words, you can track activity that occurs on a particular topic (or other content) without having to constantly review the site.

The following sub-sections assume that you are familiar with the Website Interface and Administration Interface, which are explained in more detail in Chapter 2. You can always refer back here when needing to use the notification system, such as when you need to sign up for notifications.

1.4.1. Signing up for notifications

Signing up for subtree notifications means to mark content to be monitored for changes. In the public siteaccesses, marking can be done when creating content (such as marking the **Notify me about updates** checkbox when creating a discussion forum topic), and when viewing content (such as clicking the **Keep me updated** button when viewing a discussion forum or discussion forum topic). The label and location of these buttons or checkboxes may vary depending on the type of content, and on whether or not your site uses custom templates.

You can also sign up for subtree notifications directly through the **Notification settings** interface, which is described below. In addition, in the Administration Interface, editors and administrators can click "Add to my notifications" from the context-sensitive pop-up menu (see Section 2.4, “Context-sensitive pop-up menu”) in the three left-most tabs.

Tip

To receive notifications about all content changes on your site, subscribe to the top-level node of the Content branch (in the **Content structure** tab).

To receive notifications about new user registrations, subscribe to the top-level node of the Users branch (in the **User accounts**) tab.

1.4.2. Accessing and managing notifications

Logged in users can view and manage their personal notification subscriptions via the **Notification settings** interface (illustrated in Chapter 8 with regard to the collaboration system) from both the front-end of the site (click the **My profile** link in the top right, then click the **My notification settings** link) and the Administration Interface (click the **My notification settings** link in the **My account** tab). In addition, the user must have access to the "notification" module.

The **Notification settings** interface is used to modify three aspects of this feature: the frequency of notifications; whether or not notifications are sent individually or combined into one digest; and removing the notifications setting from pages for which you previously requested notifications.

Tip

Remember to remove notifications on material you no longer wish to be notified about, in order to avoid receiving unwanted emails. To stop receiving notifications for a particular node, mark the checkbox beside the corresponding node in the **Notification settings** interface, then click the **Remove** button.

1.5. eZ Publish access requirements

The Website Interface and Administration Interface of eZ Publish can be accessed regardless of your physical location and / or operating environment as long as the following requirements are met:

- The client machine must be able to access the eZ Publish server via the internet or internal network.
- The client machine must be equipped with a supported web browser.

In Depth: Web standards and technical requirements

eZ Publish uses the XHTML 1.0 Transitional standard combined with *Cascading Style Sheets* (CSS). In addition, *JavaScript* (a lightweight programming language that extends the functionality of XHTML) is used to create a more user-friendly environment. Among other things, JavaScript is used to create an interactive tree-representation of content in the Administration Interface, and it enables the usage of *context-sensitive pop-up menus* and the Online Editor both in the Website Interface and the Administration Interface. *Ajax* (short for Asynchronous JavaScript And XML) is a web development technique that uses JavaScript to further improve the user experience. For example, Ajax is used to enhance the performance of the dynamic tree menu in the Administration Interface. If your browser supports JavaScript, it also supports Ajax.

Although JavaScript (and Ajax) enhances the interfaces, the client does not need to have support for JavaScript enabled. Thus the interfaces will function correctly in non-JavaScript browsers, but will not be as feature-rich as intended. You can still access the same operations through other buttons and links or by requesting specific URLs. However, note that if JavaScript is disabled, the Online Editor (see Section 2.1, “Online Editor”) will not be available. Instead, you will have to manually edit XML tags for the affected object attributes.

1.5.1. Supported browsers

The Administration Interface and Website Interface are designed to work in all major browsers on all of the common operating systems. We test eZ Publish with the following browsers:

- Internet Explorer 6 + (except with the Online Editor on Windows Vista)
- Mozilla Firefox 2 +
- Opera 9 + (except with the Online Editor)

Note that the interfaces should work in any browser that is capable of rendering XHTML 1.0 Transitional and supports CSS. If CSS is not supported, the system will appear without most of the design or layout, but will still be accessible.

At the time of publication, the Online Editor does not work with Internet Explorer on Windows Vista. The reason for this is that Microsoft removed the DHTML component from Windows Vista from Release Candidate 1 and later releases. The Online Editor depends on this component. Therefore, if you have Windows Vista, we recommend that you use Mozilla Firefox.

The Online Editor also makes use of JavaScript technology (see the in-depth block above). This means that JavaScript must be installed and enabled in your web browser for it to be available.

1.5.2. eZ Publish user accounts and login

In eZ Publish, access to content management operations is restricted and based on user authentication. To edit and manage content, your account must belong to the Editor or Administrator group or have corresponding permissions. User management is covered in Chapter 5.

To log in to the Website Interface, locate the **Login** link in the top right corner of the site. Clicking this link will bring up the **Login** interface, shown in the following screenshot.

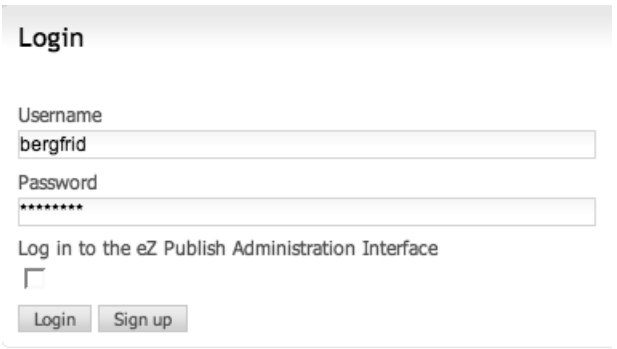


Figure 1.11. Website Interface login

To log in to the Administration Interface, access the URL for the admin siteaccess. This will display an almost identical **Login** interface to the one shown above. You can also log in to the Administration Interface through the login screen of the Website Interface by marking the **Log in to the eZ Publish Administration Interface** checkbox.

To log out, simply click the **Logout** link found in the top right corner of the Website Interface, or in the **Current user** area on the right side of the Administration Interface.

1.6. Summary

Below we have compiled a list of the most important concepts in this chapter. We encourage you to review this list and refer to the Glossary for a full listing and definitions of terms in this book.

- Content management system (CMS): used for collecting, managing, displaying and publishing content.
- Content: information (actual data) organized and stored in a structure.
- Design: visual presentation, layout and data markup.
- Siteaccess: A *siteaccess* is a collection of (override) configuration settings. The Administration Interface corresponds to the *admin* siteaccess, and the public front-end interface (possibly containing the Website Interface) corresponds to the *public* siteaccess. There is a one-to-one relationship between a *site interface* and a siteaccess.
- Template: a custom HTML file that determines presentation of a specific type of content.
- Content object / class: A content type is called a *content class* (or "class" for short), while a specific piece of content is called a *content object* (or "object" for short). A content class can be thought of as a structural blueprint for a particular type of content.

The properties of a class are referred to as *attributes*. A *datatype* is the smallest entity of storage and determines the validation, storage and retrieval for the value held by the attribute.

- Version: A content object consists of one or more *versions*, each with a version number used to identify it and a *version status* (draft, published, archived, pending or rejected). The version with the "Published" status is the one that is displayed to site visitors and referred to by the node. A draft contains content that has been saved but not yet published. Archived versions represent old content that you can revert to when needed. Each version consists of at least one *translation*, which is a representation of the content in a specific language.

- Content node: Content objects are wrapped inside (or encapsulated) and structured using *content nodes* (or "nodes" for short). Nodes allow objects to be placed in the node tree.

- Object life cycle: "Draft", "Published" and "Archived" are statuses indicating where an object is in its life cycle.

- Content node tree: Nodes are organized in a *node tree* divided into five branches, of which the Content, Media and Users branches are the major branches. The node tree is also known as the *content hierarchy*.

- Module: offers an interface for web-based interaction, providing access to eZ Publish core functionality. The most important and most utilized module is the "content" module. A module has a set of views that enable you to reach the functions provided by the module. In other words, a view represents an interface to a module. For example, the "edit" view of the "content" module lets you modify the contents of an object.

- Notification system: enables users to be informed via email about certain events, such as content being updated or added to the site (subtree notifications), or collaboration actions, such as when an article is waiting for editorial approval.

Chapter 2. Graphical Interfaces and Tools

This chapter gives an overview of the Website Interface and the Administration Interface, focusing on their respective layouts, tools and editing interfaces. We will also introduce the eZ Flow extension, which is targeted at media and news websites, and enables you to build complex frontpage layouts and pre-plan the publication schedule.

This chapter is written for everybody working with an eZ Publish website, and is foundational material for the concepts described in this book.

Recall that:

- A content type is called a *content class* (or "class" for short), while a specific piece of content is called a *content object* (or "object" for short). The properties of that content are referred to as *attributes*.
- Content objects are wrapped inside (or encapsulated) and structured using *content nodes* (or "nodes" for short). Nodes are organized in a *node tree* divided into five branches, of which the Content, Media and Users branches are the major branches.
- A site interface refers to the visual presentation, and is recognized through the address in the browser. There is a one-to-one relationship between a site interface and a *siteaccess*. A siteaccess is a collection of (override) configuration settings. The Administration Interface corresponds to the *admin* siteaccess, and the public front-end interface (possibly containing the Website Interface) corresponds to the *public* siteaccess.

In this chapter, you will find information about:

- The layout, components and functions of the Online Editor
- Content management tasks in the Website Interface
- Content management, site management and other tasks in the Administration Interface
- Functions accessed through the context-sensitive pop-up menu
- How to create and manage dynamic frontpages with the eZ Flow extension

We recommend that you read the entire chapter in the order given. Readers already familiar with the contents of *eZ Publish Content Management Basics* can skip most of this chapter or use it as a quick review. The exception is Section 2.5, “Frontpage management with eZ Flow”, as that extension was recently released and applies to those working with media sites.

2.1. Online Editor

The eZ Publish Online Editor (or "OE" for short) is an extension that enables you to create formatted text using an intuitive interface. It is named "ezdhtml" for system reference. With the Online Editor, you can easily format long passages of text for your articles, product descriptions, weblogs and so on by editing attributes of the rich text "XML block" datatype. The Online Editor's WYSIWYG interface is similar to the interfaces in word processors. There is no need to know XML or manipulate tags manually. When you click the necessary buttons, the Online Editor automatically converts your instructions into valid eZ Publish XML code.

You are working in *edit mode* when you are viewing some content in the **Content Editing Interface** (in the Website Interface) or in the **Object Edit Interface** (in the Administration Interface). This usually means that either the **Edit** or **Create here** button has been clicked. The Online Editor facilitates the manipulation of content attributes in edit mode, and is an integrated part of your eZ Publish content editing environment.

The table below shows the relationship between the interfaces and resources used in edit mode.

Table 2.1. Edit mode - involved interfaces

Main interface	**Edit interface**	**Resource**
Website Interface	Content Editing Interface	Online Editor
Administration Interface	Object Edit Interface	Online Editor

When the Online Editor is activated for your site (see Section 3.6, “Extension management”), it will appear every time you edit text that is stored using the "XML block" datatype (see Section 1.2, “The content model”). Note that the Online Editor is not a stand-alone text editor; in other words, you can only edit eZ Publish content with it, and only when you are logged into the system.

The Online Editor has five components, three of which are shown in the screenshot below:

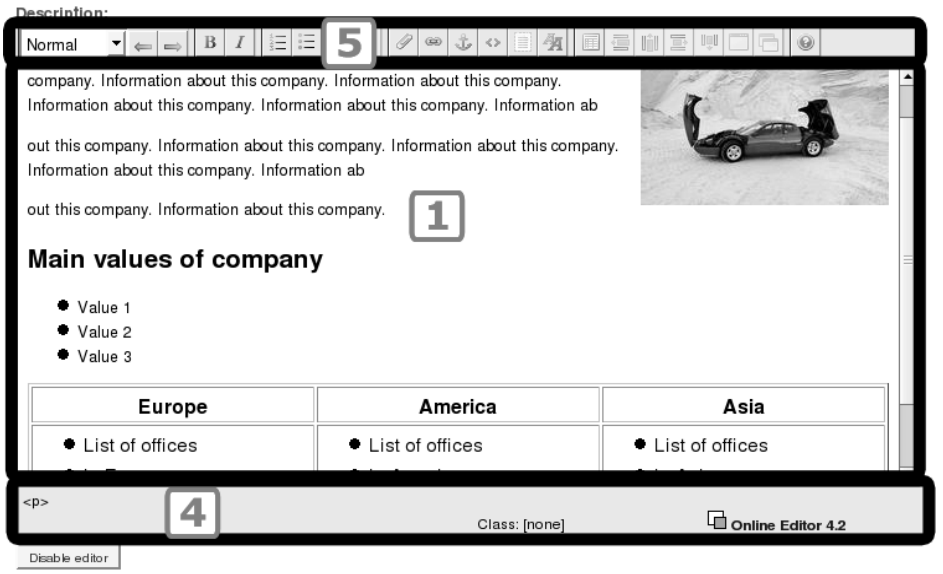


Figure 2.1. Online Editor components

1. The *OE text area* contains an editable visualization of XML code as formatted text.

2. The *context-sensitive pop-up menu* (described in more detail later) can be accessed by right-clicking anywhere in the **OE text area**. In addition to classic commands like "undo", "cut" and "copy", the pop-up menu contains specific items that can be used to manipulate the selected / current element.

3. *Modal dialogs* provide access to additional editing tools, depending on the current setting. For example, when you insert a link, a modal dialog is displayed where you can specify various link details.

4. The *OE status bar*, located directly below the **OE text area**, displays useful information about the element where the cursor is currently located.

5. The *OE toolbar* contains tools for editing the content that is displayed in the **OE text area**. The buttons are very similar to those found in many word processors. The following screenshot shows the different buttons:

Figure 2.2. OE toolbar buttons

Hover your mouse over the buttons in the **OE Toolbar** to display tooltips describing each button's function. When a particular button is disabled, this means that the button's function is not applicable in the current context.

Refer to the *eZ Publish Content Management Basics* book for a detailed walk-through of content editing procedures in general and the Online Editor in particular.

Tip

At the time of publication, eZ is working on a new version of the Online Editor. In addition to some new features, the most prominent changes are more user-friendly modal dialogs and replacement of the pop-up menu by buttons on the **OE toolbar**. More information will be available on http://ez.no when the new Online Editor is released.

2.2. Website Interface

Most basic content management tasks can be performed via the front-end Website Interface, which is part of the "ezwebin" extension. It has a simpler graphical user interface (GUI) than the Administration Interface (described in Section 2.3, “Administration Interface”). The screenshot below shows the Website Interface.

Figure 2.3. Website Interface

2.2.1. Navigating the Website Interface

The Website Interface is an enhancement to public siteaccesses, providing additional functionality to editors and administrators when they are logged in. In other words, it is what site visitors see, plus more. For simplicity, any public siteaccess on a site using the "ezwebin" extension is commonly referred to as the Website Interface. Navigating the Website Interface is done as you would normally browse your site: by clicking around (via menus, links, and the sitemap) and searching.

2.2.2. Website Toolbar

The **Website Toolbar**, seen in Figure 2.3, “Website Interface” above (below the main menu and above the banner), is the main feature that makes the Website Interface much simpler than the Administration Interface. It provides easy access to content editing op-

erations related to the content that is being viewed. Note that the **Website Toolbar** iteself is not used for site navigation; instead, it is displayed on each page as you use the standard navigation elements to move around the site.

The buttons on the **Website Toolbar** change depending on the context. For example, when you are viewing a content object, the **Website Toolbar** usually includes a button to edit the current object and another to create an object beneath the current node in the node tree. (We say "usually" because this also depends on user permissions and the settings for the particular content class.)

2.2.3. Content Editing Interface

When you click the **Edit** or **Create here** button on the **Website Toolbar** when viewing a page, the **Content Editing Interface** is opened. The screenshot below shows the **Content Editing Interface** for an article after the **Edit** button has been clicked. Note that we have cropped out the fields below the summary due to print size limitations.

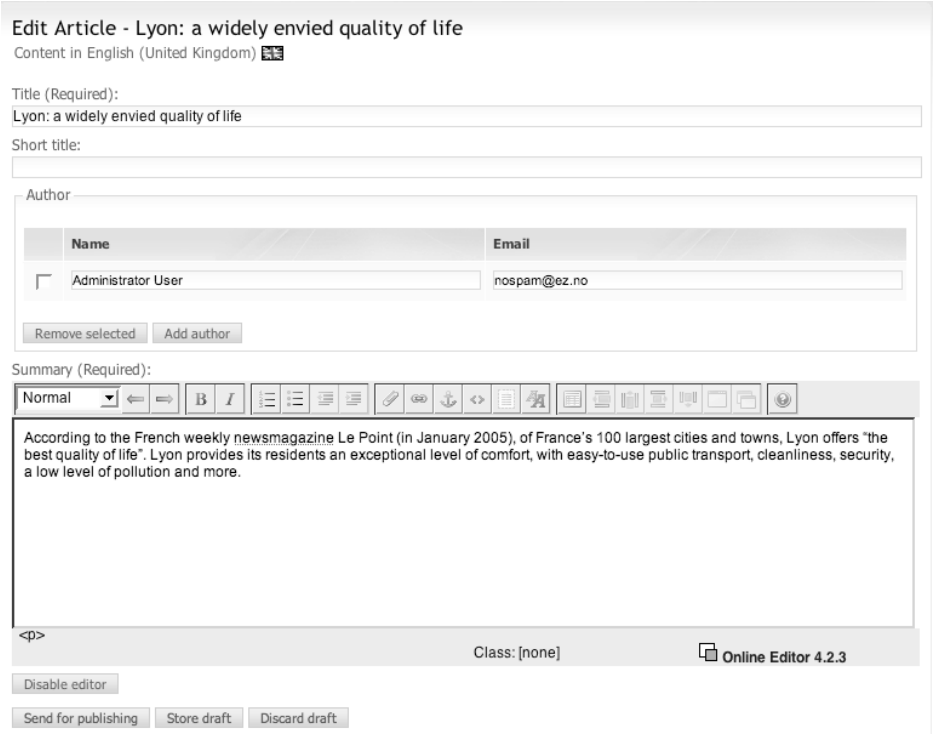
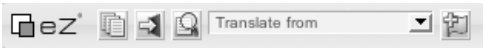


Figure 2.4. Edit mode for an article

The two most important tools for an editor in the Website Interface are the **Website Toolbar** and the Online Editor (as described previously). In edit mode, the **Website Toolbar** looks as follows:



The items on the toolbar provide access to the functions listed in the table below. When you are finished editing, choose either to publish the content (by clicking the **Send for publishing** button), to save the updated version as a draft without publishing it and without exiting edit mode (by clicking the **Store draft** button), or to save the updated version as a draft and exit edit mode (by clicking the **Store and exit** button on the **Website Toolbar**). Alternatively, you can click the **Discard draft** button to exit edit mode without storing your changes.

Table 2.2. Edit mode Website Toolbar functions

Function	Description
Manage versions	Used to view and revert to previous versions of an object.
Store and exit	Used to store a draft of the object and exit edit mode.
Preview	Used to view the modified object as it would appear to site visitors.
Language selection	Used to specify a language for a translation. Applies only to multilingual sites.
Translate	Used to translate an object. Edit mode will be put in a special *translator mode* where the text of each attribute in the source language is shown above the input fields. (Applies only to multilingual sites.)

2.2.4. Settings accessible through the Website Interface

In general, there are two kinds of settings that can be edited through the Website Interface: site settings and user profile settings. *Site settings* are only available to Administrator users, and can be accessed via the **Site settings** link in the top right corner. From there, you can edit information such as the company logo, the title of the site and more. For more information, see the *Website Interface User Guide* at http://ez.no/doc/extensions/website_interface.

The *user profile* is associated with each individual user account and is only accessible when the user is logged in. Here, you can change your password, update personal information, manage drafts and notifications and view Webshop order information (if your site has a Webshop). See Section 5.3.2.1, “Accessing your personal space”.

2.3. Administration Interface

The Administration Interface is the advanced back-end interface, providing powerful tools for content management and editing as well as site management, configuration, customization and development. It is always included in an eZ Publish installation, in contrast to the Website Interface, which is an optional extension. In other words, you can change almost everything about a site in the Administration Interface, although some of these functions are only relevant to site administrators and developers. Most of the tasks and procedures that are described in this book are performed in the Administration Interface.

The visual presentation of content is stripped down, with all design elements, layout and styling related to the public siteaccess removed. However, the Administration Interface has its own layout, restricted to structural organization of content, with navigation menus, access to management operations and special-purpose embedded interfaces (to manage content versions or create new user accounts, for example).

2.3.1. Layout of the Administration Interface

The screenshot below shows the Administration Interface. If you compare this screenshot to Figure 2.3, "Website Interface" you will see that the left menu and **Sub items** windows in the Administration Interface contain the same elements as shown in the horizontal menu of the Website Interface. Also, notice that there are many more tools and operations available with the Administration Interface.

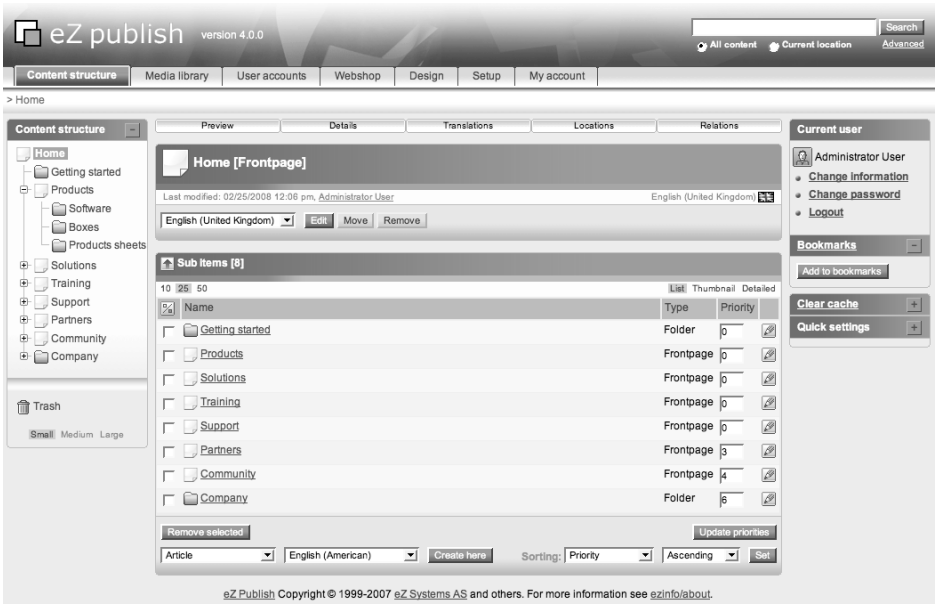


Figure 2.5. Administration Interface

The *main menu* is a collection of *tabs* located below the eZ Publish logo and the **Search** interface. Each tab represents a group of administration functions, such as user or Webshop management.

The *secondary menu* (also called the "left menu") provides access to content and interfaces that are associated with the tab that was selected from the main menu. If your browser has JavaScript support enabled, the secondary menu of the first three tabs is shown as an interactive tree.

The *main area* displays the actual content and interfaces that are associated with the selected object or menu item. This is where most of the work is done. When viewing content here, you will see a row of switches at the top that are used for toggling windows on and off.

The *path* is a representation of your current position in the navigation hierarchy of the site, or the functionality provided by the module and view (for example, "Settings / View" when accessing the "view" view of the "settings" module, or "Cache admin" when accessing the "cache" view of the "setup" module). For the three left-most tabs, the last path element shows the name of the content you are currently viewing, prefixed by links to its superior node(s).

The *right area* displays information specific to the user who is currently logged in. Among other things, this is where you find the **Bookmarks** window, which contains a list of the current user's bookmarks. These are internal eZ Publish bookmarks that link to different nodes in the tree. Do not confuse this with web browser bookmarks. Below the **Book-**

marks window is the **Clear cache** panel, which will be discussed in Chapter 3. The **Quick settings** window has some developer tools that are beyond the scope of this book.

Searching for content is done by entering text in the *Search interface* in the top right. In eZ Publish, default search behavior is case insensitive and matches exact words or phrases (see Chapter 11, *Search engines and finding content* for more information).

2.3.2. Navigating the Administration Interface

There are various ways to navigate the Administration Interface. The two most common methods are clicking around and searching. You can also navigate to a specific node by clicking on your pre-configured bookmarks. The three left-most tabs enable you to navigate the Content, Media and Users branches of the content node tree, while the menus of the other tabs provide access to functionality and management operations. Here, we describe how to navigate content, and the next chapter (Chapter 3, *The Setup tab*) describes how the left menu of non-content tabs work.

When you are looking for a specific piece of content (typically because you want to do something with it, like edit or move it), you can navigate the tree and examine the nodes.

The most convenient method for navigating the tree is to use the left menu. This menu is used to explore the node tree by opening different branches just as you would do when navigating a local file system.

The + and - buttons to the left of each node (not to be confused with the similar buttons used to hide or display the left menu itself) are used to unfold and fold the branches of the tree at any level. Collapsing can also be done from the context-sensitive pop-up menu. Note that if JavaScript is not enabled in your browser, the tree in the left menu is completely unfolded at all times.

The tree can also be navigated using the path and the **Sub items** window. In both cases, you click the text links that correspond to the node to which you wish to navigate. Note that this takes much more time than using the tree menu because every click forces the page to reload, while folding and unfolding the tree menu does not.

By default, only a subset of the node tree is included in the left menu, excluding display of all but the main container nodes. In other words, it displays the main structure but not all individual pieces of content. This restriction is applied in order to prevent a noticeable drop in performance when having to list all of the nodes on large sites. Thus, you will sometimes be forced to use the **Sub items** window together with the left menu in order to navigate to your desired location.

2.3.3. Administration Interface tabs

The following table gives a brief overview of the tabs of the Administration Interface.

Table 2.3. Administration Interface main menu tabs

Menu item tab	Description
Content structure	The **Content structure** tab displays the Content branch of the node tree. This is a tree-structure representation of most of the content that site visitors see.
Media library	The **Media library** tab displays the Media branch. The Media branch stores data that is frequently used by other nodes. It is typically used to store images, animations and documents that are related to nodes located in the Content branch.
User accounts	The **User accounts** tab displays the Users branch. This branch stores users and user groups in a structured manner. The **User accounts** tab is explained in Chapter 5.
Webshop	The **Webshop** tab is used to view and modify information that is related to the built-in e-commerce engine (the Webshop). It is explained in Chapter 15.
Design	The **Design** tab is used to view and modify the visual elements and layout of the site.
Setup	The **Setup** tab displays the main site configuration area and is explained in Chapter 3.
My account	The **My account** tab displays the current user's personal area. This area provides access to user-specific data and configuration, such as access to the user's own drafts and user account information.

2.3.4. Selectively displaying windows by toggling switches

The horizontally-aligned switches at the top of the main area determine which windows (also called "panels") are shown in the main area. A blue background (dark and shaded in the screenshot) indicates that a switch is on and thus the window containing the related information is currently being displayed. Due to print size limitations, Figure 2.6, “Horizontally-aligned switches - all except Preview enabled” is cropped to omit the **Sub items** window below the **Relations** window, and the **Preview** switch is not enabled. All of the main area windows should be shown.

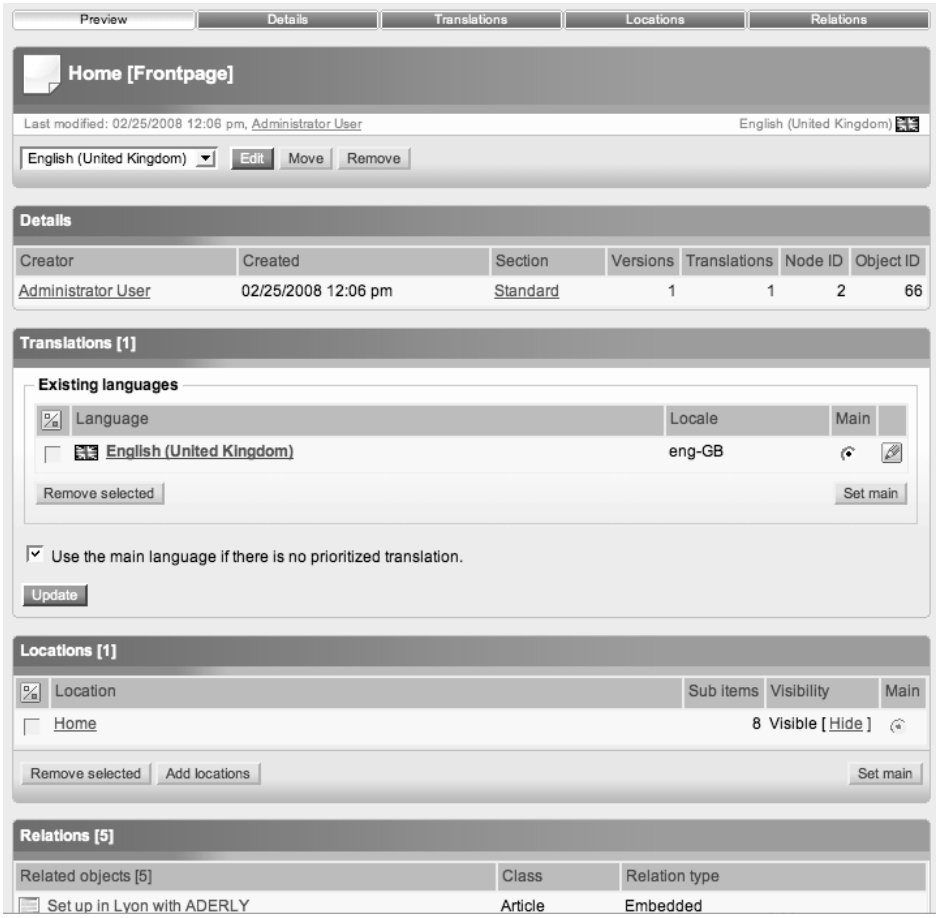


Figure 2.6. Horizontally-aligned switches - all except Preview enabled

The following table gives an overview of the windows in the **Content structure**, **Media library** and **User accounts** tabs. Note that the **User accounts** tab has two additional switches (**Roles** and **Policies**), which are described in Chapter 5.

Table 2.4. Overview of main area switches

Switch / window	Description
Preview	Displays the contents of the object that is referenced by the selected node. If the **Preview** switch is disabled, the corresponding window does not disappear completely (as it does for the other switches), but the window's contents are hidden. When enabled, it shows the attributes of the object, such as the title and body of an article.

Switch / window	Description
Details	Displays additional information about the selected node and the object that it encapsulates.
Translations	Displays the existing languages for the published version of the object that is being viewed. The currently selected translation is displayed in bold characters. This is not shown for single-language sites (see the "Single-language site" entry in the Glossary).
Locations	Displays the nodes that are associated with the object that is being viewed. The currently selected node is displayed using bold characters.
Relations	Displays information about objects that are either used by or that make use of the current object. In eZ Publish, any object can be used by any other object (see the "Object relation" entry in the Glossary). This feature is typically useful for relating or reusing information that is scattered around in the system.

The **Sub items** window is always visible at the bottom of the main area when viewing content in the three left-most tabs. It displays the node's children, meaning the nodes beneath the current node in the node tree. You can sort, remove and edit existing objects in this area. Initiate edit mode for a new object (opening the **Object Edit Interface** to create a new object) by using the **Create here** interface at the bottom left of the window.

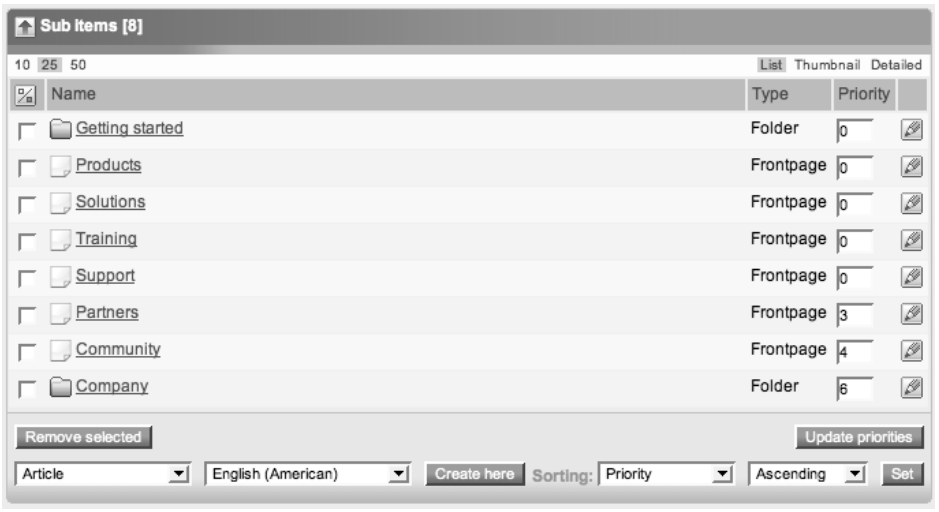


Figure 2.7. Sub items window

2.3.5. Object Edit Interface

The **Object Edit Interface** is a special-purpose interface embedded within the Administration Interface in which the editing of content objects takes place. By default, the **Object Edit Interface** makes use of the Online Editor. The **Object Edit Interface** also provides additional information to assist you when editing content, and provides access to features such as version management and previewing content as it would display on the front-end of a site.

The following screenshot shows how the **Object Edit Interface** is typically displayed. It usually consists of six windows:

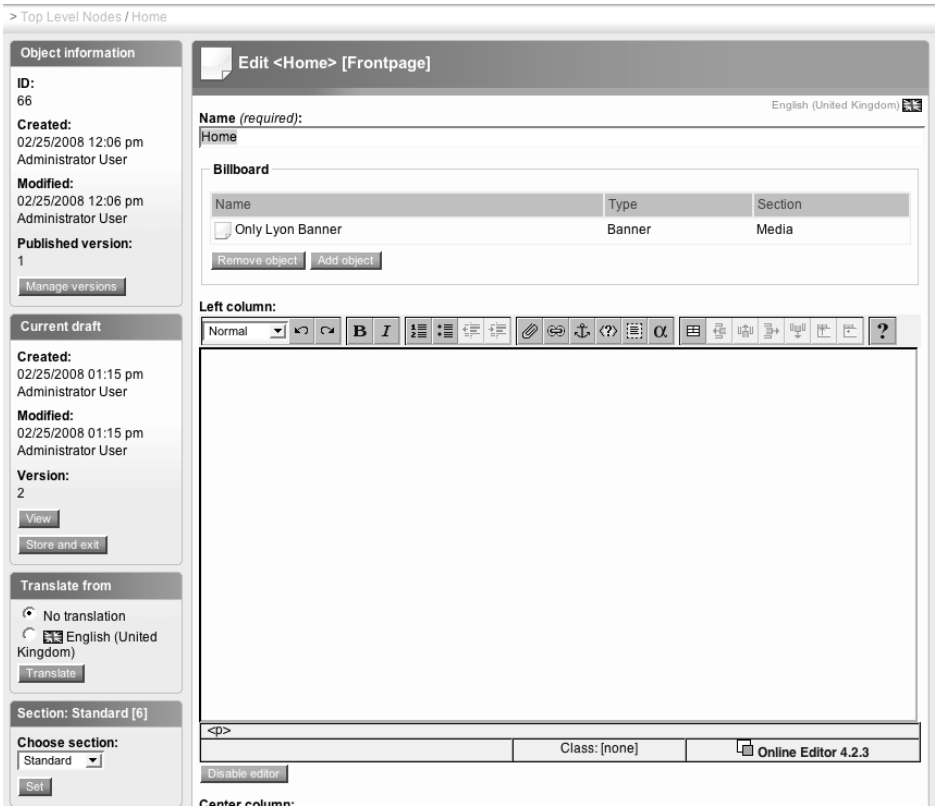


Figure 2.8. Object Edit Interface - Frontpage object

1. The **Object information** window displays information about the object that is being edited. This includes timestamps for when and by whom the content was created and modified. You can access the **Version history** interface here by clicking the **Manage versions** button.

2. The **Current draft** window displays information similar to that of the **Object information** window, but about the version - not the object - that is currently being edited. You can access the **Version preview** interface here by clicking the **View** button.

3. The **Translate from** window displays information about the existing languages. You can also disable translator mode or select the language on which the current translation will be based.

4. The **Main edit** window enables you to modify the contents of the attributes of the selected object and to enter content when creating new objects. For example, if an article object is being edited, this window enables you to change the title of the article, the summary and the body. The attributes will be displayed in the same order as they were set up when the object's class (which defines the actual data structure) was created. At the bottom are the **Send for publishing**, **Store draft** and **Discard draft** buttons.

   For multilingual sites, the **Main edit** window can be put into *translator mode* when adding a new translation or editing an object that already has multiple translations. In translator mode, you are shown the text of each attribute in the source language above the input fields where you enter the translated text in your target language.

5. The **Related objects** window (not shown in the screenshot above) makes it possible to relate other objects to the one that is being edited.

6. The **Section** window displays information about the section that the current object belongs to and enables you to change this. Sections are described in Chapter 4

2.3.5.1. Exiting the Object Edit Interface

You should always exit the **Object Edit Interface** by clicking the **Send for publishing**, **Discard draft** or **Store and exit** button. Note that clicking the **Store draft** button will not end the editing session. Clicking the Back button in your web browser is a very common mistake, responsible for many stale drafts that can possibly lead to editing conflicts at a later time.

Figure 2.9. Object Edit Interface button bar

2.4. Context-sensitive pop-up menu

The *context-sensitive pop-up menu* (or "pop-up menu" for short) contains functions that are specific to the item for which the menu was invoked. There are many ways to invoke the pop-up menu, including right-clicking anywhere in the **OE text area**. For most of the Administration Interface features explained in subsequent chapters in this book, the pop-up menu is activated by clicking the icon (with the left mouse button) to the left of

the name of the node either in the **Sub items** window or the left menu of the Administration Interface, as shown below:

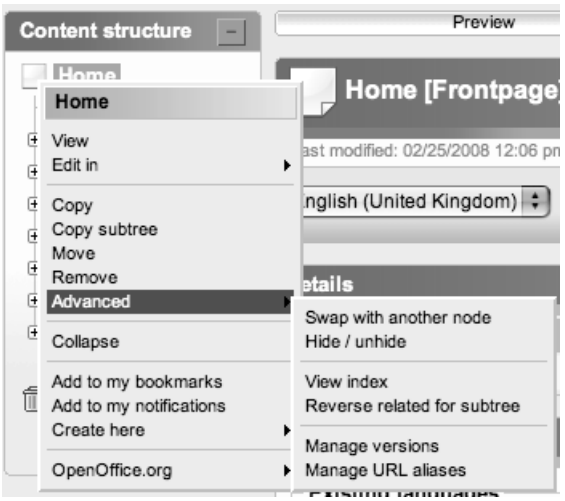


Figure 2.10. Context-sensitive pop-up menu

The title at the top of the pop-up menu is the name of the node that was clicked (in this case "Home"). Many of the items on the pop-up menu deal with common content-related tasks. For example, clicking the "Add to my notifications" item adds the node to your list of personal notifications, as was explained in Section 1.4, “Notifications”.

The "Advanced" sub-menu includes access to the following advanced features:

- Swapping nodes (see Chapter 10)
- Hiding and unhiding nodes (see Chapter 6)
- Sitemaps with different starting points (see Section 10.1, “Viewing a sitemap”)
- Managing URL aliases (see Section 12.5.1, “Node aliases interface”)

The "OpenOffice.org" sub-menu is explained in Chapter 14.

When viewing some content object in the main area of the Administration Interface, left-clicking the class icon in the title bar of the **Preview** window opens a pop-up menu targeted at webmasters and developers:

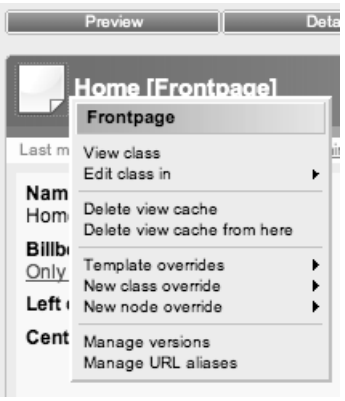


Figure 2.11. Pop-up menu for developers

This pop-up menu provides shortcuts for viewing and editing a class (see Section 3.2, “Viewing and editing content classes”), cache management (see Section 3.4, “Cache management”), overrides (see the *eZ Publish Basics* book), version management and URL management (see Section 3.3, “URL management”).

2.5. Frontpage management with eZ Flow

eZ Flow is an eZ Publish extension aimed at media sites that make rapid and frequent content changes. For example, it enables you to easily create a newspaper frontpage or a TV station portal. For system reference, the eZ Flow extension is called "ezflow". The following screenshot shows an eZ Flow frontpage.

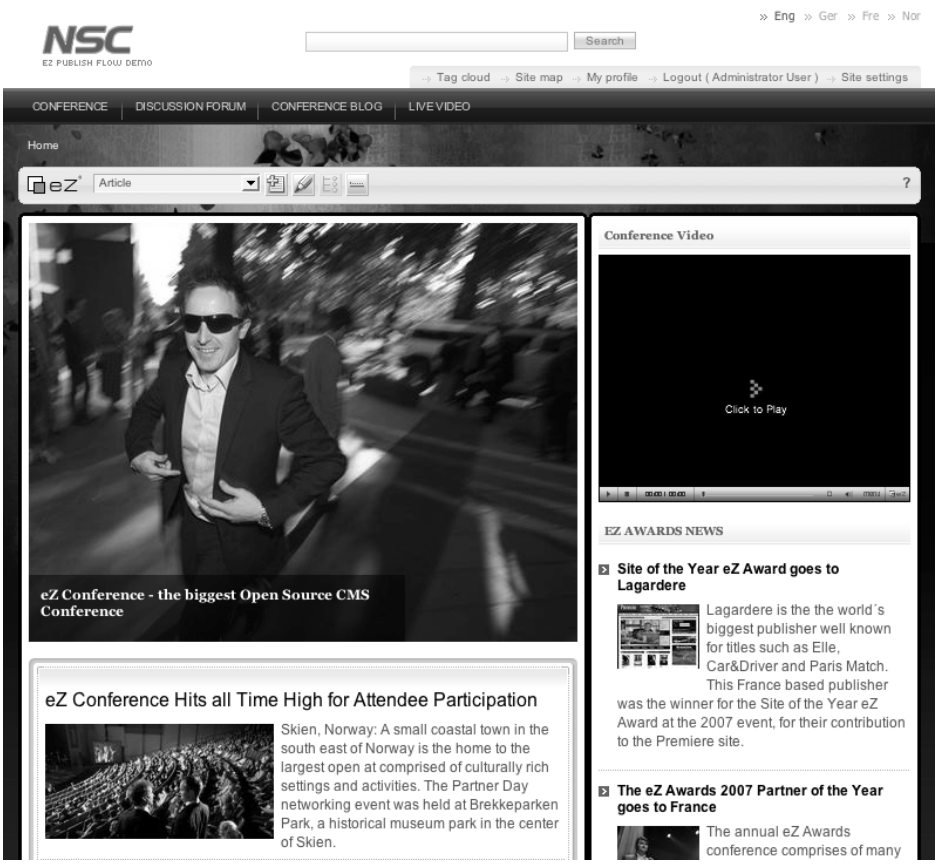


Figure 2.12. eZ Flow frontpage

The Website Interface is an integrated part of eZ Flow, as can be seen by comparing the above screenshot to Figure 2.3, “Website Interface”. In other words, with eZ Flow you also have access to the **Website Toolbar**, the Online Editor and the **Content Editing Interface**, as well as the Administration Interface.

The list below shows some of the features specific to eZ Flow. All of the standard and built-in eZ Publish features (such as instant web content editing and integration with standard word processing applications) are also available.

With eZ Flow, you can:

- build complex page layouts through the graphical user interface, using zones and blocks in frontpages;
- pre-plan the publication schedule and handle the flow of content publications over time; and

- rotate content and the placement of content on frontpages.

For information about technical configuration and customization, refer to the *eZ Flow Setup* manual at http://ez.no/doc/extensions/ez_flow.

2.5.1. eZ Flow concepts

Most of the eZ Flow concepts relate to the `Frontpage` class, which is described later. Recall from *eZ Publish Content Management Basics* that a frontpage displays content from many locations in the content node tree. In other words, it is not limited to displaying only its direct sub items, as folders are. Publishing on a frontpage usually means to embed content that is already published elsewhere.

2.5.1.1. eZ Flow zones

With eZ Flow, pages are configured into *zones*. Zones are the components of layouts, and each zone can contain one or more blocks of content. Each zone has a name, specified in the `zone.ini` configuration file (see Section 3.5, "The configuration model") and a placement, specified in a special-purpose template.

2.5.1.2. eZ Flow layout

A *layout* is a combination of zones placed on a page. The following screenshot shows the built-in layouts. Other layouts can be set up by your site administrator.

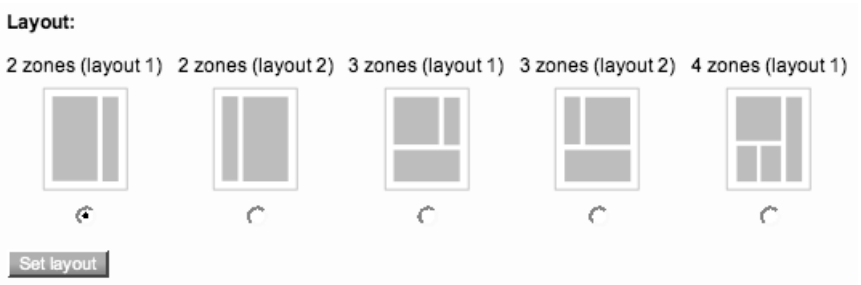


Figure 2.13. eZ Flow zone layouts

2.5.1.3. Blocks

A *block* is a section within a zone that contains a particular type of content. Each block has a type; a maximum number of displayed items (configured in the `block.ini` configuration file); and an editable name.

The block system includes two block categories - manual and special - that dictate how content is assigned. *Manual blocks* are assigned specific objects. *Special blocks* contain

content such as tag clouds (Section 9.1, “Content relevance with tag clouds”) or banner ads. The contents of blocks are displayed according to *block templates*.

2.5.1.4. Block lists

Recall from the previous chapter that the statuses of a content object (draft, published and archived) represent various stages in the object's life cycle. The items of a block go through similar stages: what is planned for publication; what is currently published; and what was previously displayed. These three stages correspond to the queue, online and history listings. We will examine these lists in detail related to publication planning in Section 7.9, “Content flow with eZ Flow”.

For manual blocks, publication within a block is not the same as publishing an object within the node tree. For example, multiple editors might publish articles in the same folder, but a frontpage manager would select from among these to decide what to publish for a specific block.

2.5.1.5. Overflow

The number of items a block can display at once is dictated in the `block.ini` configuration file, according to the block type. An *overflow rule* determines what happens when an item is no longer displayed within a block. This corresponds to the point in time when an item moves out of the online list in order for another item to move from the queue to the online list. For example, the former item can be moved to another block or be pushed to the currents block's "History" list. Overflow is covered more extensively in Section 7.9, “Content flow with eZ Flow”.

2.5.2. Frontpage class

The `Frontpage` class is not specific to eZ Flow; it is available in all eZ Publish sites. Frontpages in eZ Flow and in sites without eZ Flow look similar (for example, compare Figure 2.12, “eZ Flow frontpage” and Figure 2.3, “Website Interface”. Both frontpages have a "Name" and "Tag" attribute, and both are containers beneath which you can publish other content.

However, there is a significant difference in the class definition. Objects of the standard `Frontpage` class uses left, right and center columns in attributes of the "XML block" datatype. In eZ Flow, the *Layout* datatype stores information (name, flow, rotation) about the blocks it contains. It is used for structuring and positioning content and takes care of all time-based operations for content scheduling.

Another big difference is the use of more flexible display templates in eZ Flow, enabling you to change the layout and add and remove blocks when editing the object, instead of having to change the class definition.

In eZ Flow, objects of the `Frontpage` class are created as you would create any other content object: by choosing the desired location and clicking the **Create here** button. First you need to specify a name and select a layout. Then you can add blocks to the zones of the chosen layout (as described in Section 2.5.5, "Creating a new block") and fill them with content. The editing interface is explained next.

Tip

The `Global layout` class is a variation of the `Frontpage` class for making a reusable layout for general objects. It is limited to the Global zone layout, which is a single-zone layout. Refer to the eZ Flow documentation at http://ez.no/doc/extensions/ez_flow for more information.

2.5.3. eZ Flow interface

Content editors will notice that edit mode looks a bit different when editing frontpage objects with eZ Flow. The following screenshots show the editing interfaces through the front-end of a site and in the Administration Interface. These are special-purpose cases of the **Object Edit Interface** (seen in Figure 2.8, "Object Edit Interface - Frontpage object") and the **Content Editing Interface** (seen in Figure 2.4, "Edit mode for an article"). Edit mode for a Frontpage object in eZ Flow is called the **eZ Flow** interface.

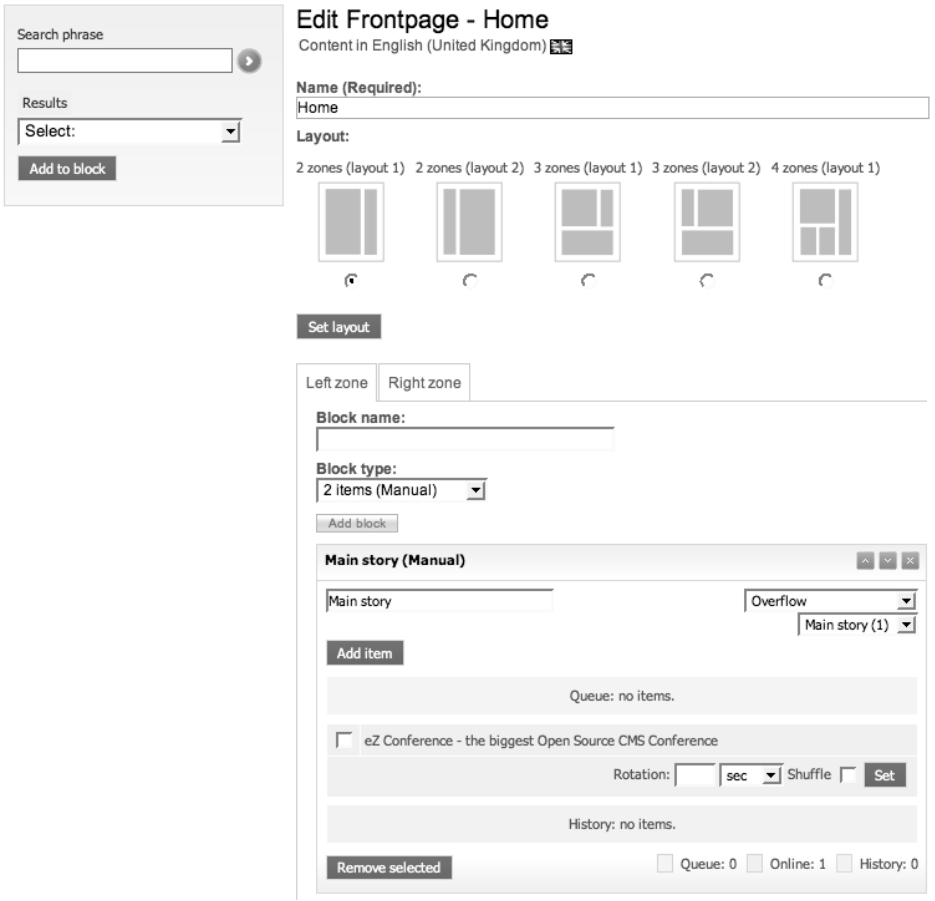


Figure 2.14. eZ Flow interface - front-end

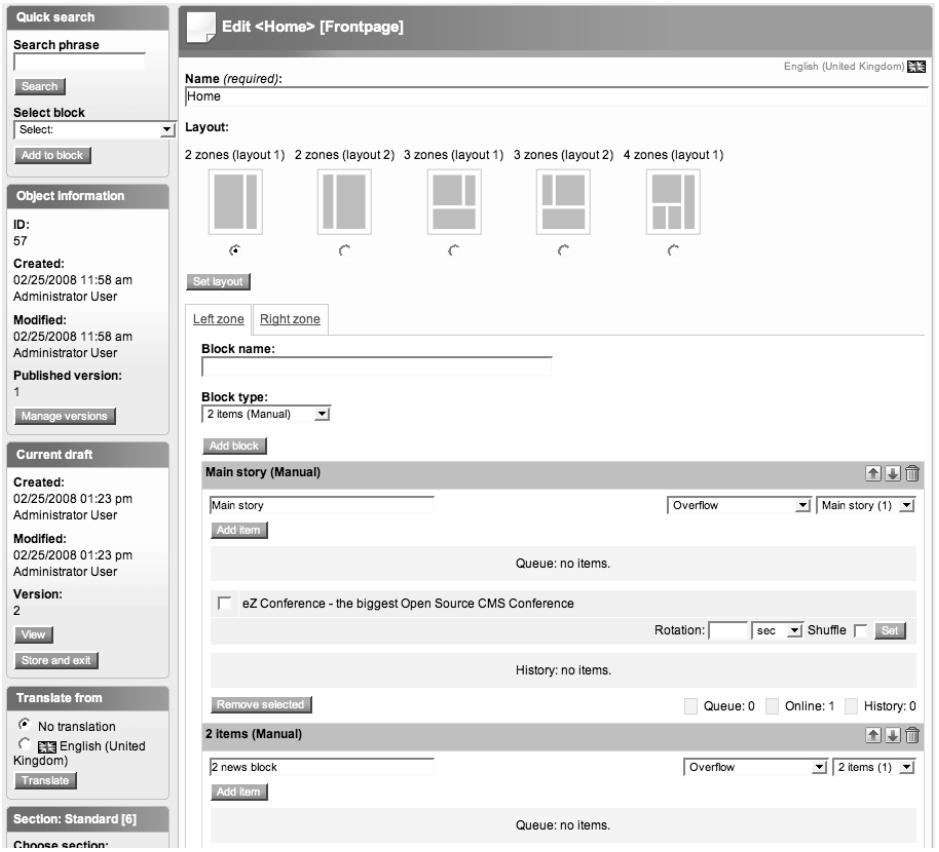


Figure 2.15. eZ Flow interface - back-end

As shown above, a new **Quick search** window has been added to the top left corner, providing easier access to find and add content to a block. For the **Object Edit Interface**, it is located above the **Object information** window (pushing the other windows down). Otherwise, all of the edit mode windows are present and in their regular positions. The **Quick search** window is described in Section 2.5.6, “Adding content to a block”.

We will base our explanations in this chapter on the **eZ Flow** interface in the Administration Interface. The positioning of elements such as input fields, labels and buttons, and also color schemes differ slightly between the front- and back-end interfaces, but the main elements and functionality are the same.

Each zone is represented by a tab in the **eZ Flow** interface. These can be seen directly below the **Set layout** button in the above screenshots. By default, these are named based on their location within a specific layout, such as "Left zone", "Right zone", "Bottom left zone" and "Bottom right zone". If you are looking at a newly created frontpage for which no layout has yet been specified, you do not have any zones yet, and consequently

no zone tabs either. Each tab shows the blocks within that particular zone. To change tabs, simply click on one of the other tabs.

Each zone's tab includes a simple interface for adding new blocks to the zone, followed by a listing of the existing blocks of that zone. If you have not created any blocks yet, the **Main edit** window looks similar to this:

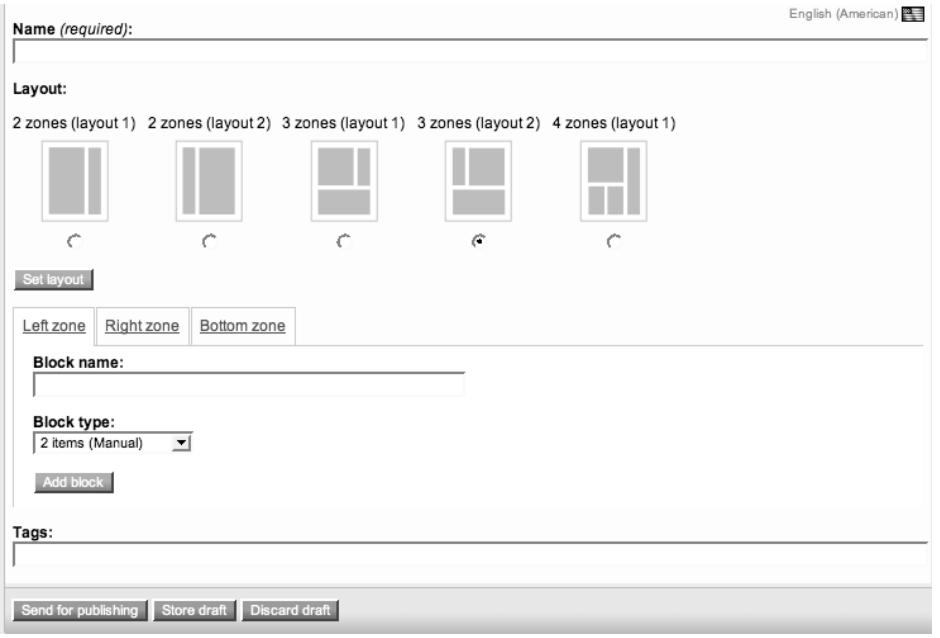


Figure 2.16. Zone without any blocks in edit mode

The screenshot below shows an example block in edit mode:

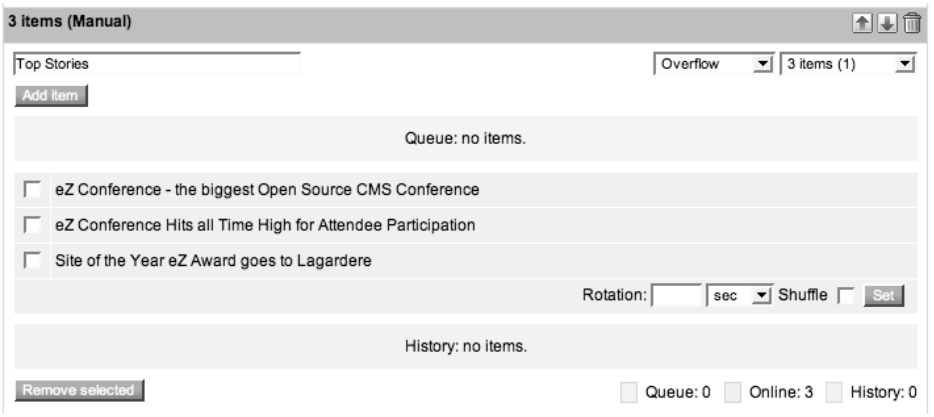


Figure 2.17. Example block in edit mode

The list below gives an overview of the various edit mode elements within a block.

- *Title bar*: reveals the block type ("3 items" in our example) and category ("Manual"), and includes arrow buttons to move the block's position up and down, relative to other blocks within the zone. It also has a **Delete** button beside the arrow buttons for removing the block.

- *Block configuration area*: located below the title bar, and includes an editable block name, as well as dropdown lists for setting the overflow rule and, in some cases, for customizing the display style for the block type.

- *Add item button*: located below the block name for manual blocks. Used for adding content to the "Queue" list of a block, as described in Section 2.5.6, “Adding content to a block”.

  Choose source button: shown instead of the **Add item** button for special blocks. Used for specifying the source location from which content should be automatically pulled.

- *Listings*: "Queue", "Online" and "History". You can re-order the items in the "Queue" list only by dragging and dropping an item to a new position in the list. The maximum number of items in the "Online" list is specific for each block type. One list item is shown per line. At the bottom of the "Online" list is the interface to control the automatic rotation of content. See Section 7.9, “Content flow with eZ Flow” for more information. The bottom right corner displays how many items each list currently contains.

- *Remove selected button*: located at the bottom left corner. Used in combination with checkboxes to remove items from the block's listings.

The following sections explain how to set the layout type, create a block, and add content to it.

2.5.4. Setting the layout type

The layout type of a frontpage is set in the **eZ Flow** interface (when editing the object) by marking the radio button below the desired layout, clicking the **Set layout** button and clicking the **OK** button in the confirmation dialog window. The top of the main area will then show that the draft was successfully stored. If this is a new version of an already published frontpage, you need to click the **Send for publishing** button for the chosen layout to take effect. The old version will still be available in the **Version history** interface.

Warning

Changing the layout type after you have created blocks will reset the frontpage to have empty zones. In other words, the existing structure will be removed. Doing this on a live site will result in a blank frontpage if you publish it immediately. If you are sure that you want to change the layout type, you should recreate the block structure and fill it with some content before you click the **Send for publishing** button.

2.5.5. Creating a new block

Initially, a zone is empty and you have to add blocks to it in order to fill it with content. This is also the case if you have just clicked the **Set layout** button. The following procedure shows how to add a block. Make sure that you are looking at the **eZ Flow** interface. In other words, you should first be in edit mode for the frontpage.

1. Select the zone to which you want to add a block by clicking one of the zone tabs. If you do not have any zone tabs, first set a layout type as previously described.

2. Enter a name for the new block in the **Block name** input field. The name will be shown as the heading for the block when the frontpage is displayed. You can change this later if needed.

3. Select a block type from the **Block type** dropdown list.

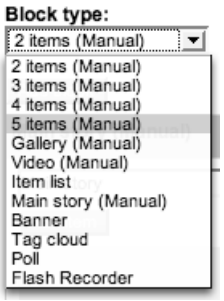


Figure 2.18. Block type dropdown list

4. Click the **Add block** button. This will add a new block of the specified type below the existing blocks. Use the arrow buttons on the title bar to move it up or down in relation to the other blocks in the zone.

5. In the block configuration area, you can optionally set an overflow rule. If the block type has multiple display styles, you can select this from the right-most dropdown list.

6. Add content as described in the next section.

7. Repeat the above steps for additional blocks.

8. When done, click the **Send for publishing** button to make your changes live.

Tip

You can remove a block by clicking the **Delete** button on the right side of the block's title bar.

2.5.6. Adding content to a block

There are multiple ways to add content to a block. For special blocks, all you have to do is to specify a source. Clicking the **Choose source** button opens the **Browse** interface where you can mark the desired object(s), then click the **Select** button. The Node ID of the source will then be shown below the **Display type** dropdown list.

For manual blocks, you have two options. You can either use the **Add item** button within the block, or the **Quick search** window in the top left. Both options are covered by the following procedures. We assume that you are already editing the frontpage.

Tip

To remove some content from a block, simply mark the corresponding checkbox and click the **Remove selected** button in the bottom left of the block.

2.5.6.1. Populating a block using the Add item button

The following procedure shows how to add content to a block by using the **Add item** button.

1. Click the **Add item** button below the **Block name** input field.

2. This will open the **Browse** interface. Select one or more items by marking the corresponding checkboxes, then click the **Select** button. To select multiple objects with different parent nodes, you have to repeat this procedure.

   The selected items will be added to the queue:

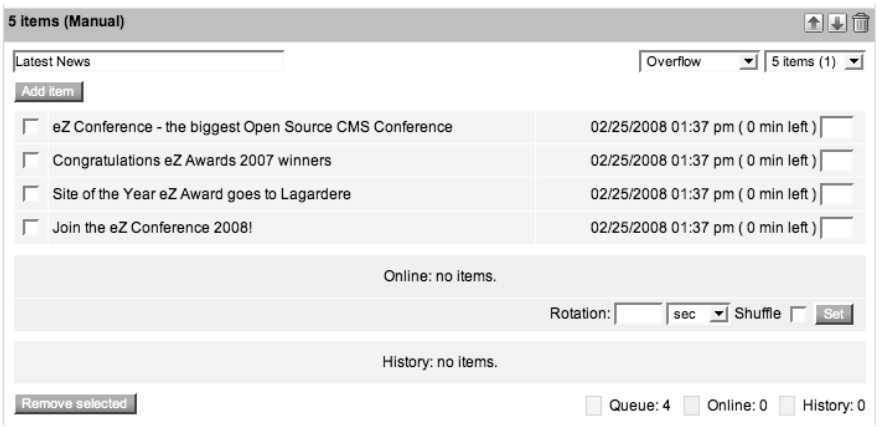


Figure 2.19. Queue with selected items

3. When you are finished adding items, click the **Send for publishing** button. By default, new items are scheduled for immediate publication and will be moved to the "Online" list, as shown in the following screenshot. You will learn more about scheduling in Chapter 7.

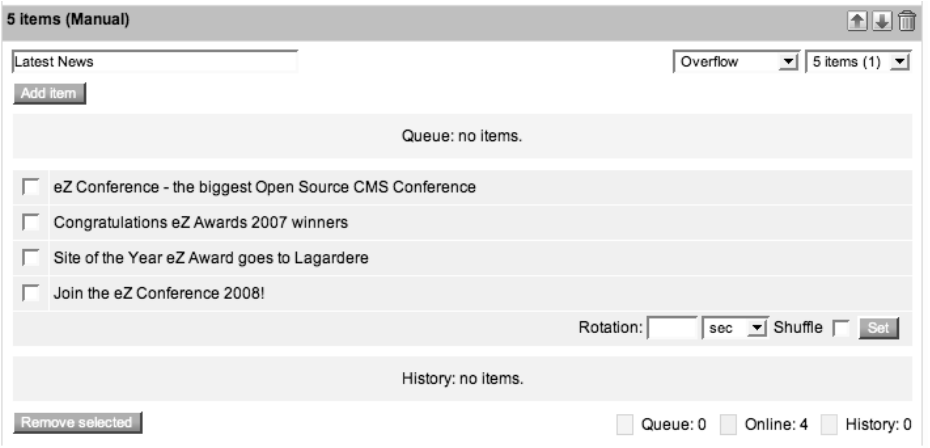


Figure 2.20. Newly added content is published

2.5.6.2. Populating a block using the Quick search window

The following procedure explains how to add content to a block by using the **Quick search** window.

Figure 2.21. Quick search window

1. Enter a search word or phrase in the corresponding field in the **Quick search** window, then click the **Search** button. (This is displayed as an arrow on the front-end of the site).

2. The search results will be shown in the **Quick search** window below the **Search** button:

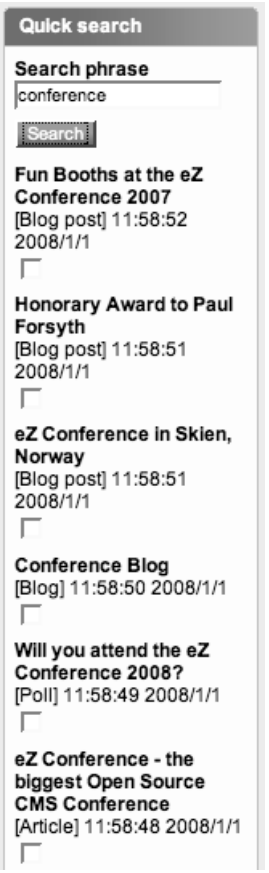


Figure 2.22. Quick search window - results

3. Pick the desired content from the search results by marking the corresponding checkbox(es).

4. Select a destination block from the **Select block** dropdown list at the bottom of the **Quick search** window, then click the **Add to block** button.

   The selected items will be added to the queue. Repeat the procedure as needed. When you are finished adding items, click the **Send for publishing** button.

2.6. Summary

Below we have compiled a list of the most important concepts in this chapter. We encourage you to review this list and refer to the Glossary for a full listing and definitions of terms in this book.

- Online Editor (OE): a WYSIWYG, embedded rich text editor for creating formatted pages in edit mode using an intuitive interface. It includes the **OE text area**, **OE status bar**, **OE toolbar**, modal dialogs and context-sensitive pop-up menu.

- Website Interface: Front-end interface available through the "ezwebin" extension. It is an enhancement to public siteaccesses, providing additional functionality to editors and administrators when logged in. In other words, it is what site visitors see, plus more. It includes the **Website Toolbar** for easy access to content management operations related to the content that is being viewed, and the **Content Editing Interface** for edit operations.

- Administration Interface: Back-end interface for providing powerful tools for content management and editing as well as site management, configuration, customization and development. It includes the **Object Edit Interface** for editing operations.

- Context-sensitive pop-up menu: provides quick access to functions that are specific to the item for which the menu was invoked.

- eZ Flow: an eZ Publish extension aimed at media sites with rapid and frequent content changes. A *layout* is a combination of *zones* placed on a frontpage. A *block* is an area within a zone that displays the content. The **eZ Flow** interface is a special-purpose edit mode for managing frontpages.

Chapter 3. The Setup tab

This chapter focuses on the **Setup** tab of the Administration Interface. It gives a comprehensive description of the tab's layout and contents, providing advanced content managers and webmasters with the understanding and interface knowledge needed to carry out their tasks. Selected features are covered in more depth, including eZ Publish URLs, cache management, the configuration model, workflows and extensions.

This material applies to content managers and webmasters working with the Administration Interface and is a prerequisite for the following chapters. In addition, it is relevant to those who seek to expand their knowledge of eZ Publish concepts.

Recall that:

- The Administration Interface tabs make up the main menu, and are horizontally aligned below the eZ Publish logo and the **Search** interface.
- The **Setup** tab represents the main site configuration area.
- A site interface refers to the visual presentation, and is recognized through the address in the browser. There is a one-to-one relationship between a site interface and a *siteaccess*. A siteaccess is a collection of (override) configuration settings. The Administration Interface corresponds to the *admin* siteaccess, and the public front-end interface (possibly containing the Website Interface) corresponds to the *public* siteaccess.

In this chapter, you will find information about the following:

- The layout of the **Setup** tab
- What cache is and how it is managed
- How to interpret eZ Publish URLs and how you can specify your own URL aliases
- Link management
- Extension management
- The configuration model and INI files
- The components of the workflow system, including workflows, workflow groups, triggers and events

This chapter and book focuses on the parts of the **Setup** tab that are relevant to advanced content managers or webmasters. Interested readers should refer to http://ez.no/doc for more technically detailed information.

The first section covers the **Setup** tab in general. The subsequent sections cover specific features that are managed and accessed through this tab; these sections are organized by giving the conceptual material first, then explaining interface and usage. Readers only interested in the concepts can thus skip the second half of a section and move on to the next concept. The chapter closes with a summary that can also be used as a quick introduction to the concepts introduced here.

3.1. Setup tab - layout and contents

This section provides screenshots and an illustrated walk-through of the layout and contents of the **Setup** tab. The **Setup** tab is associated with the Setup branch of the node tree, similar to how the **Content structure** tab is associated with the Content branch.

Access to the **Setup** tab is restricted (see Chapter 5) and should only be given to advanced users (such as experienced content managers and webmasters) and site administrators. For some user groups, the tab may be disabled. In order to perform many of the procedures in this book, you need to log in to the Administration Interface with an Administrator user account. Live sites, especially large ones, will typically have a user group and associated role for advanced content managers and webmasters.

Tip

You cannot limit a search to items within the **Setup** tab, because this area does not contain site content. Also, you cannot use the eZ Publish bookmarks feature on pages in this tab.

The screenshot below shows the general layout of the **Setup** tab, displaying the **Available extensions** window.

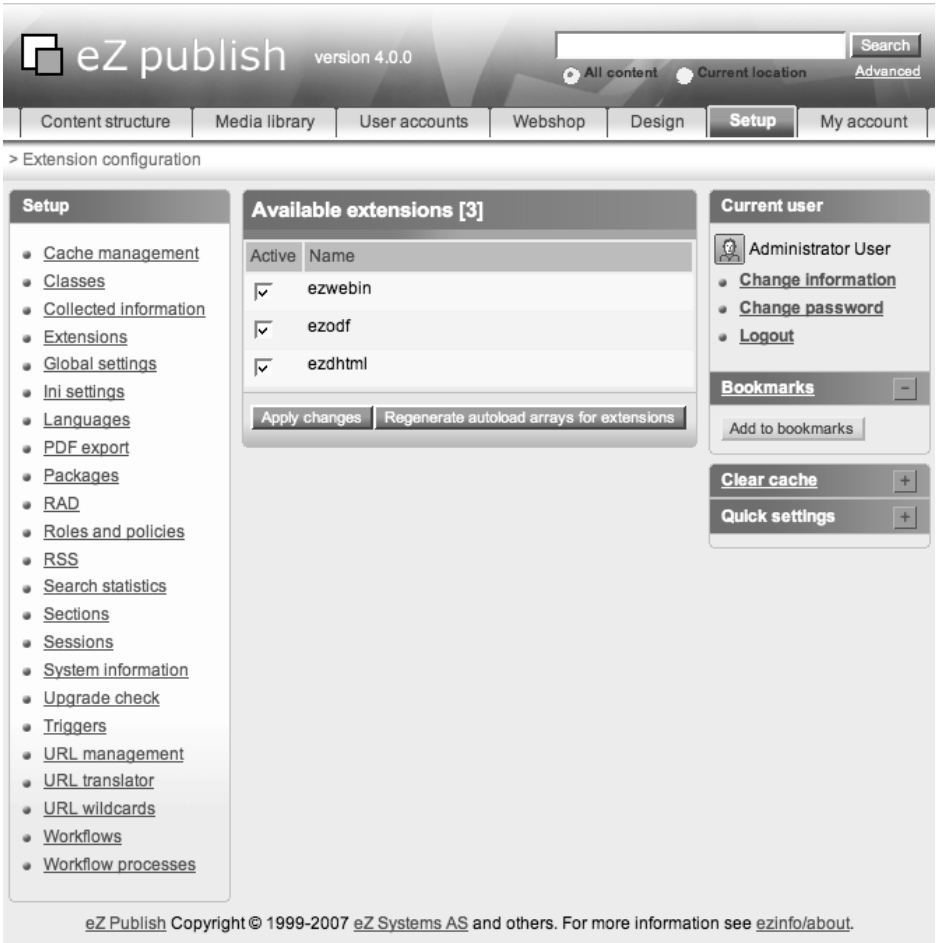


Figure 3.1. Setup tab layout

We encourage you to compare this to Figure 2.5, “Administration Interface” to explore the differences between the **Setup** tab and the three left-most tabs (the **Content structure**, **Media library** and **User accounts** tabs). First, observe that the six areas are present in their normal position. In other words, the **Search** interface, main menu and path are found horizontally at the top of the page, and the left menu, main area and right area are aligned side-by-side below these elements.

Second, observe the different appearance of the left menu. In the three left-most tabs, this menu represent a branch of the node tree. Therefore, it usually contains a hierarchical structure, where each entry is a content node with an icon representing its content type and the name of that node as a clickable link. As seen in the above screenshot, the left menu for the **Setup** tab is a simple one-level bulleted list, sorted alphabetically according to the English text of the links. In other words, if your Administration Interface is in a different language, these links might not be in alphabetical order.

Since the **Setup** tab does not deal with content, some more functionality has been omitted. The entire left menu cannot be minimized / opened using the + and - buttons in the menu header, in contrast the three left-most tabs. Also, the left menu in the **Setup** tab has no link to the trash and you cannot change the width of the menu.

In other words, the left menu of the **Setup** tab is fixed and links to categories exclusively for configuration, management and some development features, but not content. The one exception is when you click the **Collected information** link, which provides access to data obtained in feedback forms, polls, and so on; this data is stored in a dedicated part of the database, separated from but related to the object itself.

When clicked, most of the links will reload the page with some special-purpose windows in the main area. Some links will redirect you to other parts of the Administration Interface, such as the **Roles and policies** link, which sends you to the **User accounts** tab.

The following table describes the categories in the left menu of the **Setup** tab.

Table 3.1. Setup tab categories

Setup category	Description
Cache management	Provides access to both general and fine-grained cache control, as well as the generation of static cache. Cache is described below in Section 3.4, “Cache management”.
Classes	Provides access to view and manage individual content classes and class groups. Classes are described below in Section 3.2, “Viewing and editing content classes”. This is where you open the **Class edit** interface.
Collected information	Here you can access content stored in the system that has been obtained from users, such as the contents of a feedback form. See the "Information collection" entry in the Glossary.
Extensions	Available eZ Publish extensions can be enabled and disabled using this feature. This topic is explained in more detail in Section 3.6.2, “Managing extensions through the Administration Interface”.
Global settings	Deprecated. See INI settings below.
Ini settings	Used to view and edit configuration settings for any combination of INI file and siteaccess. See Section 3.5, “The configuration model” for more information.

Setup category	Description
Languages	Here, you can view and update the set of languages in which you can create and translate content on your site. This is explained in Chapter 12, *Working with multilingual sites*.
PDF export	Deprecated. Refer to http://ez.no/doc for more information on this topic.
Packages	Provides an interface to import, create and remove packages for your site. Refer to http://ez.no/doc for more information on this topic.
RAD	Provides access to *Rapid Application Development Tools* (RAD Tools). This applies only to developers.
Roles and policies	Redirects you to a permission management part of the **User accounts** tab, which is explained in Chapter 5, *User management and the permission system*.
RSS	Provides an interface to view, set up and maintain *Really Simple Syndication* (RSS) imports and exports. Refer to http://ez.no/doc for more information.
Search statistics	Lists statistics of the phrases that have been searched for on your site, the number of times each phrase has been entered, and the average number of results returned per phrase. The **Search statistics** interface is described in Section 11.9, “Search statistics”.
Sections	Provides access to section management. This is covered in Chapter 4.
Sessions	Shows which users are currently logged in, how many anonymous visitors are browsing your site, etc. Refer to http://ez.no/doc or the "Session" entry in the Glossary for more information.
System information	Provides a list of system information for your eZ Publish installation, including the site URL, the eZ Publish version, active extensions, and also hardware and software details for your web server and database. Knowing where to find this information can be useful if you need to get support on your site. This information is also important to system administrators and developers.

Setup category	Description
Upgrade check	Provides an easy way to check file consistency and / or database consistency prior to an upgrade. This helps site administrators to prepare the current installation for an upgrade.
Triggers	Used for associating workflows with triggers. Triggers initiate workflows. See below about workflows.
URL management	Provides an interface for inspecting and editing published URLs. See Section 3.3.3, “Link management”.
URL translator	Lets you view and manage the global URL alias list. See Section 3.3.2, “URL aliases”.
URL wildcards	Lets you set up URL forwarding with wildcards. See Section 3.3.2, “URL aliases”.
Workflows	Provides access to set up and manage sequences of actions, for example for editorial approval of content. The workflow system is described in Section 3.7, “The workflow system”.
Workflow processes	Applies to workflows; see above.

The main area contains special-purpose windows tailored to the functionality accessed. This area does not have a fixed layout and common windows as for the three left-most tabs. In addition, note that the horizontally-aligned switches shown at the top of the main area in Figure 2.5, “Administration Interface” are not part of the **Setup** tab.

As you will see throughout the chapter, the special-purpose windows are quite diverse in their layout. Explore the links to get an idea of the different interfaces and functions. The **Setup** tab is not necessarily technically complex, yet it provides a powerful tool to view and manage various aspects of your site. If in doubt, consult someone more experienced with the Administration Interface, such as your site administrator. Clicking the links in the left menu only reloads the page; it does not change anything on your system.

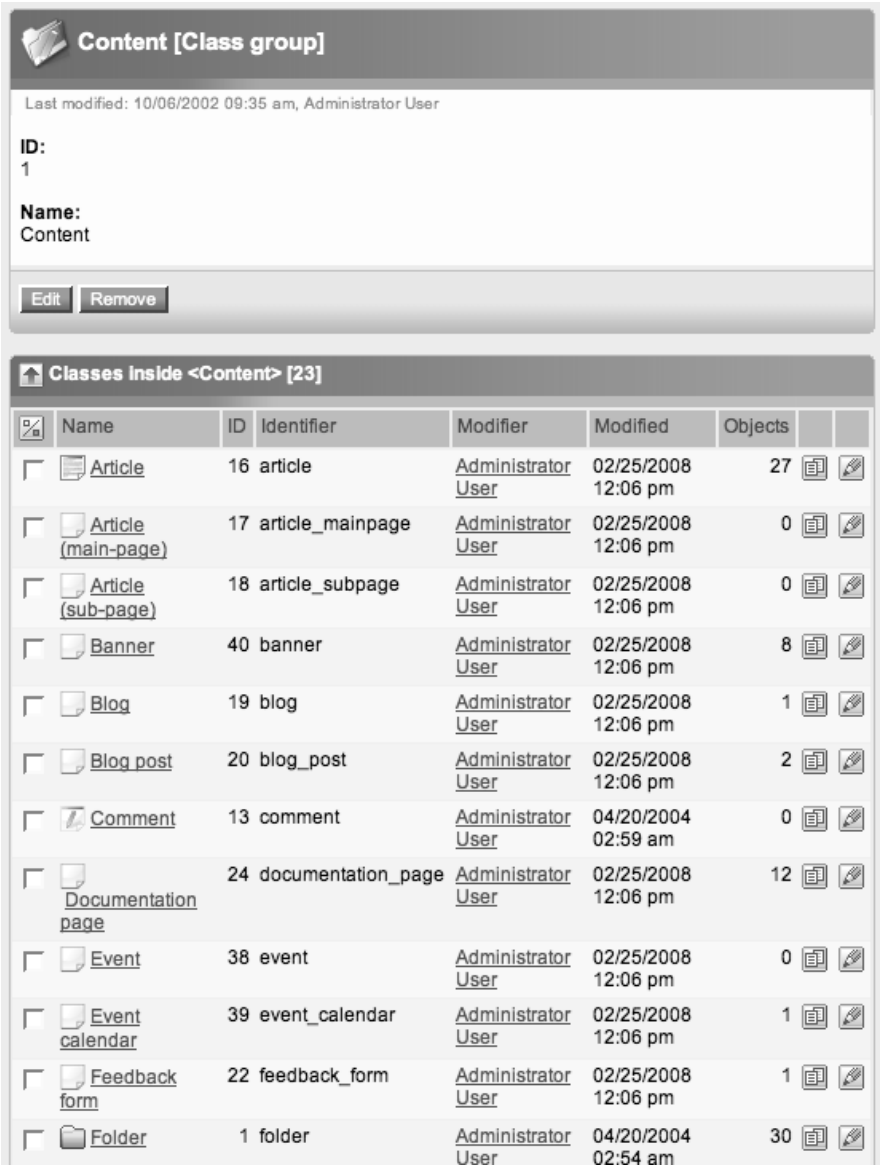
3.2. Viewing and editing content classes

Content managers sometimes need to check certain details about a class definition. For example, they might need to know whether an attribute is searchable, if it has a default value, and, if so, what that value is. The following procedure shows how to check this, and is useful to know throughout this book.

1. Access the **Class groups** window by clicking the **Classes** link in the left menu of the **Setup** tab.

2. Select one of the *class groups* by clicking on its name. By default, "Media" holds multimedia related classes, "Users" holds user management related classes, and "Content" holds almost everything else. If your site has custom classes, they might be located within a custom class group. This will take you to the *class list* for the selected group.

   You can alternatively (replacing steps 1 and 2) select "View class" from the pop-up menu accessed by clicking the icon on the title bar when viewing the content object in question (see Figure 2.11, “Pop-up menu for developers”).



Name	ID	Identifier	Modifier	Modified	Objects
Article	16	article	Administrator User	02/25/2008 12:06 pm	27
Article (main-page)	17	article_mainpage	Administrator User	02/25/2008 12:06 pm	0
Article (sub-page)	18	article_subpage	Administrator User	02/25/2008 12:06 pm	0
Banner	40	banner	Administrator User	02/25/2008 12:06 pm	8
Blog	19	blog	Administrator User	02/25/2008 12:06 pm	1
Blog post	20	blog_post	Administrator User	02/25/2008 12:06 pm	2
Comment	13	comment	Administrator User	04/20/2004 02:59 am	0
Documentation page	24	documentation_page	Administrator User	02/25/2008 12:06 pm	12
Event	38	event	Administrator User	02/25/2008 12:06 pm	0
Event calendar	39	event_calendar	Administrator User	02/25/2008 12:06 pm	1
Feedback form	22	feedback_form	Administrator User	02/25/2008 12:06 pm	1
Folder	1	folder	Administrator User	04/20/2004 02:54 am	30

Figure 3.2. Class list for Content class group

3. The classes are listed alphabetically. Select a class by clicking on its name. This will open the **Class view** interface:

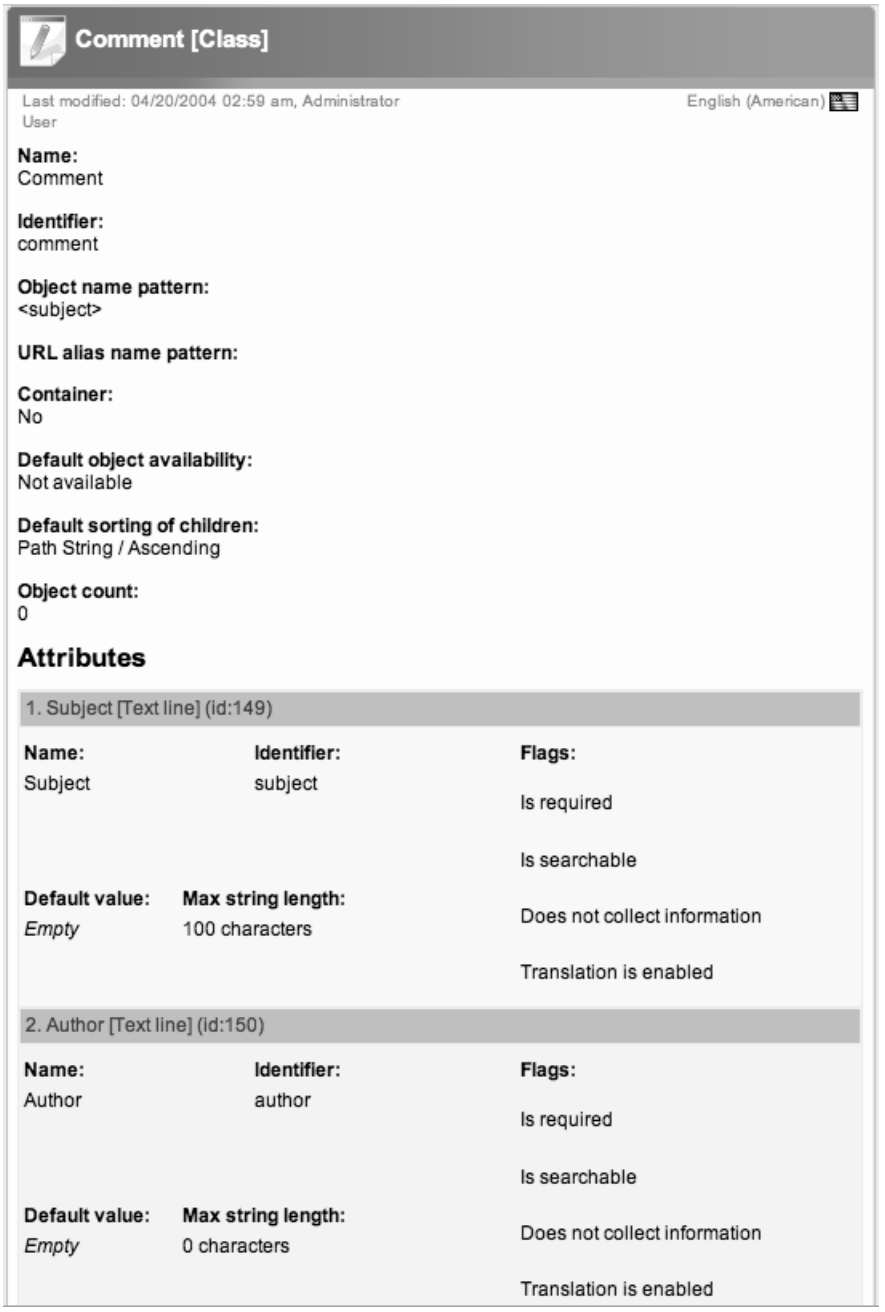


Figure 3.3. Class view interface - Comment class

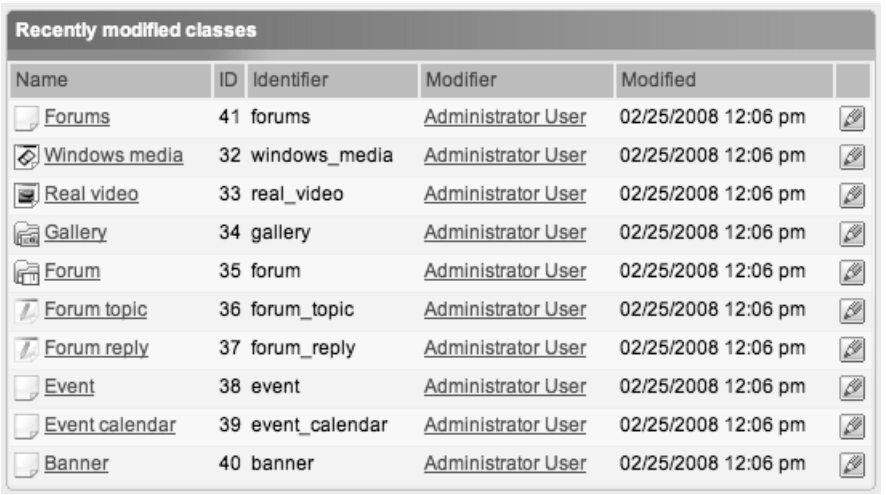
The display has some similarities to the **Preview** window used to view content objects. The title bar shows the class name and the class icon. Class properties like its

name and whether it is a container are listed above the attributes. For each attribute you can view its name, identifier and generic controls (or "flags"), in addition to datatype-specific properties. For example, you can view the maximum length of a text line, the default number of rows for an XML block, the default marking of a checkbox, and more.

4. When you are finished viewing the class, you can go back to the class list by clicking the browser's Back button. Or, you can navigate to any location within the Administration Interface. Alternatively, you can open the **Class edit** interface (explained below) by clicking the **Edit** button at the bottom of the page.

Tip

Below the **Class groups** window is a list of the most recently modified classes. If the class you wish to view or edit is on this list, you can access it directly by clicking its name or corresponding **Edit** button.



Recently modified classes

Name	ID	Identifier	Modifier	Modified	
Forums	41	forums	Administrator User	02/25/2008 12:06 pm	
Windows media	32	windows_media	Administrator User	02/25/2008 12:06 pm	
Real video	33	real_video	Administrator User	02/25/2008 12:06 pm	
Gallery	34	gallery	Administrator User	02/25/2008 12:06 pm	
Forum	35	forum	Administrator User	02/25/2008 12:06 pm	
Forum topic	36	forum_topic	Administrator User	02/25/2008 12:06 pm	
Forum reply	37	forum_reply	Administrator User	02/25/2008 12:06 pm	
Event	38	event	Administrator User	02/25/2008 12:06 pm	
Event calendar	39	event_calendar	Administrator User	02/25/2008 12:06 pm	
Banner	40	banner	Administrator User	02/25/2008 12:06 pm	

Figure 3.4. Recently modified classes

In most cases, it is sufficient just to check the class definition. However, sometimes you might want to change something, to make an attribute searchable, for example. In some rare situations, webmasters are also required to add additional attributes. The following procedure shows how to access the **Class edit** interface.

1. Click the **Classes** link in the left menu of the **Setup** tab.

2. Locate the class and click the **Edit** button. There are three alternative places to do this:

- Directly from the list of recently modified classes (see Figure 3.4, “Recently modified classes”).
- From the class list of a group (see Figure 3.2, “Class list for Content class group”).
- At the bottom of the **Class view** interface (see Figure 3.3, “Class view interface - Comment class”).

You can alternatively (replacing steps 1 and 2) select "Edit class" from the pop-up menu accessed by clicking the icon on the title bar when viewing the content object in question (see Figure 2.11, “Pop-up menu for developers”).

3. This will open the **Class edit** interface:

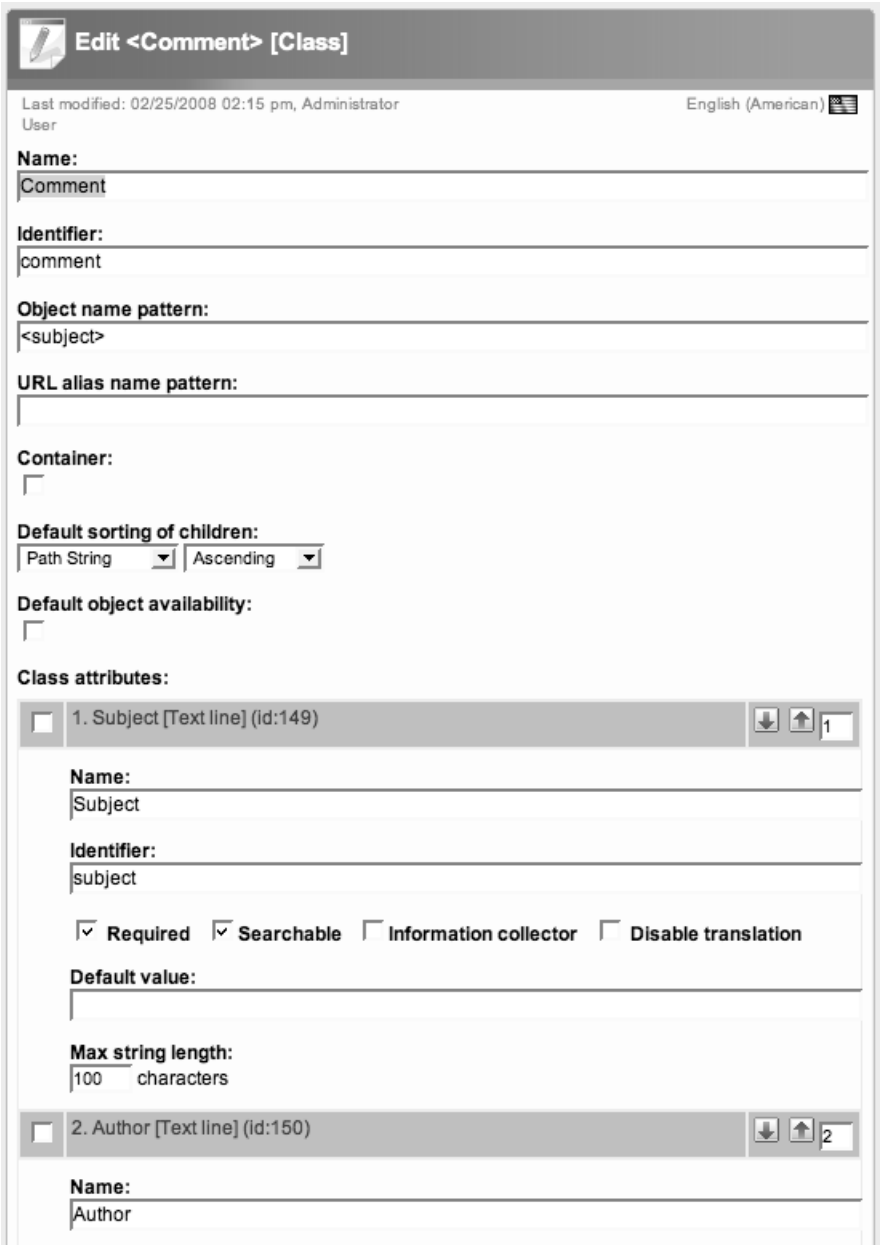


Figure 3.5. Class edit interface - Comment class

Refer to *eZ Publish Basics* or http://ez.no/doc/ez_publish/technical_manual for more information about editing classes.

4. To abort, click the **Cancel** button (rather than the browser's Back button). The **Apply** button works like a "save draft" operation, while the **OK** button saves the changes and exits edit mode.

3.3. URL management

This section explains eZ Publish URL concepts, and explores practical URL management tasks through the Administration Interface.

3.3.1. eZ Publish URLs

In order to access a webpage (or file) you have to tell your browser where to find it. A *Uniform Resource Locator* (URL), such as "http://ez.no", is the way to specify this. A URL is also commonly referred to as a "web address".

Before we examine eZ Publish specific URLs, consider the basic structure of a URL, as illustrated below:

Figure 3.6. Basic URL structure

eZ Publish URLs always consist of a protocol and a hostname. If your site is not hosted in the root directory, the directory to your eZ Publish installation will also be specified. As will be explained, what is called the "path" in the screenshot above is also referred to as index parameters.

In eZ Publish, two types of URLs co-exist. *System URLs* are the raw identifiers used internally and normally not displayed to site visitors when viewing content. *Virtual URLs* are more user friendly and represent simplified versions of system URLs.

An example of a system URL is "http://www.example.com/content/view/full/138", while its corresponding virtual URL is "http://www.example.com/company/about_company".

We explore the composition of these URL types in the subsections below. Readers not interested in this in-depth material may skip this and continue reading Section 3.3.2, “URL aliases” below. However, we strongly recommend that users who will work with URL aliases learn these concepts. Readers interested in how URLs are parsed internally in the system should refer to http://ez.no/doc/ez_publish/technical_manual.

3.3.1.1. System URLs

A system URL contains various information about either the content or functionality it identifies. At first glance, URLs of this type are not very user friendly, but they have a clear and well-defined structure that makes them easy to work with. "http://www.example.com/content/view/full/138" is a stripped and simplified system URL, only showing the hostname of the web server or website and the index parameters. Here, "www.example.com" denotes the hostname. The following components are index parameters, which, for system URLs, always include the module ("content" in our example) and view ("view" in our example). Do not confuse index parameters with view parameters. View parameters, such as "/full/138" in the example, are a subset of index parameters.

You might also encounter the expanded version of system URLs: "http://www.example.com/index.php/siteaccess/content/view/full/138". The additional elements between the hostname and the index parameters are "index.php", which is the name of the main index file; and, "siteaccess", which refers to the name of the siteaccess (for example, "eng").

Content-related system URLs often have the Node ID at the end. You can find the Node ID by examining the **Details** window, by looking at the URL in your browser when viewing the target content, or by hovering over the icon representing that content in the left menu or **Sub items** window.

Working with eZ Publish URLs, you will soon get familiar with this notation, and as an advanced content editor or webmaster you will usually only have to worry about the index parameters.

3.3.1.2. Virtual URLs

In contrast to system URLs, virtual URLs are more user friendly and easier to remember, and are what you normally see when browsing the front-end of a site. A virtual URL lets you use web addresses that identify content objects by their location rather than the module, view and Node ID. In the previous example, "http://www.example.com/company/about_company" refers to exactly the same webpage as "http://www.example.com/content/view/full/138". As is true in this case, virtual URLs can sometimes be shorter than the corresponding system URL.

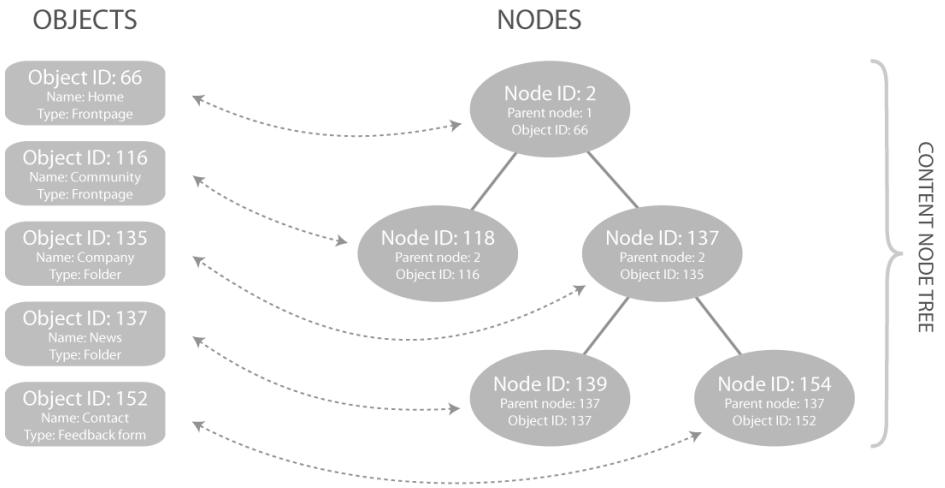
"content/view" in the system URL example reveals the module and view used; the number 138 is a view parameter that refers to the content node (its Node ID); and "full" is a view parameter specifying a template resource. "company/about_company" in the virtual URL represents the navigation path for the node's tree location.

When you publish a content object, eZ Publish automatically creates a virtual URL for you. In other words, you do not need to manage this manually for all content published on your site. However, it is possible to create additional virtual URLs, and this is covered later. When a new translation is added to an object, the system automatically generates

a new set of virtual URLs (based on the translations) for the node(s) that encapsulate that object.

3.3.1.3. URL mapping table for system and virtual URLs

eZ Publish maintains a table that maps system URLs to the corresponding, automatically-generated virtual URLs for all published content. This is recorded for each of the object's node assignments. In other words, if a content object has multiple locations, its system URL will be listed multiple times in the table, one for each location. Note that the URLs in this table are handled completely by the system and cannot be changed using the Administration Interface. The relationship between the content hierarchy and the URL table is illustrated below.



URL TABLE

Source/virtual address	System/destination address
Community	content/view/full/118
Company	content/view/full/137
Company/News	content/view/full/139
Company/Contact	content/view/full/154

Figure 3.7. Objects, nodes and the URL table

3.3.2. URL aliases

Even though eZ Publish generates a virtual URL for you when you publish a content object, in some situations you might want to create your own virtual URLs, called "URL aliases". In other words, a *URL alias* is an alternative, custom, virtual URL. Technically, a URL alias is any alias for a system URL, including both system-generated virtual URLs

and custom virtual URLs. For simplicity, "URL alias" is commonly used to refer to custom virtual URLs, since it is those aliases that you can control.

For example, you might want to replace "http://www.example.com/company/about_company" with "http://www.example.com/about". URL aliases are useful in many cases: to create even easier-to-remember URLs; to redirect users if content has moved; to achieve search engine optimization goals; and to provide multiple ways to access the same page.

3.3.2.1. Node URL aliases

Node URL aliases (or "node aliases" for short) are aliases for system URL destinations in this form: "content/view/full/node_id", where "node_id" is the ID number of a node. As previously mentioned, whenever you publish an object, eZ Publish automatically generates a virtual URL (per translation). This is considered an automatically-generated node URL alias. You can create and manage additional, custom node URL aliases as described in Chapter 12.

3.3.2.2. Global URL aliases

Global URL aliases (or "global aliases" for short) include all user-defined virtual URLs except node URL aliases. In other words, if the destination URL does not follow the "content/view/full/node_id" form, it is a global alias. We describe how to manage them next.

3.3.2.3. URL translator interface

The screenshot below shows the **URL translator** interface with no global aliases yet:

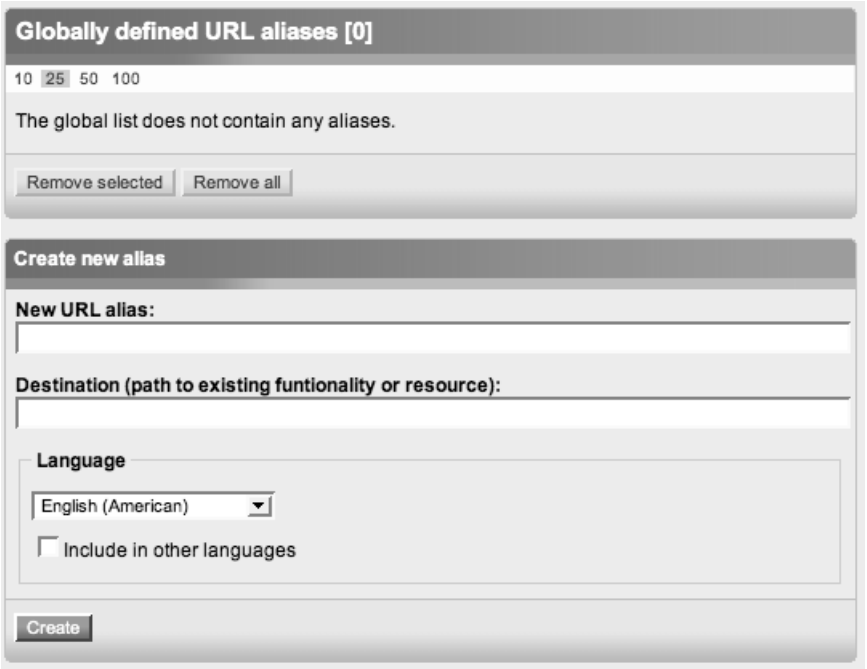


Figure 3.8. URL translator interface

The top window shows the list of global aliases (and lets you remove some or all of them), while the bottom window is used to create new URL aliases. The list is sorted by the alias text, not the path to which it points. The **Always available** column reveals whether the alias will work for all siteaccesses, regardless of their language configuration.

The **Create new alias** window has two input fields. The **New URL alias** field is for the desired alias, and the **Destination** field is for the existing virtual or system URL to which it should apply.

The **Language** dropdown list and the **Include in other languages** checkbox are used for multilingual URL aliases. URL aliases for nodes are created in accordance with the existing translations of the objects encapsulated by the nodes. The dropdown list will contain the site's translation languages, which are described in Chapter 12. For single-language sites, the list will contain only one language, and the checkbox does not have any practical use.

3.3.2.4. Creating a new global alias

New global URL aliases are created using the **URL translator** interface shown above. The required steps are outlined in the procedure below.

Tip

Node URL aliases are usually created in the **Node aliases** interface for the corresponding node, but can alternatively be added from the **URL translator** interface. However, they will not appear in the global list after being created.

1. Open the **URL translator** interface by clicking the corresponding link in the left menu of the **Setup** tab.

2. Input the desired URL alias, such as "findme", in the **New URL alias** input field.

3. Input the path to the existing functionality or resource, such as "content/search" (module/view combination) or "company/news" (node path), in the **Destination** input field.

4. For multilingual sites, select which site language the alias should be associated with in the **Language** dropdown list. Mark the **Include in other languages** checkbox to have the alias work for all siteaccesses regardless of their language configuration.

5. Click the **Create** button. The new URL alias will be added to the global URL list (except if it is a node URL alias):

Figure 3.9. New global URL alias

3.3.2.5. URL wildcards

We have now seen that there are system URLs and auto-generated virtual URLs, as well as node and global URL aliases. A *URL wildcard* is a URL alias that applies to multiple destinations at once. For example "/developer/*" can be an alias for "/dev/{1}". Each asterisk (*) in the alias corresponds to a number in curly brackets in the destination. The numbers match in ascending order of asterisks from left to right (if you have multiple wildcards in the same alias). URL wildcards are managed through the **URL wildcard** interface described below.

URL wildcards provide the following benefits:

- To compensate for moved content so that links to your content from external sites will still function. For example, if you had multiple nodes below an "education" folder that was renamed to "development", you could use wildcard URL forwarding to ensure that all links below the "education" folder would point to the new address. In this case, the alias would be "/education/*" and the destination would be "/development/{1}".
- To offer automatic correction of a URL that is often mistyped. For example, you could put "/prod*" as the alias and "/products" as the destination so that any variations in the spelling after "prod" would still redirect to the "products" folder.

3.3.2.6. URL wildcard interface

The **URL wildcard** interface is accessed by clicking the **URL wildcards** link in the left menu of the **Setup** tab. This interface is shown in the screenshot below.

Figure 3.10. URL wildcard interface

It is not very different from the **URL translator** interface shown in Figure 3.8, “URL translator interface”. The main differences are that it cannot be associated with a specific translation language and that it supports URL redirection (described below).

The top window shows the list of URL wildcards (and lets you remove some or all of them), while the bottom window is used to create new URL wildcards. The list is sorted by the alias text, not the path to which it points.

The **Create new alias** window has two input fields. The **New URL alias** field is for the desired alias, and the **Destination** field is for the existing virtual or system URL to which it should apply.

When the **Redirecting URL** checkbox is marked, when the URL alias is accessed, the URL in the browser will automatically change to the existing virtual URL. In this case, the entry in the **Type** column will display "Forward". When the checkbox is not marked, the URL in the browser will remain as the user typed it but the displayed page will still correspond to the destination URL. In this case, the entry in the **Type** column will display "Direct".

3.3.2.7. Creating a new URL wildcard

The following procedure outlines how to create a new URL wildcard, which redirects all URLs that start with "/prod" (such as "/producs" or "/produce") to "/products".

1. Open the **URL wildcard** interface by clicking the corresponding link in the left menu of the **Setup** tab.
2. Input the desired wildcard, such as "/prod*", in the **New URL wildcard** input field.
3. Input the path to the existing destination URL, such as "/products", in the **Destination** input field.
4. Mark the **Redirecting URL** checkbox if you want to use this feature.
5. Click the **Create** button. The new URL wildcard will be added to the URL wildcard list.

Figure 3.11. New URL wildcard

3.3.3. Link management

This section explores the **URL management** interface for inspecting and editing published URLs. These are links that are explicitly added by users, either by embedding the links in rich text content, or by using an attribute of the "URL" datatype. In other words, published URLs have nothing to do with system or virtual URLs. Note that links of the "eznode" and "ezobject" type are not part of link management. This is because they are

generated based on the specified Node ID or Object ID of an existing content object and are valid unless you remove the linked node or object.

These published URLs are stored in a separate part of the database, isolated from other content, and should not be confused with the previously described URL mapping table. The separation enables the system to automatically check the validity of the links (described later). It also enables you, through the **URL management** interface, to view and check all of the links on your site in one central location:

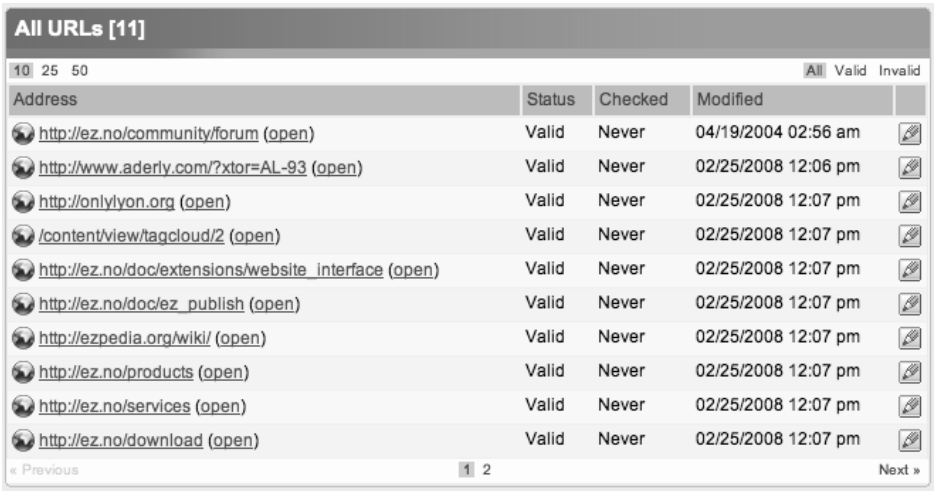


Figure 3.12. URL management interface

The above screenshot shows the **URL management** interface, accessed by clicking the corresponding link in the left menu of the **Setup** tab. In the top left corner, you can change the number of URLs to list per page. In the top right corner you can choose to filter the listed URLs to display all of the URLs or only the valid or invalid ones. If the list stretches over several pages, you will find a paginator at the bottom of the window.

The columns in the list display the following for each URL:

- **Address**: shows the URL as a clickable link (that will take you to a page showing information about the URL, such as which objects use it), and also a link labeled "open", which will open the URL in a new window.
- **Status**: displays "valid" or "invalid". There is a script (described in the in-depth section below) that site administrators can run to check the validity of all links.
- **Checked**: displays "never" or the date for when it was last checked.
- **Modified**: displays the date for when it was last modified.
- **Edit** button: used to edit the particular URL.

In Depth: Link checking script

eZ Publish comes with a script (`linkcheck.php`) that site administrators can run to check if the links on a site work. This script can be run on demand, or be scheduled to run periodically. When the script is run, it automatically attempts to access the URLs stored in the database table one by one, and records the responses.

All URLs are marked as valid by default. If the link checking script is run and receives a response containing an error message (like "404 Page not found"), or if no response is received at all, the URL will be marked invalid. In other words, the script identifies broken links on your site for you. Once a link is checked, regardless of the outcome, the time is recorded in the "checked" field. Subsequently, invalid URLs and the objects that are using them can be easily identified and edited using the **URL management** interface.

The following screenshot shows the **URL view** interface that is opened when you click a URL in the **Address** column.

Figure 3.13. URL view interface

The top window reveals a subset of the information found in the URL database table. Clicking the address link will open the page the URL specifies. At the bottom of this window is the **Edit** button. Clicking this button enables you to directly modify the URL through a simple interface. You can also access this edit interface by clicking the **Edit** button next to a link in the **All URLs** window.

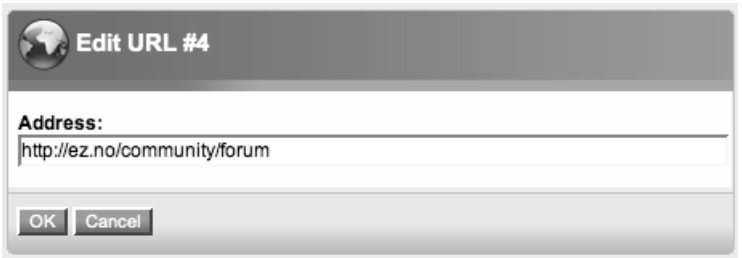


Figure 3.14. URL edit interface

The purpose of the **Objects using URL** window is to reveal where on your website this URL is used. Clicking the object name in the left column will bring up the **Version preview** interface for the relevant version of that object. Clicking the **Edit** button to the right of the entry will open the **Object Edit Interface** for that content object.

Therefore, if there is something wrong with a URL, you have the option to directly edit the problem URL or to edit the object(s) that use the URL. Modifications made through the **URL edit** interface will affect all occurrences of that URL. In contrast, if you choose to update the URL by editing an object that uses it, only that occurrence is modified.

3.4. Cache management

Cache is a mechanism commonly used to boost performance and the user experience in websites. The general idea is to keep frequently used information in a temporary storage area for rapid access. As a result, full pages, or parts of pages, do not have to be dynamically generated each time they are accessed.

Let us look at a real-life metaphor that explains the general principle behind caching. Suppose that an employee is responsible for going down to the local coffee shop and getting the daily morning cups for the other members of his team. On the first day of this arrangement, he has to go around and ask all of his team members what their order is, take a note and carry out the favor. The next day, he approaches them again asking for their order, and most of them answer "the same as usual". He has to admit that he cannot remember, ask for the full information all over again, and get the coffee. However, now he knows that the orders do not change, at least not very frequently. He then decides to keep the order note at his desk, and brings it along every morning he goes down to the coffee shop. His note now represents the cache, and the orders represent the different webpages.

The team members are pleased, getting their favorite coffees shortly after arriving at the office, without being bothered by the repetitive order questions. The employee who gets the coffee is happy because now he does not have to waste time running around asking the same question every day. At times, someone will let him know that from now on they want diet milk in their latte, or just today they need an extra strong espresso. He simply

updates his note for the person that came by, and all the team members get served as desired.

In eZ Publish cache is, by default, stored in special-purpose directories on the web server's file system. It can alternatively be stored in the database. Caching is tightly coupled with the module execution described in Section 1.3.3, “Module execution and web requests”. Also, recall that the template system is component-based (a page being made up of multiple templates), and that the result of a module execution is included in the main template (the pagelayout). In eZ Publish, just as there are many elements that are combined to generate pages, there are many cache types. Caching can range from INI settings to entire pages.

With eZ Publish, you can specify what to store in the cache and how long it should be kept there (called the *time to live* and measured in seconds). Certain cache items can be configured to never expire or to never be cached, expiring instantly. These issues are managed exclusively by developers and site administrators, who must balance the need to serve content quickly with the need to provide up-to-date content. Interested readers should refer to the *eZ Publish Performance Optimization* article series on `http://ez.no`.

Cache refreshing consists of two steps: clearing and regenerating the cache. While clearing is performed rather quickly (eZ Publish basically deletes one or more files on the web server), regenerating can be a time-consuming task, and can negatively affect your site performance if all caches are cleared at the same time. Both steps are normally performed by the system according to specific values and conditions. However, as an advanced content manager you can clear the different types of caches on demand, triggering the system to subsequently regenerate the cache.

3.4.1. Cache management interface

Cache management is mainly about how long cache should exist before it is renewed. For developers, this means to set values and conditions for automatic refreshes. For content managers and webmasters, this means to explicitly force the cache to be cleared when needed.

Sometimes advanced content managers and webmasters find that they need to bypass the predefined rules for when cached material should expire. For example, if you have updated a configuration setting (for example, you added an additional site language to a siteaccess), or made a change to a class definition, the effects will not show until the relevant cache is refreshed. Through the Administration Interface, you can force the cache to be refreshed using a simple interface. The screenshots below show part of the **Cache management** interface, which is accessed by clicking the corresponding link in the left menu of the **Setup** tab.

Figure 3.15. Cache management interface - Clear caches window

The **Clear caches** window is located at the top of the **Cache management** interface, and contains buttons for executing cache operations. Clicking one of these buttons will clear the caches belonging to the specified cache type group (described later), or all of the caches (upon clicking the **Clear all caches** button). Additional information is displayed in tooltips when you hover over each button.

The **Fine-grained cache control** window below enables more detailed cache control. Mark one or more of the checkboxes corresponding to specific cache types, then click the **Clear selected** button to clear them:

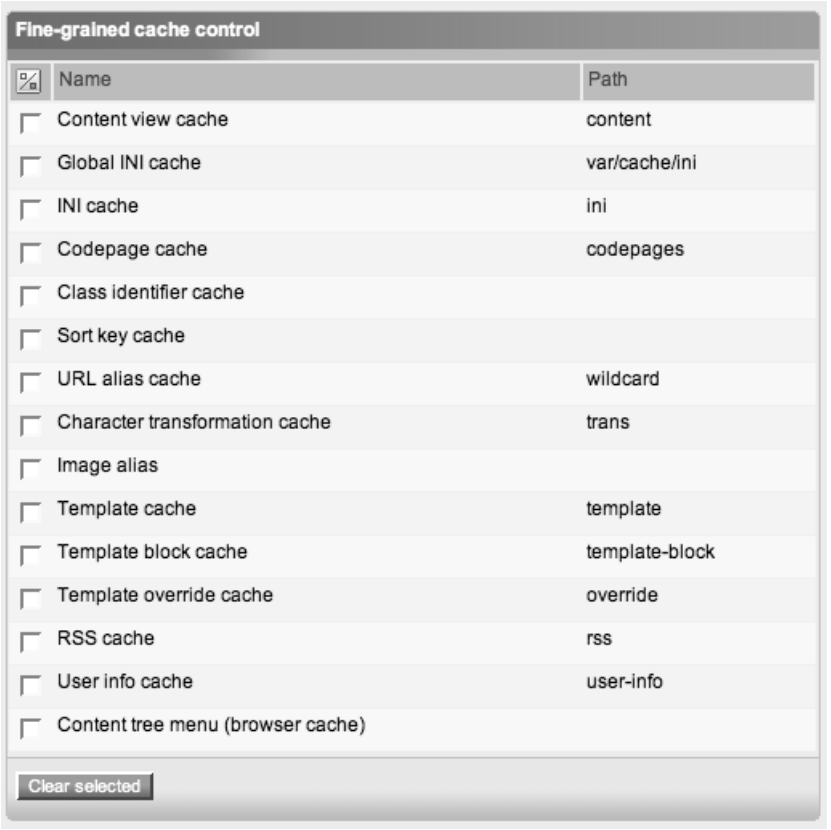


Figure 3.16. Cache management interface - fine-grained cache control

3.4.2. Cache types

The cache mechanism makes use of *cache types* in order to group similar cache files together and enable easy retrieval and management. This means that you can work with cache in terms of "INI cache" or "template cache", rather than individual files with names such as `875bf40b060cb1c4eaa46879dd6c9f85.php` (which is not very human readable, but suits the system well).

The most important cache types are further combined into three main groups:

- Template overrides and compiled templates
- Content views and template blocks
- Configuration (ini) caches

Recall that timely refreshing of caches is built into the system, and under normal circumstances will handle all of the cache management for you. For example, content view caching (described in the in-depth block below), ensures that content updates are followed by cache updates. Within the context of this book, you will most likely only need to clear the configuration (ini) caches, as described in Section 3.4.3, “Cache usage”. You may also find instructions to clear a specific cache type in some procedure, for example when installing an extension or setting up a new feature. As a result, we will not explain each of the cache types here and will subsequently refer to certain cache types on an as-needed basis. What is important is that you know the general cache concepts and the interfaces used to clear the cache. If you are unsure about which cache type to clear in certain situations, consult your site administrator.

In Depth: Content view caching

Content view caching (or "view caching" for short) is a fundamental part of the cache system. This mechanism only works for the "view" view of the "content" module (see Section 1.3, “Modules and views”). For now, it is sufficient to think of view caching as being used when you are viewing content on the front-end of a site.

Recall that the content displayed on your webpages is the published version of content objects and represented by nodes in the content hierarchy.

Whenever eZ Publish is requested to output information about a node (either by a system URL or a virtual URL), it executes the program code that is associated with the "view" view of the "content" module. Upon completion, the view returns a result to the module, which then returns it to the rest of the system.

The returned result is then used by the system when the main template - the pagelayout - is rendered. If view caching is enabled, the entire result of the module will be cached and recorded in a dedicated sub-directory on the web server. As a consequence, the "view" view of the "content" module will only be run if the system is unable to locate a view-cached version of the result; otherwise, a cached version will be inserted in the pagelayout.

A page may be displayed differently depending on access control parameters and user preferences. Different users who are logged in with unique permissions or preferences do not necessarily see the same content as anonymous users, or even as users with the same type of permissions and preferences. Because of this, eZ Publish builds not only one cache, but multiple view caches for the nodes based on roles and user preferences. Other parameters such as language and URL parameters can be used to further differentiate and customize cache files.

eZ Publish automatically updates parts of the view cache when a new version of an object is published. These parts include all published nodes of the object, the parent nodes, and nodes representing objects that have common keywords with the published object. The latter only applies if at least one of the attributes of the object uses the "Keywords" datatype (see Section 9.1.1, “Keywords datatype”). For example, if you create and publish an article about a football game, and enter "Sports" as a keyword, the view cache will be updated for other content objects that also have "Sports" as a keyword.

In addition, the view caches are cleared for nodes of related and reverse related objects that have relations of the "common" and "XML embedded" types. For example, if you modify an image object that is embedded in several articles, the caches for these articles are also cleared.

3.4.3. Cache usage

This section explains how to clear all caches and the configuration caches through the **Cache management** interface (shown in Figure 3.16, “Cache management interface - fine-grained cache control”). The procedures are similar when clearing other cache types. Cache files can also be manually deleted through the file system, but this is only recommended for expert users.

Tip

As a rule of thumb, you should clear cache as fine-grained as possible, and only when you update something and the changes do not take effect. For example, if you update an INI setting, you do not want all caches to be rebuilt, especially on a large site.

You should not clear all caches on a live site, particularly not at times with high load. If this is done when a lot of users are browsing your site, the web server might not be able to keep up. As a result, your entire site might become temporarily unavailable until the web server is restarted.

3.4.3.1. Clearing all caches

There are usually two situations in which you would want to clear all of the caches: you have been modifying many aspects of a development site and want to ensure that everything is regenerated to reflect the modifications; or you are not sure which specific cache type(s) to clear. The following list shows the ways to clear all the caches:

- Click the **Clear all caches** button in the **Clear caches** window.

- Click the **Invert selection** button in the top left corner of the **Fine-grained cache control** window to mark all cache types, then click the **Clear selected** button.
- Expand the **Cache** panel below the **Current user** panel in the right side of the Administration Interface, select "All caches" in the dropdown list, then click the **Clear** button.

3.4.3.2. Clearing configuration caches

Clearing the configuration caches forces the system to re-read the configuration files, which is necessary if you have changed some settings. The list below shows the alternative ways that you can clear the configuration caches:

- Click the **Clear Ini caches** button in the **Clear caches** window.
- Mark the checkbox corresponding to Global INI or INI cache (or both) in the **Fine-grained cache control** window, then click the **Clear selected** button.
- Expand the cache panel below the **Current user** panel in the right side of the Administration Interface, select "Ini settings" in the dropdown list, then click the **Clear** button.

3.5. The configuration model

This section covers the configuration model and the basic structure of INI files.

The first part explains the configuration model and uses the main site configuration file to illustrate concepts. This general approach is followed by an illustrated walk-through of the graphical user interfaces for viewing and managing these files through the Administration Interface. In other words, this is targeted at content managers who might need to check or finetune some settings, and is not by any means a complete guide for developers. For more technically advanced information, refer to the *eZ Publish Basics* book or http://ez.no/doc/ez_publish/technical_manual.

3.5.1. What is an INI file?

An eZ Publish configuration file, more commonly referred to as an *INI file*, represents settings that control the behavior of some specific part of the system, such as image sizes, the tree menu, payment gateways, and the Administration Interface. For example, the `image.ini` file contains information such as for height and width values for the different image sizes ("small", "original", "thumbnail" and so on), used in cases such as when there are embedded images in "XML block" attributes.

Configuration files are stored on the file system in a dedicated directory named `settings` (see Figure 1.2, “Directories below the eZ Publish root directory”). Both the hierarchy of folders and the structure of individual configuration files follow strict rules. The "INI file" name is attributed to the file extension `.ini`.

INI files are stored on the file system as text files that can be edited in your favorite text editor (you can also to do limited management in the Administration Interface, which will be discussed in Section 3.5.4, "Settings view interface"). This is in contrast to site content, which is stored in a database and always accessed through a site interface.

eZ Publish comes with a set of default configuration files. The default configuration files contain all the possible directives (with default settings) along with brief explanations. These default files serve as the base in the configuration override system, as explained next.

3.5.2. Configuration overrides

The *configuration model* consists of some INI files with settings and a *configuration override system*. It is this system that stitches information scattered throughout all the configuration files together to create just the right subset of settings for the current context. For example, it selects the specific design of the Administration Interface or the front-end public user design.

The `settings` directory requires a specific hierarchical structure. In this hierarchy, default configuration files are stored separately from overrides that are specific to a siteaccess, which again are separated from global overrides. This enables eZ Publish to check the relevant directories in a predefined order of steps, as illustrated below.

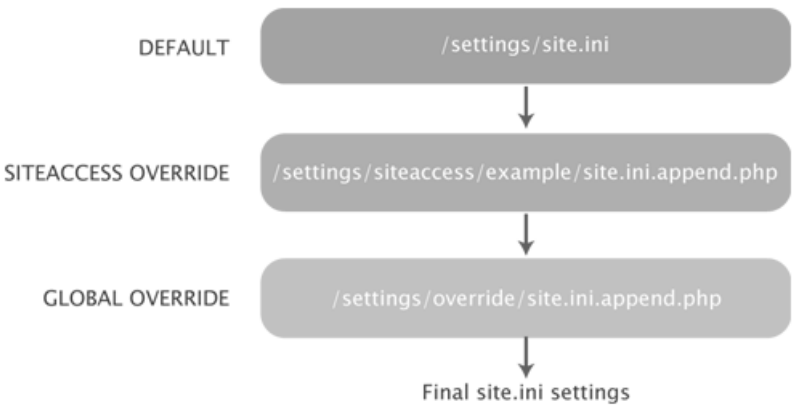


Figure 3.17. Configuration override example

The default configuration settings are overridden by settings specific to a siteaccess, which are again overridden by global override settings. In other words, settings are supplied with at least the default values in the first step; these default values can be replaced in step 2 and / or 3, with the final value of a setting being the one that gets used. Step 2 and 3 can also add new settings for customized solutions.

Warning

The default configuration files should only be used for reference and should never be modified. Changes should be made to either the global or per siteaccess override files. In this book, we explain how to modify settings through the **Settings view** interface, which does not allow you to modify the default configuration files.

In Depth: Format of an eZ Publish INI file

An eZ Publish configuration file has the following properties:

- The file is a PHP file, typically located in the *settings* directory. Configuration files are also found in other directories, such as the *settings* directory for an extension.
- The contents of the file are enclosed with PHP comment markings.
- The file is divided into blocks; each block contains a collection of settings.
- A setting can usually be set to "enabled" or "disabled", a string of text or an array of strings. If the name of a setting ends with a pair of square brackets, this means that the setting accepts an array of values.
- Lines starting with a hash (#) are comments that provide more information to users about the settings. These lines are not parsed by eZ Publish.

Let us illustrate this with an example extracted from the main configuration file, *site.ini*:

```
...
# This line contains a comment.
[DatabaseSettings]
DatabaseImplementation=ezmysql
Server=localhost
User=bms
Password=123456
Database=happyfeet
Charset=
Socket=disabled

# This line contains another comment.
[ExtensionSettings]
ActiveExtensions[]
```

```
ActiveExtensions[]=ezwebin
ActiveExtensions[]=ezdhtml
ActiveExtensions[]=ezodf
...
```

Above, there are two blocks, titled "[DatabaseSettings]" and "[ExtensionSettings]". The first block tells eZ Publish the necessary information about the database so that content can be stored and retrieved correctly. The second block lists the extensions to be used. In this example, they are "ezwebin" (giving the Website Interface), "ezdhtml" (giving the Online Editor), and "ezodf" (giving the OpenOffice.org import / export feature). You will learn more about extensions in Section 3.6, “Extension management”.

3.5.3. Main configuration file

The main configuration file in eZ Publish is *site.ini*. Here, and in the overrides of it, developers and administrators configure important information, including which database and design to use; the name and path to the directories your installation uses; available extensions; and various settings about templates, users, caching, siteaccesses and more.

We have already looked at the "[DatabaseSettings]" and "[ExtensionSettings]" blocks in "In Depth: Format of an eZ Publish INI file" above. The settings in the "[DebugSettings]" block regulate if, what and to whom debugging information should be displayed. When debug output is enabled during development, you will frequently see some text output at the bottom of a webpage beneath the regular content. However, when you site has gone live, this should be disabled entirely or limited to be shown when your developers or site administrator access the site to explore a problem or implement additional features. If you see debug information when it should be hidden, you should contact these people and ask them to check the "[DebugSettings]" block.

Settings in the "[DesignSettings]" block are typically overridden for each siteaccess. The *site design* is considered the most significant design resource, and eZ Publish will always search for templates in this design first. For example if you want to "skin" your site, this settings block must be updated accordingly.

A final example is the "[MailSettings]" block, which regulates email communication to and from your site. For example, you might want to update the "AdminEmail" setting to specify the email address to which administrator notifications - including important updates when new users are created or webshop orders are received - should be sent. It is also possible to set the default sender address for email sent from your site. This is particularly useful if you want to redirect all replies to a no-spam address.

3.5.4. Settings view interface

The left menu of the **Setup** tab has two links for accessing configuration files. Clicking the **Global settings** link will bring up the **Object Edit Interface** for *common INI settings* - these are global overrides and will affect your site as a whole. Clicking the **Ini settings** link will bring up the **Settings view** interface, which is described below. Global settings and the "Ini setting" datatype have been deprecated, and we recommend that you use the **Settings view** interface for configuration file modifications. Ask your site administrator or look in the technical documentation at http://ez.no/doc/ez_publish/technical_manual/ if you do not know which configuration file contains what you are looking for.

The **Settings view** interface of the **Setup** tab has two main uses. First, you can use it to check the value of a setting. This is very handy for content managers, since you can find information about the setting that is affecting one of your tasks through the Administration Interface, without having to access the file system. Second, you can use this interface to modify settings.

Tip

When you have opened the **Settings view** interface for a given configuration file, use the search feature in your browser to quickly locate a settings block or specific setting. Keep in mind that block and settings names are case sensitive and do not have any white space.

Some special-purpose interfaces of the **Setup** tab (such as the **Language management** interface; see Chapter 12, *Working with multilingual sites*) may involve indirectly working with the settings contained in INI files. They edit some of the INI files, but you will not be explicitly viewing the settings in the text files.

3.5.4.1. Viewing the settings of a configuration file

To bring up the **Settings view** interface for a particular configuration file, follow this procedure:

1. Click the **Ini settings** link in the left menu of the **Setup** tab.

2. Specify which configuration file to view, and for what siteaccess:

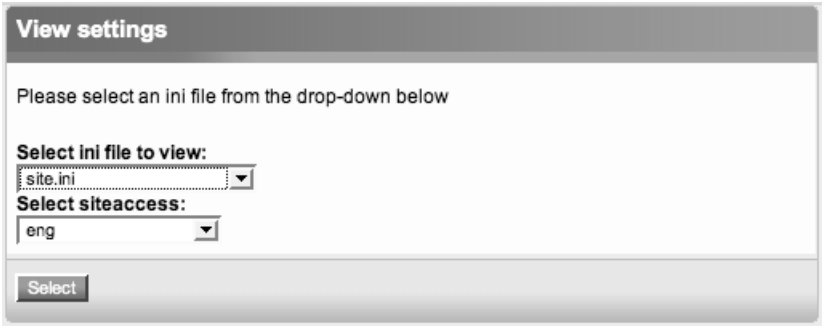
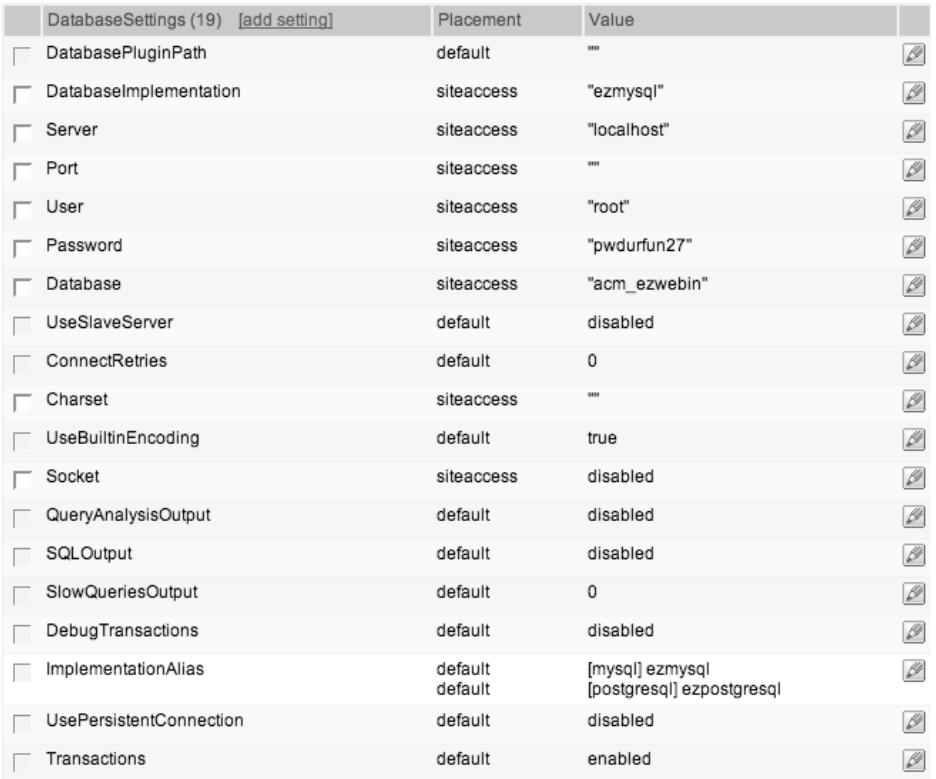


Figure 3.18. Settings view interface - select file and siteaccess

3. Click the **Select** button to reload the page with an additional window below the **View settings** window, listing all of the settings in the configuration file. The name, placement and value is listed for each setting, grouped according to the block structure. The placement, such as "default", "siteaccess", "override" or "extension", refers to the step in the override process (as previously described) in which the value was determined.



DatabaseSettings (19) [add setting]	Placement	Value
DatabasePluginPath	default	""
DatabaseImplementation	siteaccess	"ezmysql"
Server	siteaccess	"localhost"
Port	siteaccess	""
User	siteaccess	"root"
Password	siteaccess	"pwdurfun27"
Database	siteaccess	"acm_ezwebin"
UseSlaveServer	default	disabled
ConnectRetries	default	0
Charset	siteaccess	""
UseBuiltinEncoding	default	true
Socket	siteaccess	disabled
QueryAnalysisOutput	default	disabled
SQLOutput	default	disabled
SlowQueriesOutput	default	0
DebugTransactions	default	disabled
ImplementationAlias	default default	[mysql] ezmysql [postgresql] ezpostgresql
UsePersistentConnection	default	disabled
Transactions	default	enabled

Figure 3.19. Settings view interface with settings for a configuration file

The above screenshot shows the "[DatabaseSettings]" block for the "eng" siteaccess, which is the default front-end public siteaccess. If you compare the values to the settings listed in "In Depth: Format of an eZ Publish INI file" previously, you will recognize many of the entries.

3.5.4.2. Editing a settings value

The value of a specific setting listed in the **Settings view** interface can be edited through the **Settings edit** interface. As an example, we go through the procedure to change the email address of the site administrator.

1. Access the **Settings view** interface for the desired configuration file and siteaccess, as described in previous procedure. In our example, we access the *`site.ini`* file for the "eng" siteaccess.

2. Locate the desired block and setting, and click the corresponding **Edit** button. In our example, we locate the "AdminEmail" setting in the "[MailSettings]" block. This will open the **Settings edit** interface:

Figure 3.20. Settings edit interface

3. Ignore everything above the **location** radio buttons. You can select a different radio button, but in most cases this is not required.

   Update the contents of the **Settings value** input field as desired. For example, replace "bms@ez.no" with "webmaster@ez.no".

4. Click the **Save** button. The **Settings view** interface is re-loaded to show the same INI file for the same siteaccess, where you should be able to confirm the result of what you just edited.

You can remove existing setting overrides by marking the corresponding checkbox and clicking the **Remove selected** button at the bottom of the listing in the **Settings view** interface. In most cases, we recommend that you do not remove any overrides, and that you check with your site administrator before doing so. This is only available for entries

that have the "override" or "siteaccess" placement, because the default configuration files contain essential settings that should never be altered.

3.6. Extension management

This section explains the concept of eZ Publish extensions and shows how to manage them through the Administration Interface. We close with an example of how to enable an extension, in this case the Online Editor. Developer documentation on how to create and upload an extension can be found in the *eZ Publish Basics* book.

3.6.1. What is an eZ Publish extension?

An extension is basically a plugin to eZ Publish that provides additional custom functionality, without requiring major changes to the underlying system. For example, if your company decides to use the built-in Webshop feature, a payment gateway can be added via an extension.

The plugin system enables you to preserve custom-made extensions when the core eZ Publish installation is upgraded to a newer version. Also, extensions can be re-used on multiple eZ Publish installations. For example, the "ezwebin" extension can be added to any eZ Publish site to provide the Website Interface.

Extensions can be created by eZ Systems, eZ Partners, or any eZ Publish user. An extension usually comes with its own documentation explaining how to install and use it.

The following extensions are covered in this book:

- Online Editor (ezdhtml); see Section 2.1, “Online Editor”
- Website Interface (ezwebin); see Section 2.2, “Website Interface”
- eZ Flow (ezflow); see Section 2.5, “Frontpage management with eZ Flow”
- eZ Find (ezfind); see Chapter 11, *Search engines and finding content*
- eZ Open Document Format (ezodf); see Chapter 14, *OpenDocument Text and Microsoft Word documents*

Installing an extension consists of a minimum of two steps: extracting the files to the appropriate location on the web server's file system, and activating the extension. Content managers deal mostly with the second step, which involves registering the extension in eZ Publish either through the Administration Interface or a configuration file, for the entire site or for a subset of the siteaccesses.

In Depth: eZ Publish 4 autoload generation

eZ Publish 4 uses the PHP 5 *autoload feature* to load program code. Because of this, when new (or updated) code is introduced to the system (when performing an eZ Publish upgrade or adding an extension), the autoload arrays need to be updated. Contact your system administrator for assistance.

3.6.2. Managing extensions through the Administration Interface

The **Setup** tab provides an intuitive and simple interface for enabling and disabling any extensions that have been properly placed in the `extension` directory on the file system (see Figure 1.2, "Directories below the eZ Publish root directory").

The following screenshot shows the list of available extensions that is displayed when you click the **Extensions** link in the left menu. It is usually referred to as the **Available extensions** window or the **Extension configuration** interface.

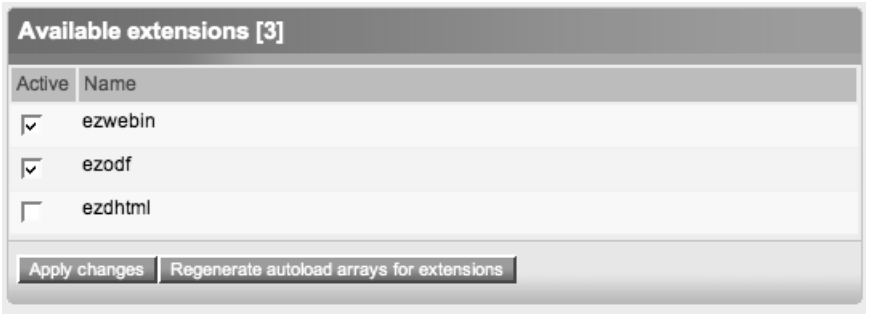


Figure 3.21. Available extensions window, Online Editor disabled

The title bar reveals the number of available extensions. These are listed with checkboxes indicating whether each extension is active. In this case, all of the names are prefixed by "ez", indicating that the extensions are made by eZ Systems.

To enable an extension, simply mark the checkbox for it and click the **Apply changes** button. To disable an extension, unmark the checkbox and click the button. In both cases, the action is carried out without a confirmation dialog. The procedure is explained in more detail in the following example.

3.6.3. Example: Enabling the Online Editor extension

This example gives you a walk-through of the procedure required to enable an eZ Publish extension.

Follow the steps below to enable the Online Editor extension (ezdhtml), assuming that it has been installed on your system (in other words, it appears in the **Available extensions** window, but is not marked as active). This situation might occur if the Online Editor was de-activated during development or if it was upgraded.

1. Make sure you are viewing the **Available extensions** window, accessed by clicking the **Extensions** link from the left menu of the **Setup** tab.

2. Mark the checkbox for the "ezdhtml" item and click the **Apply changes** button.

3. Clear all the eZ Publish caches as previously described.

4. Clear the *browser caches*. This is different from the eZ Publish cache, which is managed on the web server. Browsers usually cache visited pages in a directory on the local file system. The next time the same pages are visited, they are loaded faster because some of the information already exists on the local file system. While this feature improves the speed at which pages load, sometimes it may cause problems when you are updating the Online Editor. The method for clearing the browser cache varies between different web browsers. For Internet Explorer, it is found under "Internet Options" in the "Tools" menu. In Mozilla-based browsers, it is usually somewhere under "Advanced Preferences".

3.7. The workflow system

eZ Publish comes with an integrated workflow mechanism that makes it possible to perform different tasks with or without user interaction. It lets you specify a sequence of actions that will be executed before or after the completion of a module's function. In other words, it adds actions to the module execution process described in Section 1.3, “Modules and views”. For example, you can instruct the system to send emails to notify users whenever certain objects are edited, or you can require user approval before certain objects are published. Within this book, the concept of workflows particularly applies to the previous material on notifications, and the subsequent material on delayed publishing and the collaboration system. If these topics do not apply to you, this section can be skipped.

Here, we provide an introduction to the workflow system and its components. We also briefly cover the requirements for this system to work. However, it is by no means a complete technical guide to workflow management. Interested readers should refer to http://ez.no/doc/ez_publish/technical_manual.

3.7.1. Components of the workflow system

The *workflow system* consists of four components: events, workflows, workflow groups and triggers. The relationship between these components is illustrated below:

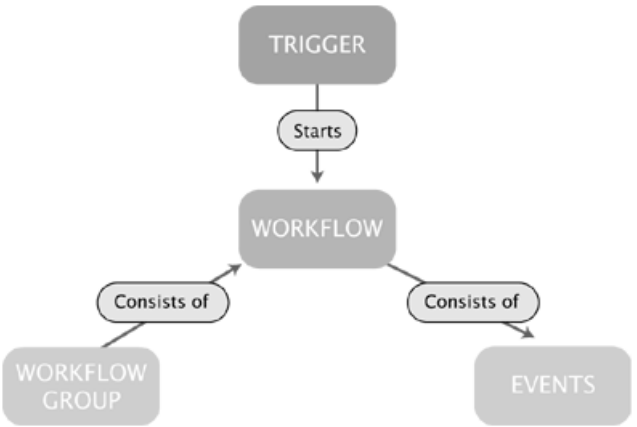


Figure 3.22. Components of the workflow system

The following sections explain each component. It is assumed that you are familiar with modules and views, which were introduced in Section 1.3, “Modules and views”.

3.7.1.1. Workflow events

An *event*, or more precisely a *workflow event* (not to be confused with calendar events), is the smallest entity of the workflow system.

A single event carries out a specific task. Currently, eZ Publish comes with the following workflow events: "Approve", "Multiplexer", "Payment gateway", "Simple shipping" and "Wait until date". The "Approve" event is vital to the collaboration system (see Chapter 8), in which certain objects must be approved by an editor before they are published. The "Wait until date" event is used for delayed publishing (see Chapter 7). These two events are explained in the corresponding chapters. The remaining events are not covered in this book.

It is also possible to add custom events to extend the system. These have to be programmed in PHP, and should only be added by skilled developers.

3.7.1.2. Workflows

A *workflow* is a sequential list of events that is initiated by a trigger. In other words, a workflow is an ordered sequence of actions. It consists of a name and one or more events. Workflows can be managed through the **Setup** tab. The **Workflow management** inter-

faces use special-purpose views provided by the "workflow" module. Clicking the **Workflows** link in left menu of the **Setup** tab displays a list of workflow groups. Clicking on one of the group names will display more information about that particular group and a list of workflows that belong to it. "Standard" is an empty default workflow group (see Section 3.7.1.3, “Workflow groups”) that comes with eZ Publish installations. Here, we have added one workflow to it for illustrative purposes.

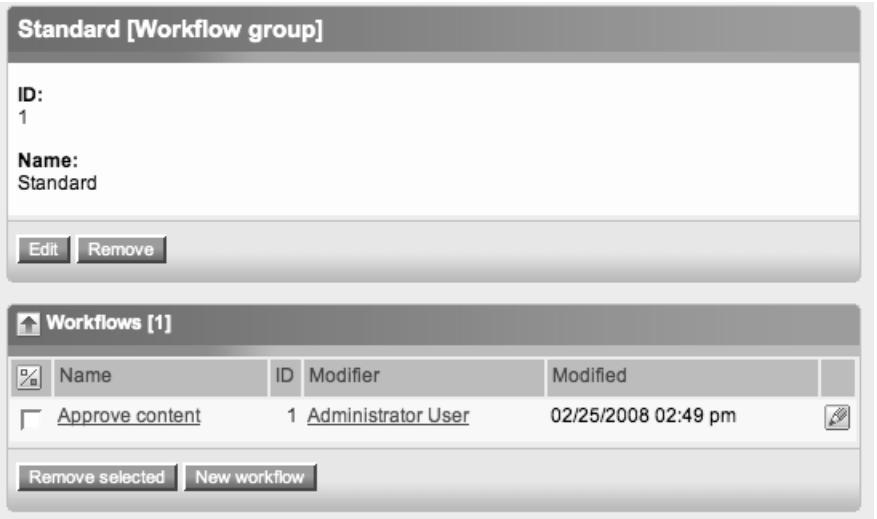


Figure 3.23. Workflow list

To view more information on a particular workflow, click on its name:

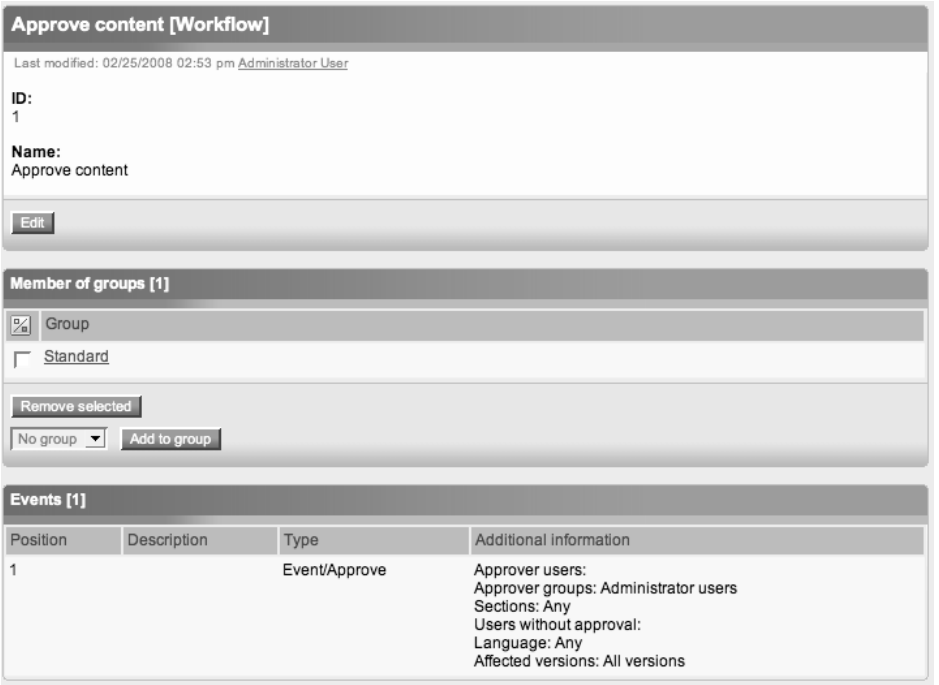


Figure 3.24. Workflow view

Detailed descriptions of the **Workflow management** interfaces and the associated operations are beyond the scope of this book. Below, we include a short introduction to the **Workflow editing** interface. The general idea is to enable you to open these interfaces and explore the information they display. In other words, this is for illustrative purposes only. We will be referring to these interfaces in Chapter 7, *Delayed publishing* and Chapter 8, *Collaboration system*.

3.7.1.2.1. Workflow editing interface

The **Workflow editing** interface is used when creating or modifying a workflow. To access it, click the **New workflow** button at the bottom of the workflow list (see Figure 3.23, “Workflow list”) or the **Edit** button corresponding to the workflow you wish to modify. At minumum, a name and one event needs to be specified:

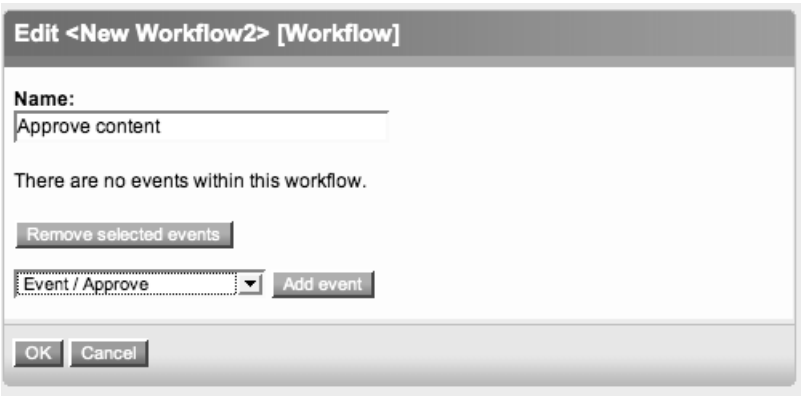


Figure 3.25. Workflow editing interface

Click the **Add event** button after selecting the desired event from the dropdown list. This will reload the page with an event configuration area inserted between the input fields of the above screenshot. This area will vary slightly depending on the event type.

Figure 3.26. Workflow editing interface with "Approve" event

3.7.1.3. Workflow groups

A *workflow group* is a collection of workflows. It has a name, and you can add or remove workflows to it. Practically, a workflow group is just a structuring tool, like a simple folder. It has no other usage other than to structure workflows.

3.7.1.4. Triggers

A *trigger* initiates a workflow. It is associated with a function of a module, and will start the workflow before or after the function is completed. Let us look at this through a concrete example. The "content" module has a "publish" function, which handles the normal process from when you click the **Send for publishing** button in edit mode until the result appears on the front-end of the site. In order to have an editor approve an object after it has been created, but before it is publicly displayed, a trigger is needed to start an "Approve" workflow event before the "publish" function finishes.

This concludes our exploration of the four components of the workflow system. The following section on requirements is a bit more technically challenging and may be skipped by those who are not interested in a look behind the scenes of the workflow system.

3.7.2. Workflow-related requirements

Proper operation of the workflow system depends on the following elements being present and correctly configured:

- The `workflow.ini` configuration file must contain the correct information.
- The workflow in question must be created (once).
- The workflow must be associated with a trigger (once).

Some workflows also require the `workflow.php` script to be run, either automatically at periodic intervals (see the in-depth block below about cronjobs) or manually. For example, the built-in "Approve" event (used in the collaboration system; see Chapter 8) and "Wait until date" event (used in delayed publishing; see Chapter 7) require the `workflow.php` script, while the "Payment Gateway" event does not. Therefore, for some aspects of workflows (such as an article being published at a future date through the delayed publishing mechanism), even if you have set up your workflow and trigger correctly, nothing will happen unless the workflow script is run. Let us illustrate this through an example. Think of how the public mail system works. You write a letter, put it in an envelope, address it, affix a stamp, and place it in the mail box. This does not make the letter appear at the intended recipient. If the postal service is on strike, the letter will not be delivered.

In Depth: Cronjobs

In eZ Publish, a *cronjob* refers to maintenance tasks that can be run periodically by the `runcronjobs.php` script. The usual procedure is for a system administrator to specify when `runcronjobs.php` should be run, then the operating

system will process the job in the background at the scheduled intervals or times. On UNIX / Linux systems, this can be done by making use of the "cron" system. On Windows, the script can be run by the "Scheduled Tasks" service.

The purpose of the `runcronjobs.php` script is to run other cronjob scripts, which are used by eZ Publish to perform tasks without direct user interaction. For example, the `linkcheck.php` script checks entries in the URL table for broken links. Another script generates email notifications, such as for all subscribed users when a post is added to a given forum topic. (Such scripts are included in eZ Publish.) The settings in the `cronjob.ini` configuration file control which cronjob scripts to execute when `runcronjobs.php` is run.

Interested readers should refer to the "Cronjob" entries of the *Features* section at http://ez.no/doc/ez_publish/technical_manual.

3.8. Summary

This chapter looked at the **Setup** tab of the Administration Interface, which is associated with the Setup branch of the content node tree and displays the main site configuration area. The contents of this tab include special-purpose interfaces for managing various aspects of your site. The layout is similar to that in three left-most tabs, except that the left menu is a flat list of links and the main area windows are not associated with a row of switches.

A Uniform Resource Locator (URL), commonly also referred to as a "web address", instructs a browser what you want to view and where to find it. In eZ Publish, two types of URLs co-exist. *System URLs* are the raw identifiers used internally and normally not displayed to site visitors. *Virtual URLs* are automatically generated by eZ Publish and are more user friendly representations of system URLs. You can create your own virtual URLs, called *URL* aliases.

Cache is a feature commonly used to boost performance and user experience in websites. The general idea is to keep frequently used information in a temporary storage for rapid access so that pages and page fragments do not always have to be dynamically generated.

The *configuration model* is based on a collection of configuration files, also known as "INI files", representing settings that control the behavior of some specific part of the system. An override system makes it possible for eZ Publish to use a configuration combination based on the current context.

An *extension* provides additional custom functionality to the core of the eZ Publish system.

A *workflow* is a sequential list of events that is started by a *trigger*. The initiation can be done before or after a function of a module completes, depending on the event. Workflows

are used in, among other things, the delayed publishing feature (see Chapter 7) and the collaboration system (see Chapter 8).

Chapter 4. Sections

The purpose of this chapter is to explain the *section* concept of eZ Publish, identify the related features and show how sections are used and managed. Sections group virtual collections of nodes. On their own, sections are nothing more than identification numbers. However, they can be used by eZ Publish in several different ways, which are listed later in this chapter.

Recall that:

- The **Setup** tab in the Administration Interface represents the main site configuration area and you need administrator-like permissions to access it.

This chapter is relevant for advanced content managers and webmasters responsible for setting up and managing sections in an eZ Publish site. This is important in situations such as defining structural content segmentation and setting up a protected area. It is useful when managing webshop discount rules (see Chapter 15) since one of the discount rule limitations is the Section limitation. This chapter is also useful for those who want to learn more about the content node tree.

In this chapter, you will find information about the following:

- The node tree and top-level nodes
- What a section is and how it can be used
- How segmentation of the node tree works
- Section management and the **Sections** interface

This book includes an example on how to create a protected area by using sections. However, since this also requires knowledge about user management, which is covered in the following chapter, you will find the example in Section 5.8, “Example: Creating a protected area”. Moreover, the relationship between templates and sections is introduced here, but readers interested in learning more about templates should consult the technical documentation at http://ez.no/doc/ez_publish/technical_manual.

Readers with a good knowledge about the content node tree and top-level nodes may skip directly to Section 4.2, “What is a section?”. Setting up section permissions is covered in Section 5.7, “Access control usage”.

4.1. Content node tree and top-level nodes

This section explores the properties of the content node tree beyond the introductory information found in Section 1.2.1, “The content hierarchy and nodes”. We start by revisiting Object and Node IDs, as well as the location concept. Then, we will explain the top-level nodes.

4.1.1. Object and Node IDs

Each content object has a unique *Object ID*, and each node has a unique *Node ID*. The Node ID numbers are used by the system to organize and keep track of the nodes. The *Parent node ID* reveals the node's superior node in the tree. The Object ID in a node pinpoints the specific object that the node encapsulates. This was illustrated in Figure 1.5, “Object - node relation”.

4.1.2. Locations

Let us review the differences between main and secondary locations. It is crucial to understand these concepts before delving into section management.

First, a node is a *location* for a content object within the tree structure, and refers to exactly one content object. Objects, on the other hand, can be referenced by more than one node, thereby appearing at multiple locations in the content node tree. This feature is known as *cross-publishing*. Note that cross-publishing is different than copying nodes. With cross-publishing, multiple nodes encapsulate one object. With copying, a new node is created that encapsulates a new object.

Second, when an object has multiple locations (and is thus associated with more than one node), only one node can be considered the main node of the object. The *main node* usually represents the object's original location in the tree (where it was first put) but can be changed. Among other things, the main node is used to avoid duplicate search hits. A search result returns only the main nodes. Also, access permissions for different sections are applied to the main nodes. The other nodes can be thought of as additional or secondary nodes (or locations). Adding a secondary location for a container object such as a folder will not duplicate the nodes below the container. In other words, cross-publishing only affects the specific object.

4.1.3. Top-level nodes

Let us now take a step back and look at the glue that binds together all of the branches of the content node tree. At a minimum, the tree consists of one node, called the *root*

node. This is the ultimate folder under which all other nodes are placed, and its properties include the following:

- The root node is the only node that does not have a Parent node ID, simply because it is at the top of the hierarchy.
- It is not associated with a content object. In other words, it is a virtual, empty construct without any content.
- Its only function is to be a parent node to the other nodes, to hold the tree together.
- You cannot navigate to it, cannot see it and cannot delete it. It simply exists.

A node that is directly below the root node is called a *top-level node*. Each top-level node is the start of a node tree branch, such as the well-known Content or Media branches. The following illustration shows the virtual root node and the standard top-level nodes:

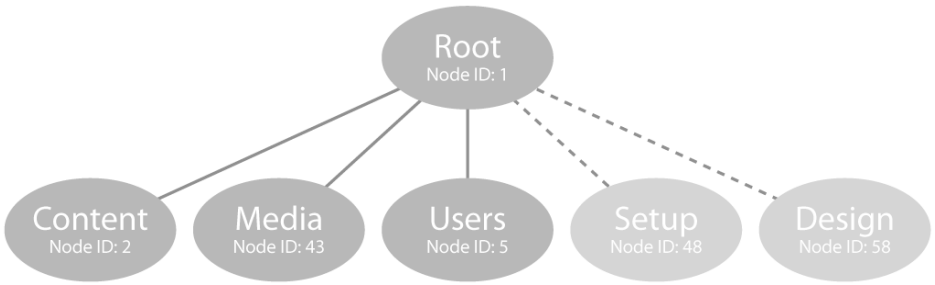


Figure 4.1. Top-level nodes

The properties of these five top-level nodes are listed in the table below. Note that you cannot remove any of these nodes, since they constitute a vital part of the system. However, it is possible to *swap* a top-level node with another node in order to, for example, change its content class. This is described in Section 10.2, “Swapping nodes”.

Table 4.1. Default top-level nodes

Top-level node name (Node ID)	**Description**
Content (2)	Parent of the Content branch and associated with the **Content structure** tab. This is the "Home" Frontpage object. It holds the majority of the content that is published on your site and is closely related to the overall organization of content and the sitemap.

Top-level node name (Node ID)	Description
Media (43)	Parent to the Media branch and associated with the **Media library** tab. This top-level node usually represents a restricted part of the site. The media library was described in *eZ Publish Content Management Basics*.
Users (5)	Parent to the Users branch and associated with the **User accounts** tab, described in Chapter 5. This top-level node usually represents a restricted part of the site.
Setup (48)	In contrast to the above top-level nodes and branches that represent ordinary content objects, the Setup branch contains configuration-related nodes that are used internally. The Setup node is associated with the **Setup** tab described in Chapter 3.
Design (58)	Contains miscellaneous nodes related to design issues for internal use, such as "look and feel" settings. It is associated with the **Design** tab.

Tip

You will normally see the names, not IDs, of the top-level nodes when working in the Administration Interface. However, knowing these special Node IDs, you will be able to recognize them within the Administration Interface as shown in the illustration below, and to use them in situations such as tweaking the sitemap view (see Section 10.1, “Viewing a sitemap”).

Figure 4.2. The Content top-level node represented as the "Home" frontpage

4.2. What is a section?

The word "section" can actually refer to several things within eZ Publish. There is a module (see Section 1.3, “Modules and views”) named "section" that provides an interface to the content engine and has views for managing sections. A section also refers to a deprecated template control structure. This definition should no longer be used, but you might come across old template syntax or references to it if you search for "section" in the documentation pages on http://ez.no.

The most important interpretation of the word "section" is as an identification number, as noted in the introduction. On its own, a *section* consists of only a *section ID*, a name and a navigation part. It is best to think about a section in terms of how its ID is used. Consequently, the most applicable definition is "a virtual collection of nodes that belong together". In other words, it is a logical hierarchy or structure, different than the content node tree but applied to the nodes within it, as shown in the illustration below.

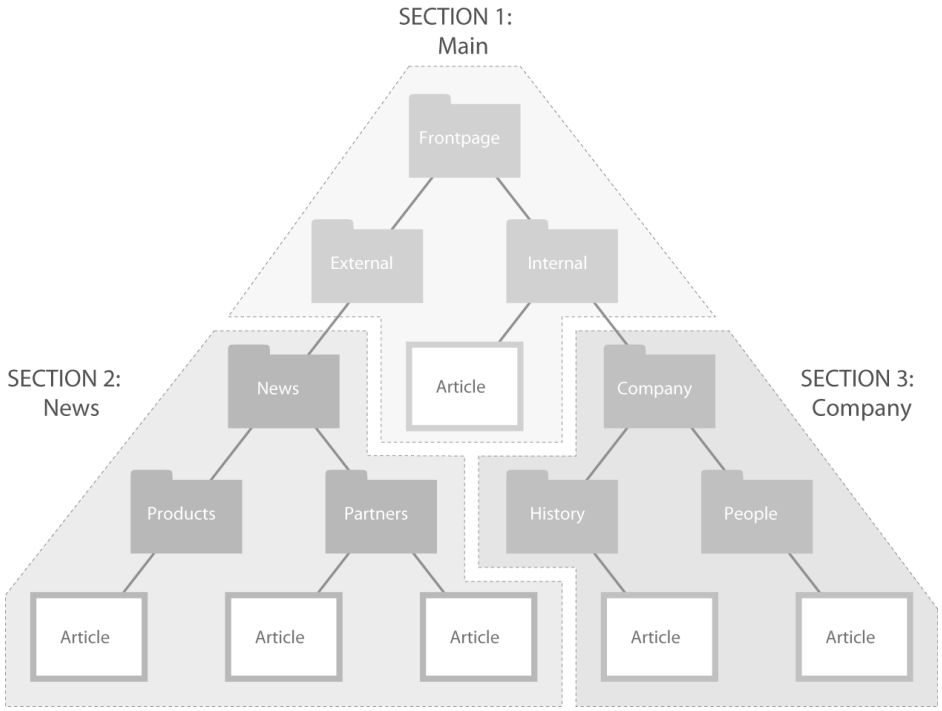


Figure 4.3. Example scenario with three content sections

Tip

Sections can only be added, modified and removed through the Administration Interface (see Section 4.3, “Section management”).

We start off our conceptual exploration of sections by looking at some of their uses. Then, we will explore the properties of a section a bit more, followed by an outline of the built-in sections.

4.2.1. Section functionality

The following functionality is provided through the eZ Publish section mechanism:

- Segmenting the content node tree into different subtrees, shown in Figure 4.3, “Example scenario with three content sections” above. Segmentation is usually a prerequisite for the other features listed below.
- Setting up custom template override rules. By using sections in the configuration of when to use which template, it is possible to have only a few rules that apply to all nodes within a logical group, in contrast to having a large collection of special-case rules. For example, you could choose a different layout for news articles than for company articles, according to their section. If the class attributes are the same, there is no need to define multiple classes. This is described in *eZ Publish Basics*.
- Limiting control and access to content (see Section 5.7.2, “Managing roles and policies”).
- Specifying content that is to be subject to approval (see Chapter 8, *Collaboration system*).
- Assigning discount rules to a group of products of the Webshop (see Section 15.8, “Discount rules”).

For advanced content managers, you will be working mostly with the first and third features. The fourth feature is related to the collaboration system and the fifth feature is applicable if your site has a Webshop.

4.2.2. Section properties

A *section ID* is an identification number that can be assigned to an object and denotes which section that object belongs to. It is stored in a special-purpose attribute within a content object, and you will not see this attribute in the main editing window. A section can be assigned to many objects, but a single object can have only one section, as was illustrated in Figure 4.3, “Example scenario with three content sections”.

It is the assignment of different section IDs to different objects that enables you to have a logical hierarchy in parallel with your sitemap-like content hierarchy. Your site can have as many virtual collections of nodes as you have section IDs.

Each section ID is also associated with a more user-friendly name, such as "Standard" or "Restricted", and a navigation part, such as the **Content structure** or **Webshop** tab. The name is used for easy identification of sections in the Administration Interface. The *navigation part* determines which main menu tab is accessed when you work with an object. In other words, when an object is requested in the Administration Interface, the system looks up the section that the object belongs to; the system then looks up the navigation part of that section and then displays the object in the tab as specified by the navigation part.

Within the Administration Interface, you will usually be presented with the section name rather than the ID, in the same way as with content objects. However, note that you can have multiple sections with identical names, similar to how you can have distinct objects with the same name.

In Depth: The ezsection object

An *ezsection* object stores the ID number of the section, the name of the section and the identifier of the navigation part with which the section is associated. It is this object that you are editing when creating or modifying a section, as is explained in Section 4.3, “Section management”. Do not confuse ezsection objects with content objects that are encapsulated by nodes in the content node tree.

4.2.3. Built-in sections

The following list shows the built-in sections for sites with the Website Interface. Note that the first five sections correspond to the top-level nodes (although the section corresponding to the Content top-level node is named "Standard") as listed in Table 4.1, “Default top-level nodes”.

- Standard
- Media
- Users
- Setup
- Design
- Restricted

The Setup section is used to isolate configuration-related objects from ordinary content. This is achieved in conjunction with the permission and template systems. Similarly, the Design section is associated with the **Design** tab. It isolates template and look settings (such as the site title and URL, labels and footer text) from ordinary content and other configuration. The Restricted section is a protected area of the site, as achieved in conjunction with the permission system. By default, it is applied to a partner area, which can only be accessed by users in the Partner user group (or higher). Additional sections can be added as needed, as described later.

4.2.4. Section inheritance

This section introduces section inheritance associated with object creation, publishing and multiple locations.

When an object is first created, but is not yet published and thus has a "Draft" status, its section is set to the default Standard section. Recall that an object is not visible to site visitors, nor does it show up in the left menu of the Administration Interface until it has been published. In other words, a Draft object does not have a location within the hierarchy, and it is only accessible to the user creating it. As a result, it does not matter what the section ID is at this time.

When the object is published, usually when you click the **Send for publishing** button, it automatically inherits the section ID assigned to the object encapsulated by its parent node. For example, if you create and publish a new object within a folder belonging to the Media section, the Media section is automatically assigned to the newly created object.

The picture is slightly more complicated when multiple locations exist for the parent node's encapsulated object. Cross-publishing is usually used only for non-container objects such as articles and documentation pages or product sheets, since adding a location does not include a node's subtree. In these cases, the section ID inheritance follows the main node of the object referenced by the parent node.

Let us illustrate this by looking at a news article cross-published in the restricted "Partner" sub-branch and the publicly available "Company" sub-branch.

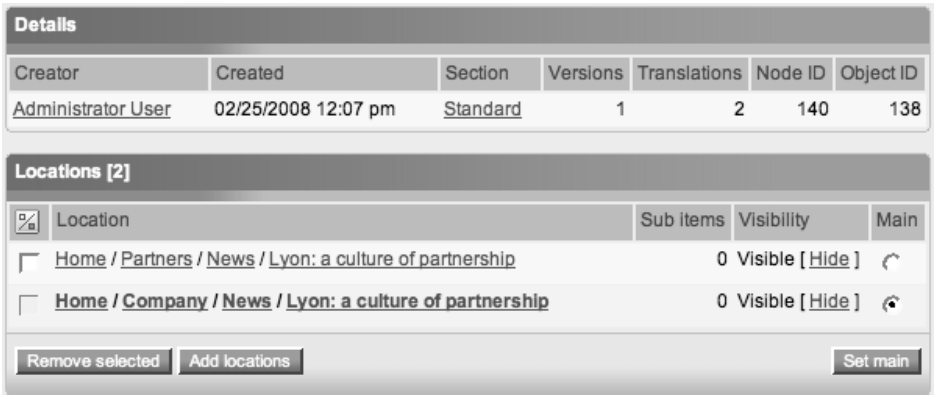


Figure 4.4. Cross-published article, Standard section

While the "Partner/News" folder belongs to the Restricted section, the article belongs to the same Standard section as the "Company/News" folder:

Figure 4.5. Cross-published article, secondary location

If a comment is added to the article at the secondary location, the comment will inherit the section ID for the Standard section from the article itself, not from "Partner/News" folder object that is in the Restricted section. If the "set main" feature is used to change the main node of the article, the section ID of the affected object will be updated. You may have to clear the content cache to see the effect on the subtree.

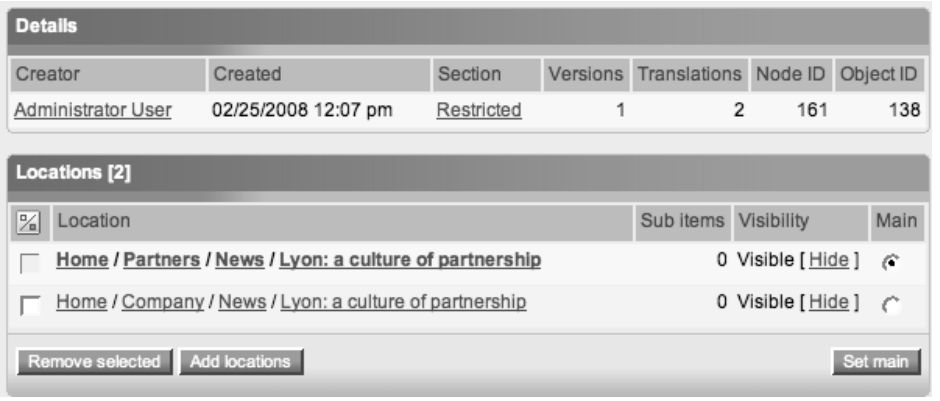


Figure 4.6. Cross-published article, secondary location changed to main

Through the **Sections** interface (described later) you may assign a section directly to a node. The effect of this is that the section ID of the object referenced by the target node will be updated. Moreover, the effect propagates all the way down the subtree of that node. Remember, however, that if any of the nodes in the subtree are secondary locations, their section assignments will remain unchanged. This is because they follow the section assignments of their main locations.

4.3. Section management

We now move on to practical section management, explaining the relevant interfaces and concrete procedures. This is an illustrated walk-through that will show you how to create, assign, edit and remove sections.

4.3.1. Managing sections at the object or node level

We have previously seen that the section ID is assigned to (and stored within) individual objects. Because of this, it is common to say that sections are implemented at the object level. But, we have also revealed that newly created objects inherit the section ID from the parent node when published. Also, section assignments propagate down a node's subtree. Therefore, section are managed at the node level, while implemented at the object level.

4.3.2. Viewing section assignments

There are three windows that display the name of the section to which an object belongs. These are the **Details**, **Sub items** (with the detailed list type toggled) and **Section** windows. Looking at the ID or name of the section assigned to an object should not be confused

with viewing the section itself through the **Section view** interface (as described in Section 4.3.7, “Viewing a section”).

4.3.2.1. Section name in Details window

The **Details** window is shown in the main area of the Administration Interface when viewing some content (and when the corresponding switch has been selected). This displays information about the selected node and the object that it encapsulates:

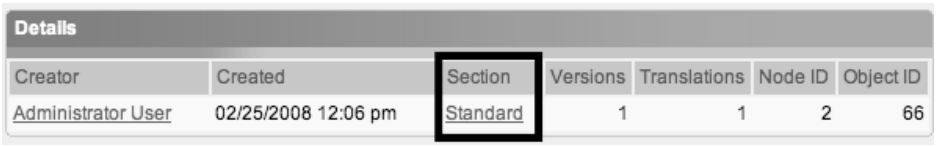


Figure 4.7. Details window for "Home" frontpage

The entry in the third column shows the section name. Clicking it brings up the **Section view** interface.

4.3.2.2. Section name in Sub items window

The **Sub items** window is shown at the bottom of the main area of the Administration Interface when viewing some content. When the detailed list type is selected (in the top right of the window), the sixth column shows the section name for each of the children of the object being viewed. Clicking one of the names brings up the **Section view** interface.

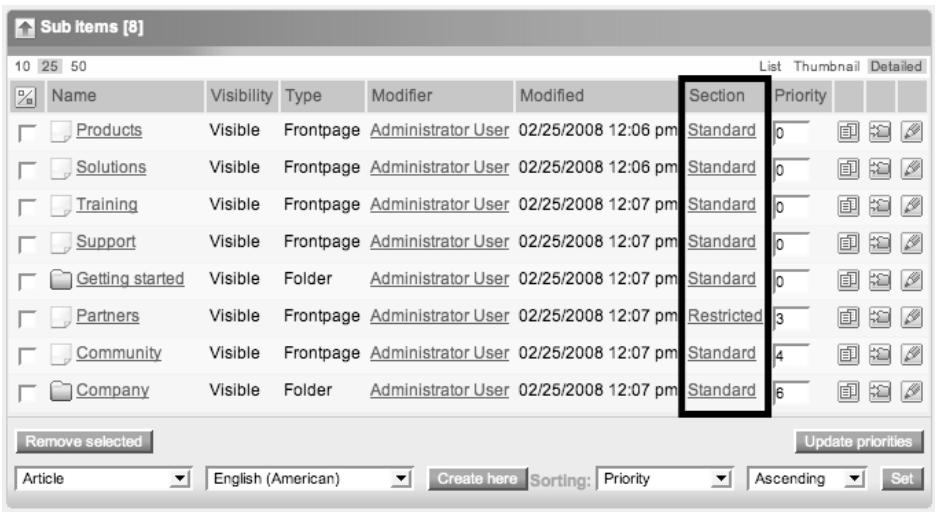


Figure 4.8. Sub items window for "Home" frontpage

4.3.2.3. Section window shown in edit mode

The **Section** window is shown in the bottom left area of the **Object Edit Interface**. This displays the name of the currently assigned section, and lets you change it. The number in square brackets in the title bar shows how many available sections you can choose from.

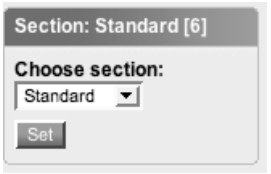


Figure 4.9. Section window for "Home" frontpage

4.3.3. Sections interface

The **Sections** interface is your most important tool for managing sections. From here you can bring up the **Section editing** interface and the **Section view** interface. You can open the **Sections** interface by clicking on the **Sections** link in the left menu of the **Setup** tab. This interface is shown below:

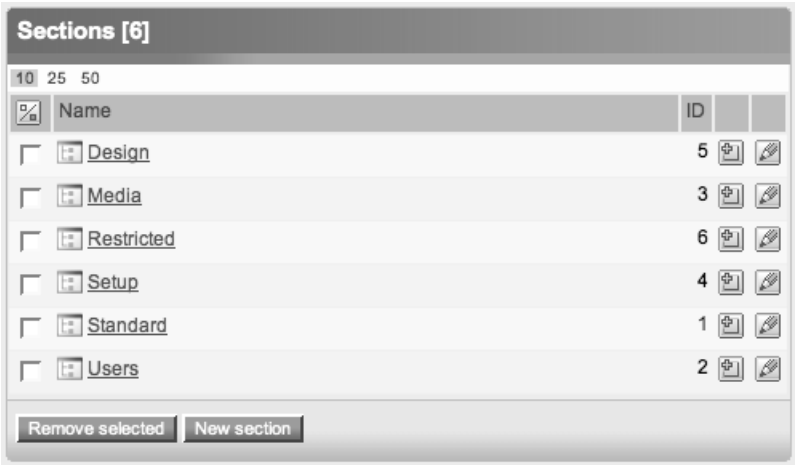


Figure 4.10. Sections interface

The **Sections** interface has a header bar showing how many unique sections there are; a main area listing the name and ID of these sections; and a toolbar at the bottom. If you have many sections, you can toggle the number of sections that are listed per page by clicking the numbers "10", "25" and "50" in the top left corner below the header bar. Section management tasks such as creating, editing, assigning and deleting are carried

out by using the buttons on the toolbar and the right side of the listing, along with the checkboxes to the left of the section listing. These tasks are explained next.

4.3.4. Creating a new section

To create a new section, all you have to do is to provide a name and a navigation part, as explained in the procedure below:

Tip

The navigation and search functionality is disabled in edit mode for a section.

Warning

The system does not require section names to be unique. However, we strongly recommend that you use unique section names. When selecting a section to assign from a dropdown list (such as in the **Section** window in the **Object Edit Interface**) you will only see the section name and be unable to distinguish between sections with duplicate names.

1. Click the **New section** button in the **Sections** interface.

   This will take you to the **Section editing** interface for the new section:

Figure 4.11. Section editing interface

2. Enter a name for the new section in the **Name** input field.
3. Select a navigation part from the dropdown list. You will recognize the names of the Administration Interface main menu tabs within this list. Recall that the navigation part dictates which tab is accessed when working with objects in the section.

4. Click the **OK** button to create the section or the **Cancel** button to abort the procedure.

The newly created section will show up in the section listing, which is alphabetically sorted by default. It will automatically be assigned the next unused section ID. As seen in the screenshot below, our new section is titled "My new section" and has an ID of "7".

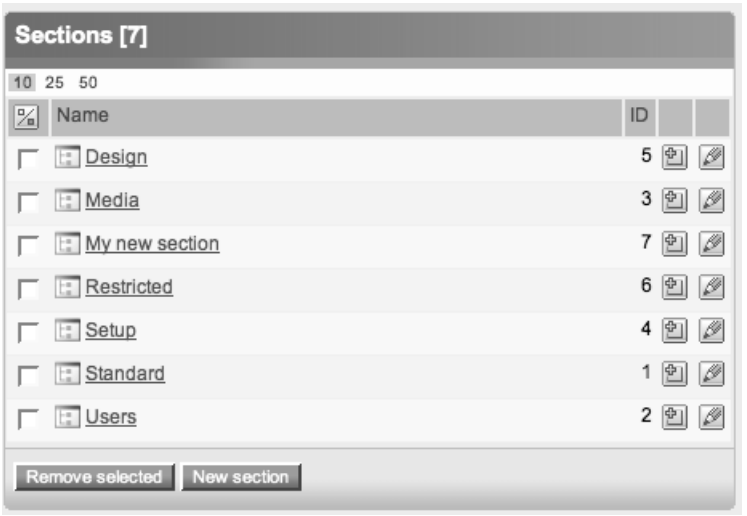


Figure 4.12. Sections interface after creating a new section

4.3.5. Assigning a section

To assign a section to a subtree, follow the procedure below. Note that you cannot assign a section directly when creating or editing the section itself.

1. Open the **Sections** interface.

2. Click the **Assign** button to the right of the desired section.

3. This will open the **Browse** interface within the navigation part you selected for the section, as shown in Figure 4.13, “Assign section - select a node” below. Navigate using the available tabs, the tree menu and the content list.

4. Use the checkboxes to select a node to which to assign the section. You can select multiple nodes. Then, click the **Select** button.

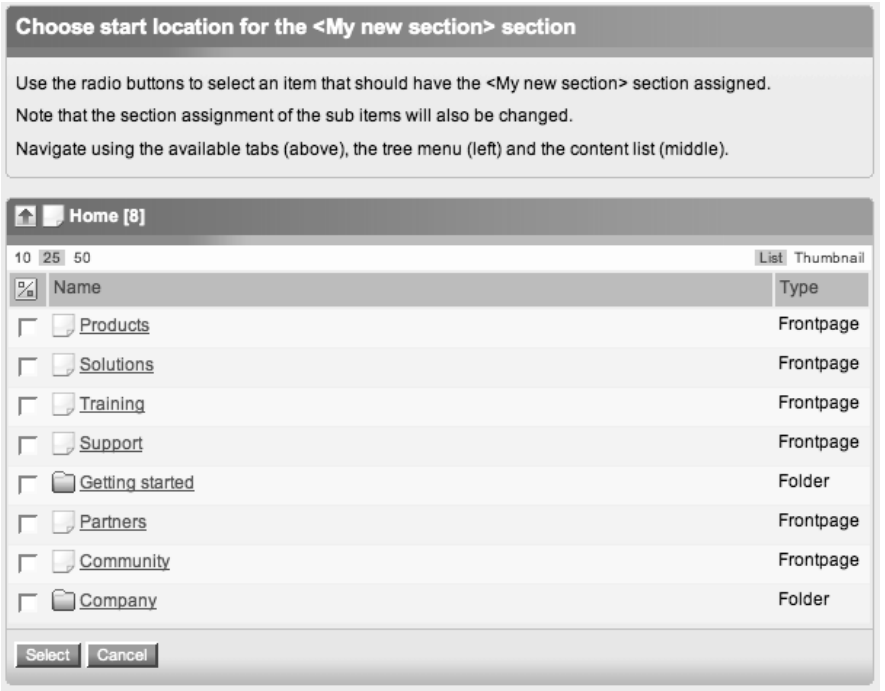


Figure 4.13. Assign section - select a node

Use the **Cancel** button to abort the operation.

The assignment will be carried out for the selected node (or more accurately, its encapsulated object) and the corresponding subtree.

Tip

You can also assign a section to a subtree by opening the target object for editing and changing the section in the **Section** window (as shown in Section 4.3.2.3, "Section window shown in edit mode") before publishing a new version. However, if you are not going to make any content changes to the object, it is best to use the **Sections** interface.

4.3.6. Editing a section

The procedure for editing an existing section (its name and navigation part) is very similar to the creation procedure. Simply open the **Sections** interface and click the **Edit** button to the right of the desired section. You can also click the **Edit** button within the **Section view** interface described below. This triggers edit mode with the **Section editing** interface. Update the values as desired and store the changes by clicking the **OK** button. Otherwise, click the **Cancel** button.

4.3.7. Viewing a section

Viewing a section means to view the ID, name and navigation part for a section. This is different than viewing which section is assigned to some content object, as was described in Section 4.3.2, “Viewing section assignments”. To view a section, first open the **Sections** interface. Then, click on the name representing the section you want to view. This will open the **Section view** interface, as shown below.

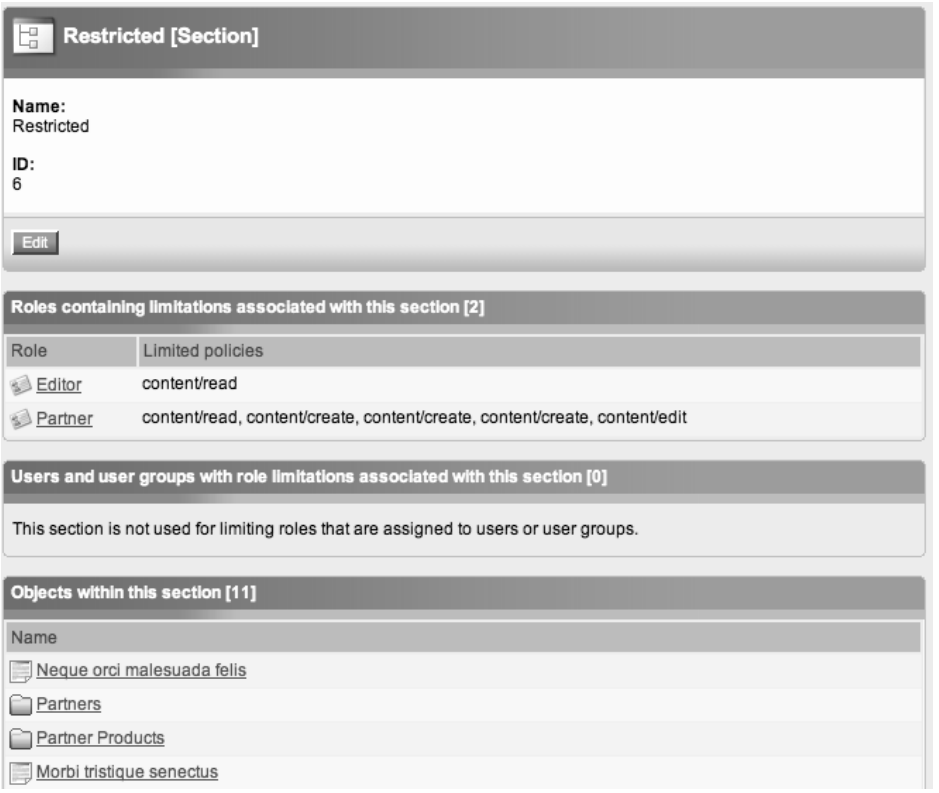


Figure 4.14. Section view interface

The **Section view** interface is a special-purpose interface within the **Setup** tab, but it does not disable other functionality, such as navigating and searching, as the **Section editing** interface does.

Some readers might recognize the general layout from the main area windows of the **Content structure** tab. The top window is a preview of the section itself, showing the name and ID and containing an **Edit** button. Clicking this button will bring up the **Section editing** interface.

The bottom window lists objects within this section, more or less like a simplified version of the **Sub items** window. The items are sorted by publication time, with the newest material on top. Clicking one of the node names will display its content in the main area in the tab corresponding to the navigation part. For example, clicking the "News" node link will display the **Content structure** tab with the "News" folder (in "Home/Partner/News") object in the main area.

The two middle windows show roles containing limitations associated with this section, and users and user groups with role limitations associated with this section. Roles are described in Chapter 5, *User management and the permission system*. These listings basically show how the section is applied for access control.

4.3.8. Removing a section

Removing a section means to delete the ezsection object (see "In Depth: The ezsection object" in Section 4.2.2, “Section properties”) containing the name, ID and navigation part. This means to remove the section from the **Sections** interface list and the dropdown list of the **Section** window (in edit mode). The operation does not remove the content to which this section is assigned, nor any other parts of the system that use the section.

In Depth: Section ID uniqueness

Section ID numbers are not recycled to avoid confusion and exploitable flaws in the permission system. For example, section IDs are sometimes hard-coded in templates. If you were to delete a section, then create a new one with the recycled section ID, the resulting display might not be as you intended.

To remove a section, follow the steps below.

1. Open the **Sections** interface.
2. Mark the checkbox to the left of the section(s) you want to remove.
3. Click the **Remove selected** button to carry out the removal, or the **Cancel** button to abort the operation.

   The system will display an error message and prevent you from carrying out the action if the section is currently being used, as shown below. This ensures that your system is not left in an inconsistent state.

Figure 4.15. Section removal prevented

Clicking the section's name within the dialog box will bring up the **Section view** interface. If you really want to delete the section, you should first take care of emptying the bottom three windows. In other words, you should remove role limitations and assign another valid section to the listed objects.

4.4. Summary

The following concepts and interfaces were covered in this chapter:

- Top-level node: A node that is directly below the root node. Such nodes are the start of a node tree branch, such as the Content or Users branch.
- A section is a virtual collection of nodes that belong together. When combined with other eZ Publish functionality, sections are used to set up template override rules; limit control and access to content; specify content that must be approved before being published; and assign discount rules to products in a Webshop.
- Section ID: an identification number that can be assigned to an object and denotes which section that object belongs to.
- **Sections** interface: your most important tool for managing sections. It is opened by clicking the **Sections** link in the left menu of the **Setup** tab. From here, you can view a list of sections and access interfaces to create, edit and remove sections and assign them to nodes.

Chapter 5. User management and the permission system

This chapter explains user management and the permission system. We cover the layout and interfaces of the Administration Interface **User accounts** tab. We also describe the four parts of access control (users, user groups, roles and policies) and illustrate how these are managed.

This material applies particularly to those who are in charge of day-to-day user management on a site. It is also relevant to all who want to learn more about the eZ Publish permission system.

Recall that:

- The **User accounts** tab in the Administration Interface is associated with the Users branch of the content node tree and the corresponding Users section. You need an editor account with extra permissions (in addition to the default permissions) or an administrator account to access it.
- Sections are used to segment the node tree by creating a virtual collection of nodes that belong together. You can then target, for example, Webshop discounts and access control specifically to that collection.
- An eZ Publish *INI file* is a configuration file that controls the behavior of a specific part of the system. See Section 3.5, “The configuration model”. The main configuration file is `site.ini`. You can view and edit the value of a particular setting through the **Settings view** interface as described in Section 3.5.4, “Settings view interface”.
- A *module* offers an interface for web-based interaction, providing access to eZ Publish core functionality. Each module has a set of *views* that enable you to reach the functions provided by the module. For example, the "edit" view of the "content" module is used for functions such as creating, editing and translating content.

In this chapter, you will find information about the following:

- The building blocks of the permission system
- The layout and usage of the interfaces and windows of the **User accounts** tab
- How to specify and manage roles and policies
- How to create, modify and remove users and user groups.
- How to enable and disable user accounts

- How to create a protected area by applying roles and policies to sections

This material is limited to what you can view and do through the graphical user interfaces. Topics such as creating login handlers or customized user templates, or importing users from external systems are beyond the scope of this book. We encourage interested readers to refer to the technical manual at http://ez.no/doc/ez_publish/technical_manual for more information.

We recommend that the entire chapter is read in the order given. The conceptual material is presented first, while the usage material is grouped in Section 5.7, “Access control usage” and Section 5.8, “Example: Creating a protected area”.

5.1. Permission system

The *permission system* controls access to your site's content and functionality. It includes a set of *user accounts* and *access permissions*.

Without permissions, access to everything on a site is completely denied; it is only by the cumulative assignment of permissions that users are permitted to view content and use site functionality.

5.1.1. Access control components

The two main parts of the permission system can be further split into four components, as illustrated in the following figure:

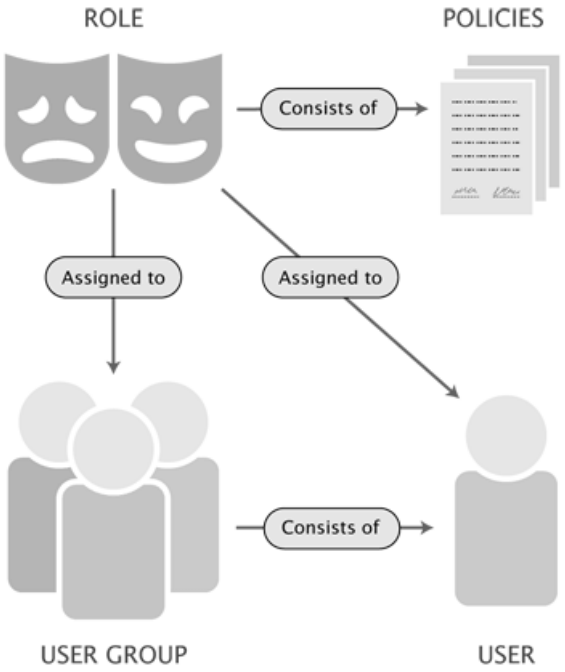


Figure 5.1. Access control components: Users, groups, policies and roles

As shown, the four components are:

- *Policy*: a rule granting access to a part of a site or site functionality (see Section 5.5.1, “Policies”).
- *Role*: a named collection of policies (see Section 5.5.2, “Roles”).
- *User*: a definition of a valid user account on the system (see Section 5.3, “Users”). User accounts can be created via three methods: through the Administration Interface by an existing Administrator user (by default, Editor users do not have access to the **User accounts** tab); through self-registration on the front-end of the site; and imported from external systems.
- *User group*: a named collection of users (see Section 5.4, “User groups”). User groups can contain sub-groups.

In the above illustration, the top two components represent the access permission part, and the bottom two components represent the user account part of the access control mechanism. Moreover, a role consists of policies, whereas a user group consists of users and possibly other groups. Roles can be assigned to user groups or individual users. Note that policies cannot be assigned directly to users or groups.

eZ Publish comes with a set of built-in user groups (see Section 5.4.1, "Predefined groups"), and at least an *Administrator user* and an *Anonymous user*. This ensures that there is a way to log in to perform site management tasks (and add more users and groups to the system), and that unregistered site visitors are permitted to view unrestricted content.

The following sections explain the four access control components in detail, and walk you through the relevant interfaces and special-purpose windows. We start by exploring the **User accounts** tab.

5.2. User accounts tab

The **User accounts** tab enables you to browse and manage nodes within the Users branch of the content node tree. This tab also provides access to the permission system so that you can view and manage roles and policies. For this reason, some windows are unique to this tab (see Section 5.6, "Special-purpose windows for access control").

5.2.1. Access to the User accounts tab

Recall that what site visitors usually see is limited to content located within the content structure, and possibly also parts of the media library. By default, there is no direct access to contents of the Users branch, except via the Administration Interface. User account information is sensitive, and should be accordingly restricted.

5.2.2. Layout of the User accounts tab

In general, the layout of the **User accounts** tab follows the same principles as for the **Content structure** and **Media library** tabs. The six areas are present in their normal positions. The **Search** interface, main menu and path are found horizontally at the top of the page, and the left menu, main area and right area are aligned side-by-side below these elements. The left menu contains content objects belonging to the Users branch. Consider the screenshot of the **User accounts** tab below:

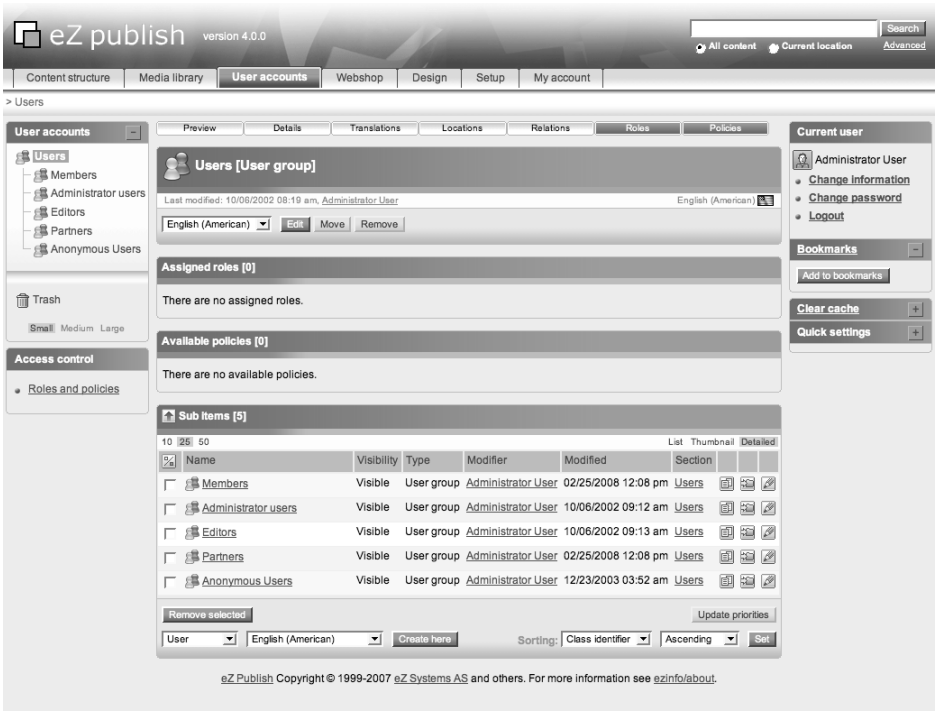


Figure 5.2. User accounts tab

Note the visual differences as compared to the **Content structure** tab (shown in Figure 2.5, “Administration Interface”):

- The title of the left menu is "User accounts", and the top-level node in this tree menu is "Users".
- There are two additional switches at the top of the main area: **Roles** and **Policies**. These are each associated with a special-purpose window (or "panel") described later, and work in the same way as the other switches.
- The **Access control** panel is below the left menu (see Section 5.6.3, “Access control panel and Role list interface”).
- The icons to the left of each node represent either users or user groups .
- The **Class selection** dropdown list in the **Create here** interface at the bottom of the **Sub items** window contains only "User" and "User group" as valid classes. This is also reflected in the "Create here" sub-menu in the context-sensitive pop-up menu.

5.3. Users

The word "user" can refer to three slightly different things:

- Objects of the `User` content class (or any other object with an attribute of the "User account" datatype; see Section 5.3.1.1, “User account datatype”).
- The account and profile that site visitors think of when they log in with their username and password. For the clarity of presentation we will use the terms *user account* and *user profile* for this.
- The person the object or account represents.

When used alone in other parts of this book, the word "user" should be interpreted as the person. In this chapter, "user" most often refers to the content object.

5.3.1. User - the content object

A content object is considered a *User object* if it contains an attribute of the *User account* datatype. All objects that have an attribute of this datatype will automatically become valid users. We explore this datatype and the built-in `User` class for an example user called "Bergfrid Skaara".

5.3.1.1. User account datatype

The "User account" datatype is critical to the permission system. You may think of it as the key property. It supports the validation, storage and retrieval of a username / password combination and an email address. All elements are required. This is a datatype with several elements, similar to the "Image" datatype (although in that case, not all elements are required).

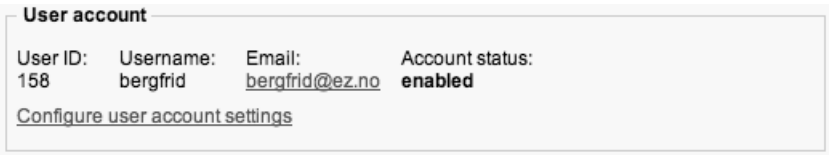


Figure 5.3. User account "bergfrid"

By enabling the **Details** switch, you can view additional information about a user:

Details

Creator	Created	Section	Versions	Translations	Node ID	Object ID
Administrator User	02/25/2008 04:28 pm	Users	1	1	162	158

Figure 5.4. Details for user Bergfrid Skaara

Usernames must be unique and they cannot be edited after a user account is created. By default, email addresses must also be unique, as explained in the in-depth block below.

In Depth: Unique email addresses

Each user account is designed to be personal, but this is not possible if multiple users share a common email address. The email address of a user account is used whenever eZ Publish attempts to communicate with the user. This includes sending confirmation emails to self-registered users so that they can activate the account, as well as email notifications (see Section 1.4, “Notifications”) when content has been updated. Information sent to one user, such as for issuing a new password, would be visible to all users that have access to the shared email account, possibly compromising site security.

Consider another example, for a site where users can submit blog posts by email or SMS. With multiple users associated with the same address, eZ Publish has no way of knowing which User object to associate with the blog post. In other words, content can be attributed to the wrong person.

You also run a higher risk of someone creating multiple user accounts for spamming or flaming your forums and blogs. Requiring unique email addresses makes it easier to block such users (see Section 5.9, “Tips and tricks for permission management”).

Therefore, eZ Publish, by default, does not allow different User objects to have the same email address. To lift this limitation, change the value of the "RequireUniqueEmail" setting of the "[UserSettings]" block of `site.ini` to "false". We strongly encourage you to be completely sure before changing the setting, as there are potential negative effects, depending on the functionality that your site provides.

5.3.1.2. The Anonymous user

Recall that the permission system is cumulative, starting at a point of no access. A prerequisite for enforcing access control is that eZ Publish knows which policies to apply.

This is determined by first checking the currently logged in user. But what if some unregistered random visitor is just browsing your pages? Browsing means that you can, at a minimum, view content. To address this situation, eZ Publish has a special-purpose user account, the *Anonymous user* (and corresponding user group).

Each time someone visits your site, they are silently logged in by eZ Publish, which sets the current user to "Anonymous". Then, the permission system can correctly apply the rules specified for this user (or group). If the visitor decides to log in with a personal account, the current user will be changed accordingly. This usually implies that the visitor is granted more rights, for example to submit comments, or to get access to the **Website Toolbar** (if he or she is an editor).

5.3.1.3. User class

The following screenshot shows an object of the `User` class in edit mode:

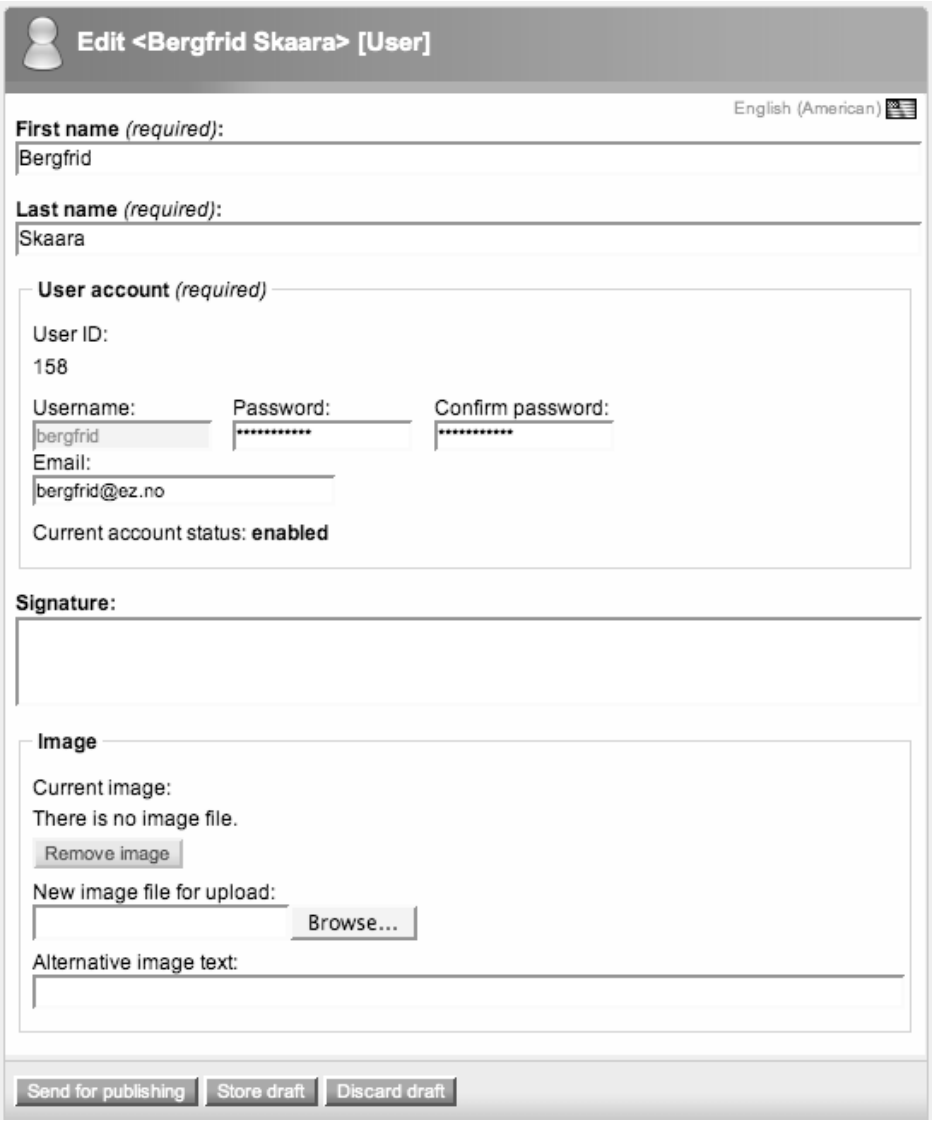


Figure 5.5. User object in edit mode

The "Signature" and "Image" attributes are typically used in forums. Note that the username part of the "User account" attribute is grayed out, since it cannot be modified after the User object has been created.

5.3.2. User - the account and profile

Although the term "user account" technically references a datatype, it is more commonly interpreted by site visitors as the "ability to log in" and "personal space". The latter is

usually referred to as the *user profile* in the front-end context, and as the **My account** tab in the Administration Interface. Note that "user profile" may also denote the information stored in the actual User object, such as first and last name, image and signature. In other words, editing your user profile means to edit the contents of the object holding your user account.

Your personal space provides access to change your password (without editing the profile), manage drafts and notifications, view orders and wish lists (if your site has a webshop), and access pending content waiting for approval (only in the Administration Interface; see Chapter 8) or a future publication date (see Chapter 7).

5.3.2.1. Accessing your personal space

In order to access your personal space, you must log in to either the front-end or the Administration Interface. In the Website Interface, click the **My profile** link in the top right corner of the website. This opens the **My profile** page:

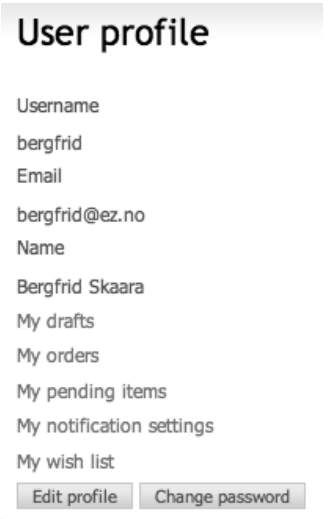


Figure 5.6. My profile page

The **My account** tab of the Administration Interface gives access to all parts of your personal space:

Figure 5.7. My account tab

Note in particular the **User panel** in the right area, where you can click links to bring up interfaces for changing information or your password. This panel is part of the overall Administration Interface layout, and is thus accessible from all tabs. In other words, you do not need to navigate to the **My account** tab in order to simply edit your profile or change your password.

Tip

You can modify the profile and password (although you cannot view passwords) of any user through the **User accounts** tab, provided that your user account has the necessary access permissions.

5.4. User groups

A *user group* is a named collection of users and can contain both individual user objects and other user group objects. It is created, stored and managed as a content object of the `User group` class. As seen in Figure 5.1, “Access control components: Users, groups, policies and roles”, both users and user groups can be associated with a set of policies that determine privileges. You will find more information about this in Section 5.7, “Access control usage”. General rules are usually assigned to groups, whereas specific, dedicated responsibilities are assigned directly to individual users.

5.4.1. Predefined groups

The default groups are pre-configured and usually define the different kinds of users expected on your site. The default groups for sites that use the Website Interface are listed below:

Table 5.1. Default user groups in sites using the Website Interface ("ezwebin" version 1.3)

Group	Description
Anonymous users	Used for the Anonymous user to let unregistered site visitors view unrestricted content.
Members	Commonly used for community and self-registered users.
Partners	Used for selected users that are allowed access to the Restricted section.
Editors	Used for content editors, managers and webmasters. Usually restricted to the Content and Media subtrees.
Administrator users	Used for the site administrator with unlimited access and for advanced content managers who need access to perform site management tasks.

5.5. Roles and policies

We saw previously in Section 5.1.1, “Access control components” that a policy is an access-granting rule, and that a role is a named collection of policies. This section goes into more detail about these concepts. We start by explaining policies, since they are the building blocks of roles.

In contrast to users and user groups, roles and policies are not stored as content objects and nodes. Unlike content objects, there are no versions or translations of a role or policy, nor can you upload them through WebDAV (see Chapter 13) or work with them in OpenDocument Text format (see Chapter 14). In other words, these access control components should be thought of as settings rather than content. However, do not confuse these with INI files (see Section 3.5.1, “What is an INI file?”), which contain other types of settings.

Roles and policies are stored in the database, and the only way you can work with them is through the **User accounts** tab.

5.5.1. Policies

A policy is a single rule that grants access to specific or all functionality of a module. You can set the following for each policy:

- Module: the highest level for which a policy can grant access. Examples include the "user" and "content" modules (see Section 1.3, “Modules and views”).
- Function: available functions depend on the module, and access can be granted to one or all functions of a module. If you want to give access to a subset (but not all) of the functions in a module, you have to create multiple policies, with one policy per function.

  Function examples include "create" and "edit" for the "content" module.
- Limitation: various levels of granularity to which the policy should apply, as described next.

Most of the modules and functions have intuitive, descriptive names, but be sure to consult your site administrator if you are unsure about what a module or function does. You can also use the *References* section in the http://ez.no/doc/ez_publish/technical_manual to find out more about a particular module. In most cases you will be dealing with read and edit permissions for the "content" module.

5.5.1.1. Function limitations in policies

In the context of access control, modules and functions are the entities to which policies apply; a *limitation* (or more precisely a "module function limitation"), restricts the circumstances under which a policy should be applied. For example, a policy might grant access to create content, but a limitation would specify that only blog posts and comments can be created. Another example is a policy that grants access to read content, but only within the Standard section (representing the Content branch).

The following list shows the types of limitations that are available.

- Use the *Class limitation* to limit a policy to objects of certain types, such as articles or blog posts.
- Use the *Language limitation* to limit a policy to object versions in specific languages, such as French translations only. This only applies to multilingual sites.
- Use the *Node limitation* to limit a policy to a specific node. For example, you might want to enable Editor users to edit the site's frontpage, but not the contents it pulls from other areas of the site. Or, you might want to provide read access to a specific node within a restricted area.

- Use the *Owner limitation* to limit a policy to objects that are owned by the user who is logged in. For example, community members might be permitted to only edit their own objects (such as posts in a forum or article comments).

- Use the *Parent class limitation* to limit a policy based on the type of the object encapsulated by the parent node. For example, a user might be permitted to comment on blog and forum posts, but not articles.

- Use the *Section limitation* to limit a policy to objects that are assigned to certain sections. This is used to set up protected areas and generally keep site management off the front-end pages.

- Use the *Siteaccess limitation* to limit a policy to a specific site interface. For example, you might grant a group of users access to log in to the Administration Interface.

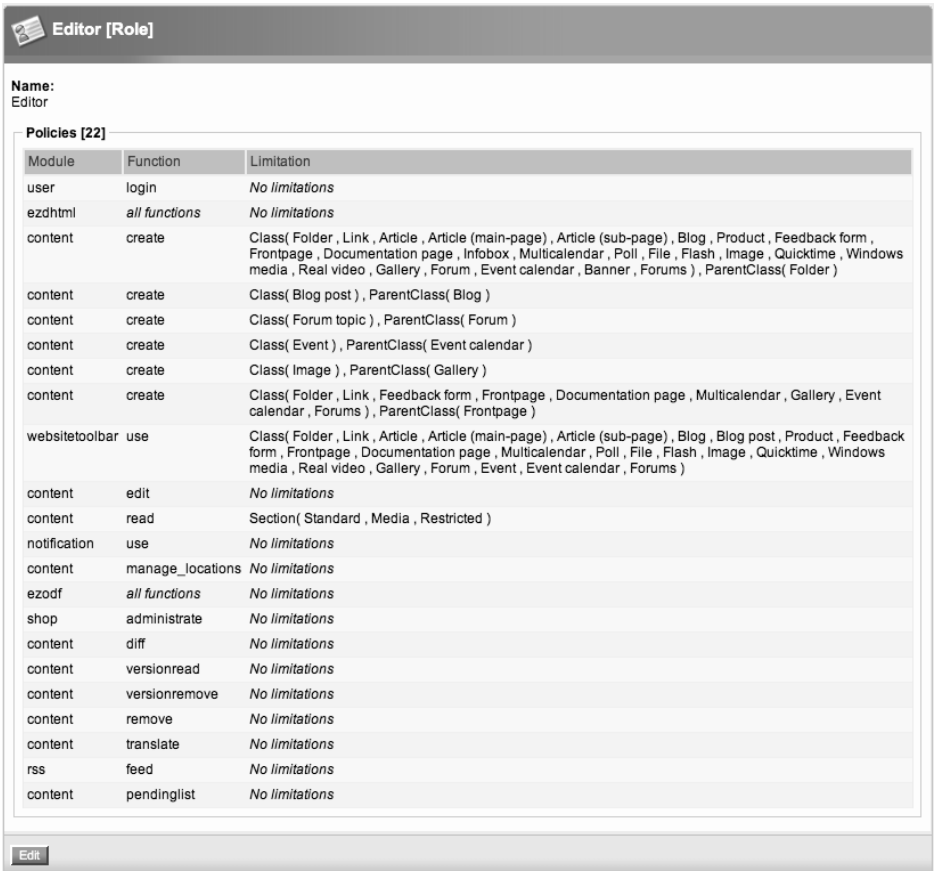
- Use the *Status limitation* to limit a policy to a certain version status (such as "Published" or "Archived").

- Use the *Subtree limitation* to limit a policy to a certain part of the content node tree. For example, a policy might allow content to be created, but only under the "Training" and "Support" nodes. This is typically used to segment editorial responsibility, and to limit areas in which public, user-contributed content is accepted.

  This limitation has some similarities to the Section limitation, but also some important differences. There might be multiple subtrees belonging to the same section, and a subtree might contain several sections.

5.5.1.2. Example: policies of the default Editor role

For illustrative purpose, let us explore some of the policies that are part of the Editor role. The built-in roles are listed in Table 5.2, “Built-in roles for sites that use the Website Interface”. Note that these rules depend on your installation; your Editor users might have other policies in their role(s). The screenshot below shows the **Role management** interface (described later) for the Editor role:



Editor [Role]

Name:
Editor

Policies [22]

Module	Function	Limitation
user	login	*No limitations*
ezdhtml	*all functions*	*No limitations*
content	create	Class(Folder , Link , Article , Article (main-page) , Article (sub-page) , Blog , Product , Feedback form , Frontpage , Documentation page , Infobox , Multicalendar , Poll , File , Flash , Image , Quicktime , Windows media , Real video , Gallery , Forum , Event calendar , Banner , Forums) , ParentClass(Folder)
content	create	Class(Blog post) , ParentClass(Blog)
content	create	Class(Forum topic) , ParentClass(Forum)
content	create	Class(Event) , ParentClass(Event calendar)
content	create	Class(Image) , ParentClass(Gallery)
content	create	Class(Folder , Link , Feedback form , Frontpage , Documentation page , Multicalendar , Gallery , Event calendar , Forums) , ParentClass(Frontpage)
websitetoolbar	use	Class(Folder , Link , Article , Article (main-page) , Article (sub-page) , Blog , Blog post , Product , Feedback form , Frontpage , Documentation page , Multicalendar , Poll , File , Flash , Image , Quicktime , Windows media , Real video , Gallery , Forum , Event , Event calendar , Forums)
content	edit	*No limitations*
content	read	Section(Standard , Media , Restricted)
notification	use	*No limitations*
content	manage_locations	*No limitations*
ezodf	*all functions*	*No limitations*
shop	administrate	*No limitations*
content	diff	*No limitations*
content	versionread	*No limitations*
content	versionremove	*No limitations*
content	remove	*No limitations*
content	translate	*No limitations*
rss	feed	*No limitations*
content	pendinglist	*No limitations*

Edit

Figure 5.8. Policies in the Editor role

The screenshot shows 22 policies, for eight different modules ("user", "ezdhtml", "content", "websitetoolbar", "notification", "ezodf", "shop" and "rss"). Excluding content policies, the policies in this role give users of this group access to log in, use the Online Editor and the **Website Toolbar**, use the notification feature, use the eZ Open Document Format extension's import / export features, perform webshop administration and access RSS feeds (see Section 9.2.2.3, “RSS feeds”). All of these are specified without limitations (except for the policy for the **Website Toolbar**, which has class limitations), and grant access to what you would expect for a content editor.

There are also six policies for the "create" function of the "content" module, all with Class and Parent class limitations. The first has a limitation of which types of objects can be created beneath folders. The next four specify rules for specific content relationships, such as for creating blog posts within a blog, events within a calendar, and so on. The last policy restricts what you can create beneath a frontpage.

The remaining policies grant access to: edit content; view content within the Standard, Media and Restricted sections; manage locations; manage versions; view pending items (for delayed publishing and content to be approved); and remove and translate content.

Compare this to what Anonymous users are permitted to do, as shown in the screenshot below. Anonymous users can only view content of the Standard section and some of the contents of the media library. They can also view RSS feeds (see Section 9.2.2.3, “RSS feeds”). No editing is permitted.

Figure 5.9. Policies in the Anonymous role

Recall from Section 5.3.1.2, “The Anonymous user” that site visitors are silently logged in by eZ Publish as Anonymous users, until they log in with a user account. In the policies shown above, we see that Anonymous users are granted access (via the "login" function or the "user" module) to some of the front-end siteaccesses, but not to the Administration Interface.

5.5.2. Roles

A role is a container and grouping tool for policies. Remember that only roles, not policies, can be assigned to users and user groups. Once you have set up a role, you can re-use it and assign it as many times as necessary. Because of this you can, for example, build an access hierarchy with cumulative rights.

It is also possible to assign a role with limitations. These limitations are much simpler than the policy limitations, with only the Section and Subtree limitations available. For

example, you can create a role that applies only to the content structure and media library parts of the Administration Interface for advanced content managers that do not need access to user management and site configuration.

The table below shows the default roles for websites that use the Website Interface:

Table 5.2. Built-in roles for sites that use the Website Interface

Role	Description	Assigned to
Anonymous	Shown in Figure 5.9, "Policies in the Anonymous role" above.	Anonymous users, members, partners
Member	Granting community access (for example, to create forum topics and replies and comments and edit these within the Standard section).	Members, partners, editors
Partner	Granting access to restricted areas.	Partners
Editor	Shown in Figure 5.8, "Policies in the Editor role".	Editors with subtree limitations (Content and Media branches)
Administrator	Granting unlimited access	Administrators

Role assignment and other permission management tasks are covered in Section 5.7.2, "Managing roles and policies".

5.6. Special-purpose windows for access control

Here, we outline the layout and functionality of the windows and interfaces specific to the **User accounts** tab. This includes the **Assigned roles** window, the **Available policies** window, the **Access control** panel, the **Role management** interface, and the **Role editing** interface. Section 5.7, "Access control usage" will go through step-by-step instructions on how to perform management tasks using these windows and interfaces.

5.6.1. Assigned roles window

The **Assigned roles** window in the main area of the **User accounts** tab can be toggled on and off using the **Roles** switch. The screenshot below shows this window for the Administrator user:

Figure 5.10. Assigned roles window - Administrator user

The title bar shows the number of roles that are assigned to the current object, which is a user or user group. In this case, there is only one role. In the listing, the first column shows the name of the role as a clickable link. Accessing this will take you to a page displaying more information about that particular role (see Section 5.6.4, "Role management interface").

The second column reveals whether the role has been assigned with limitations, and if so, it lists the limitations. The third column contains the **Edit** buttons; click one of these buttons to access the **Role editing** interface (see Section 5.6.5, "Role editing interface") for a particular role.

5.6.2. Available policies window

The **Available policies** window can be toggled on and off using the **Policies** switch. The screenshot below shows this window for the Administrator user:

Figure 5.11. Available policies window - Administrator user

The title bar shows the number of available policies. The listing shows what role, module, function and limitation the policy regulates. These were described in Section 5.5.1, "Policies".

5.6.3. Access control panel and Role list interface

The **Access control** panel at the bottom of the left menu contains the **Roles and policies** link:

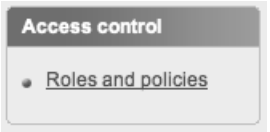


Figure 5.12. Access control panel

Clicking this link brings you to the **Role list** interface containing the **Roles** window:

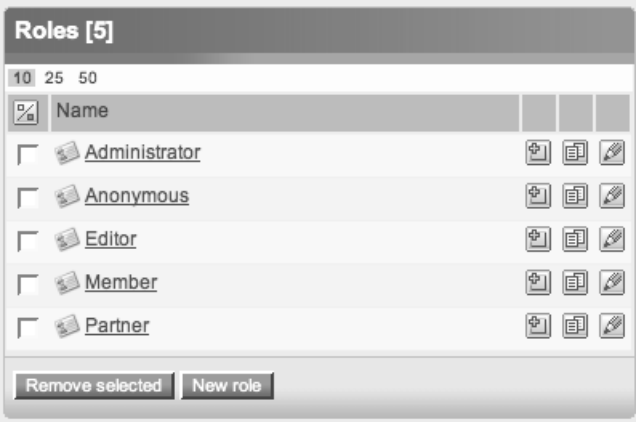


Figure 5.13. Role list interface

The title bar displays the total number of roles, which in this case is five. The listing contains an entry for each of the roles. You can change the number of roles listed per page by clicking "10", "25" or "50" directly beneath the title bar.

Clicking on the name of the role brings up the **Role management** interface (as explained below). To the right of each name are buttons to assign, copy and edit the roles. The bottom toolbar contains the **Remove selected** button, which is used together with the checkboxes to the left of the role names to delete roles. The **New role** button initiates the process to create a new role.

5.6.4. Role management interface

The following screenshot shows the **Role management** interface for the Anonymous role. The interface is accessed by clicking the name of a role in the **Role list** interface or the **Assigned roles** window.

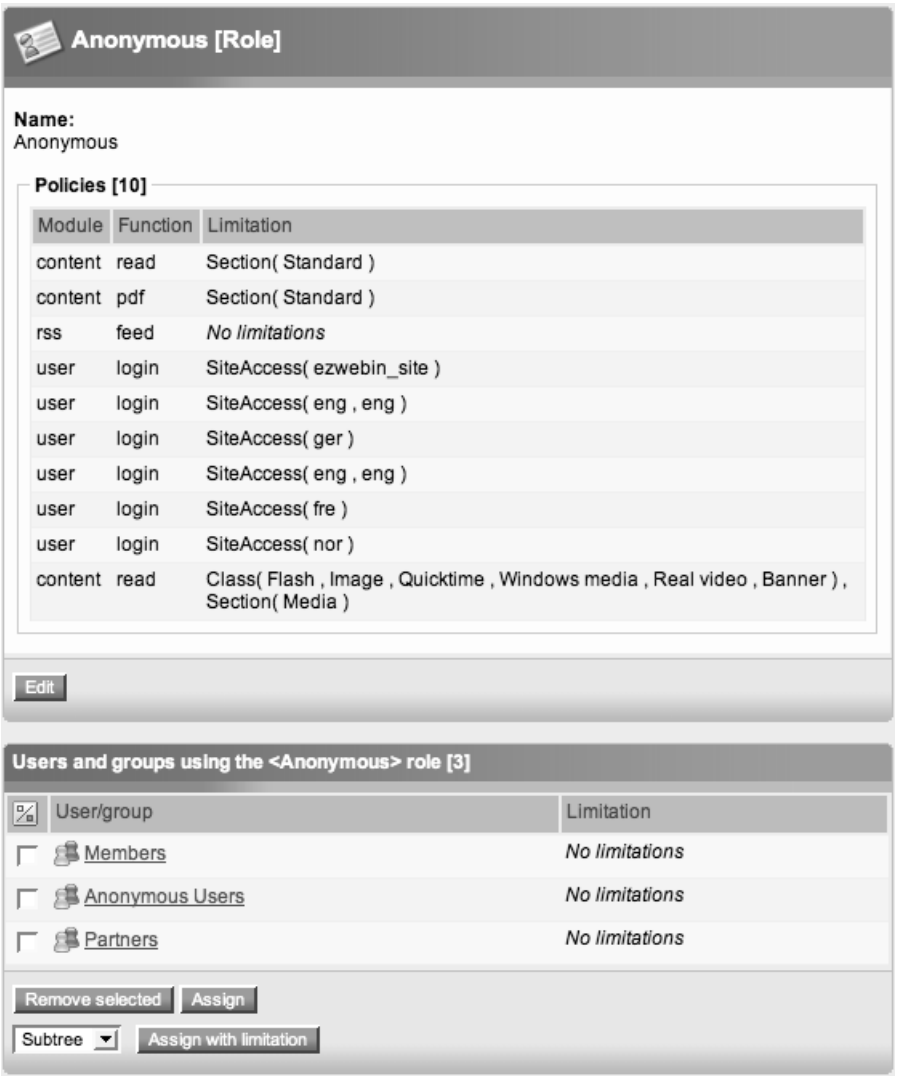


Figure 5.14. Role management interface - Anonymous role

The top window is a simplified version of the **Preview** window normally shown for content objects. The title bar shows the name of the role, which is "Anonymous" in this case. The body of the top window shows a name and a listing of all of the policies for the role. The bottom bar contains only the **Edit** button; clicking this button will open the **Role editing** interface described below.

The bottom window in the **Role management** interface shows the users or user groups to which a role has been assigned, along with the applied limitations. It also has a collection of buttons, checkboxes, and a dropdown list to assign the role to other users and user

groups or to remove existing assignments. This is described further in Section 5.7.2.1, “Creating, deleting, copying and editing roles”.

5.6.5. Role editing interface

The **Role editing** interface is a special-purpose editing interface for roles. It is accessed by clicking the **Edit** button in the **Role management** interface, the **Role list** interface or the **Assigned roles** window.

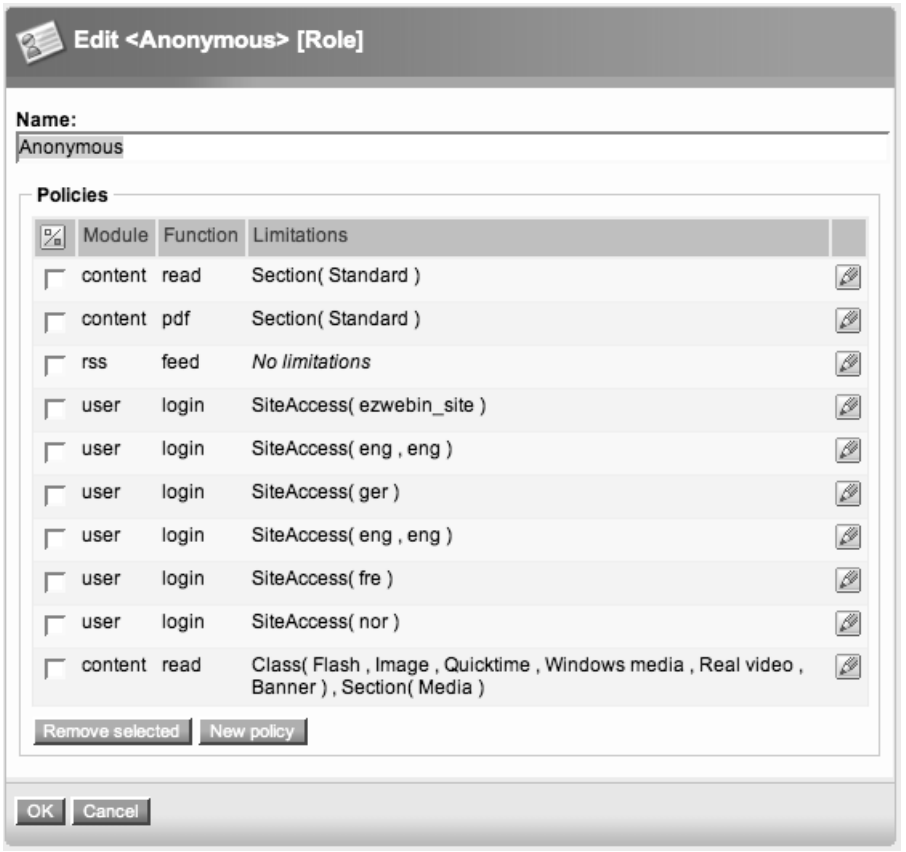


Figure 5.15. Role editing interface - Anonymous role

As seen above, the left menu is replaced by the **Role information** window, showing the name and ID of the role. The ID is only used internally.

The main area of the **Role editing** interface lets you modify the name of the role, and provides access to modify the policies within it. You can edit a particular policy by clicking the **Edit** button for it. You can remove one or more policies by marking the corresponding checkboxes, then clicking the **Remove selected** button. You can add a

new policy by clicking the **New policy** button. Changes made to any of the policies in the role or to the name of the role are saved when you click the **OK** button at the bottom. The changes can be discarded by clicking the **Cancel** button.

5.7. Access control usage

In this section, you will find information on practical procedures involving users, user groups, policies and roles in the **User accounts** tab.

Tip

If you modify the permissions for a user and want it to take effect immediately, clear the *User info cache*, as described in Section 3.4.3, "Cache usage". This is particularly useful if, for example, an editor contacts because she lacks some specific permission for the task she is currently working on.

5.7.1. Managing users and user groups

Managing users and user groups is done similarly to how you would manage other content objects. The same principles apply when creating, editing, viewing, copying, deleting, moving, translating and cross-publishing.

Tip

Pay attention to the section (see Section 4.2, "What is a section?") of the main node when cross-publishing users, in order to ensure that you do not unintentionally expose user accounts on the public site. Since cross-publishing does not affect the subtree, this does not apply to user groups.

If the main location of a user is within the Standard section (see Section 4.2.3, "Built-in sections"), site visitors would see usernames, email addresses and more in the same way as attributes in articles or blog posts. What is exposed would depend on the location selected, as well as the templates and what they are set up to display. The following screenshot shows an exposed user account accessed through the sitemap. The information shown is not critical, but you would not want everybody (especially spammers) to be able to see the email addresses of all your users!

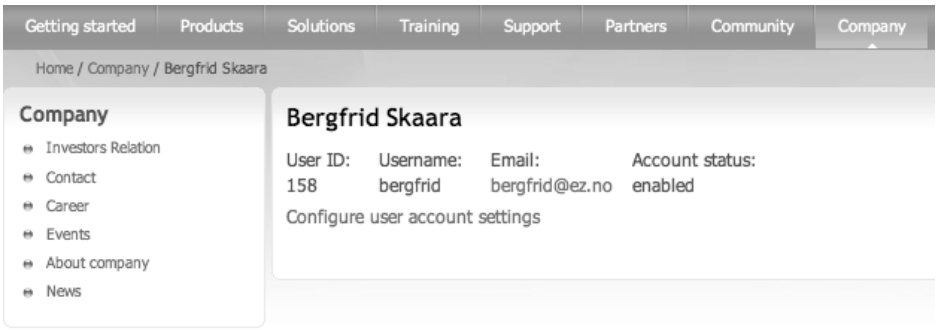


Figure 5.16. Exposed user account information

5.7.1.1. Adding a user group

The following procedure describes how to add a new user group, essentially creating a new User group object.

1. Access the **User accounts** tab.

2. Navigate to the location where you wish to add the new group. You can do this through the left menu and **Sub items** window, or alternatively by using the **Bookmarks** window.

3. Select "User group" from the dropdown list in the **Sub items** window and click the **Create here** button.

   Alternatively, bring up the pop-up menu and select "User group" under the "Create here" sub-menu.

4. This will open the **Object Edit Interface**, as shown below. If you have a multilingual site, you will first be presented with the reduced **Language selection** interface.

Figure 5.17. Edit mode for a new user group

5. Enter a name (required) and description (optional) for the user group. Mark the **Website Toolbar access** checkbox to give access to the toolbar for users within this group.

6. Click the **Send for publishing** button. The new user group will appear at the selected location.

   Alternatively, store the user group as a draft by clicking the **Store draft** button, or abort editing by clicking the **Cancel** button.

After a user group is published (and is thus visible in the node tree), you can add users and / or user groups beneath it in the node tree, and assign permissions to it.

The rest of this section focuses on user management, but many of the procedures are very similar for user groups. This is noted where relevant.

5.7.1.2. Adding users

The following procedure describes how to add a new user, essentially creating a new User object. It is similar to the previous procedure for creating user groups.

Tip

Some sites have a lot of self-registered users (who use the **Register** link at the top-right of the front-end site), especially sites with a lot of user-contributed material in forums and other community website features. Such users are located according to the "DefaultUserPlacement" setting in *site.ini*, which you can check in the **Settings view** interface. This typically references the Node ID for the the Member user group. The procedure given here is targeted at advanced content managers who must create accounts on behalf of others, such as for employees.

1. Access the **User accounts** tab and navigate to the location where you wish to add the new user. The location is usually a user group (so that the User object is created beneath it).

2. Select "User" from the dropdown list in the **Sub items** window, then click the **Create here** button.

   Alternatively, bring up the pop-up menu and select "User" under the "Create here" sub-menu.

3. This will open the **Object Edit Interface** for a new user account.

4. Enter the required information: the first and last name, email, username and password for the user. Optionally, provide a signature and image. The newly created user can update this information (except for the username) in his or her user profile later (as described in Section 5.3.2.1, “Accessing your personal space”).

5. Click the **Send for publishing** button. The new user will appear at the selected location.

   Alternatively, store the User object as a draft by clicking the **Save draft** button, or abort editing by clicking the **Cancel** button.

In Depth: The UserClassGroupID setting

"UserClassGroupID" is a configuration setting within the "[UserSettings]" block of *site.ini* that specifies the classes you can create instances of within the Users section in the Administration Interface. It regulates the options listed in the **Create here** dropdown list within the **User accounts** tab. It does not reference specific class IDs, but rather a class group ID (see Section 3.2, “Viewing and editing content classes”). By default, this setting references the ID for the Users class group. Since this class group contains the `User` and `User group` classes, these are the classes that appear in that **Create here** dropdown list. This

setting only needs to be updated if developers add custom classes to your site for user management outside the default Users class group.

5.7.1.3. Editing users

Editing a User object can be done either by the user himself / herself through the user profile when logged in (see Section 5.3.2.1, “Accessing your personal space”), or by a user with sufficient permissions in the Administration Interface.

1. Access the **User accounts** tab and navigate to the user you want to edit (or its parent node).

2. Open the **Object Edit Interface** by clicking the **Edit** button at the bottom of the **Preview** window.

   Alternatively, click the **Edit** button to the right of the user listed in the **Sub items** window of its parent node.

   You can also access the pop-up menu and initiate the editing process there.

3. Update the desired information, then click the **Send for publishing** button.

Tip

You can also edit user groups by locating them and entering edit mode, similar to what was described above for users.

5.7.1.3.1. Moving users to other user groups

There are several reasons why you might want to move a user (or group) to another user group. The most obvious one is that you want to update a user's permissions, to grant a higher or lower access level. You might also have selected the wrong location when you initially created the user. Or, you might need to place a self-registered user in the appropriate group.

1. Access the **User accounts** tab and navigate to the user you want to edit (or its parent node).

2. Click the **Move** button at the bottom of the **Preview** window.

3. This will bring up a **Browse** interface for choosing a new location for the user:

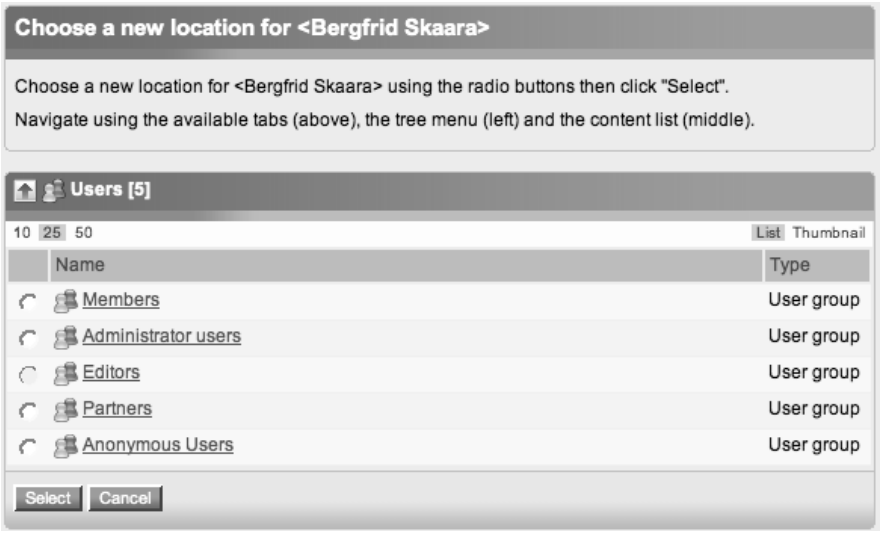


Figure 5.18. Browse interface for moving a user

Navigate using the available tabs, the tree menu and the content list. Then, mark the radio button for the desired location and click the **Select** button.

You can abort the operation by clicking the **Cancel** button.

Tip

To display more or less options at a time at each navigation level, toggle the pagination indicators in the top left corner of the **Browse** interface.

5.7.1.3.2. Self-registered users

Self-registered users are those who have clicked the **Sign up** button in the **Login** interface or the **Register** link in the top right of a front-end siteaccess, and filled in and submitted the necessary user information. Such user accounts are disabled until users have clicked the link within the confirmation email sent by eZ Publish.

After clicking the confirmation link, self-registered users are activated in the Members group in sites that use the Website Interface. To gain more privileges, an advanced content manager or webmaster must move the account into another group, or assign the necessary roles or policies to the individual account.

5.7.1.4. Inheritance of privileges

The full set of access permissions for a specific user is determined by inheritance. Recall that a role can be assigned both to user groups and to individual users. The following list shows the sources that provide the permissions to a user.

- Policies in roles directly assigned to the user.
- Policies in roles assigned to the user group to which the user belongs.
- Policies in roles assigned to user groups that are ancestors to the parent user group (if there are multiple levels of user groups).

In other words, a user inherits and accumulates privileges from all user groups located between itself and the Users top-level node (this can also include the Users top-level node if there are roles assigned to it). It is cumulative in the sense that policies for the same module and module functions are combined. For example, the Anonymous user can read content of the Standard section and part of the Media section via the policies in the Anonymous role. Members are assigned both the Anonymous and Member roles and can read content in the Standard, Media and Restricted sections. You may say that the least restrictive policy wins: if the policies in your assigned roles dictate that you can edit all content and also edit only your own content, the final outcome is that you edit all content.

Table 5.2, “Built-in roles for sites that use the Website Interface” listed which of the built-in roles are assigned to which of the built-in user groups. Since the Administrator role grants unlimited access to all modules, assigning other roles to users and user groups who are already assigned the Administrator role is redundant. This same logic explains why Editor users are only assigned the Editor role, and not the less-permissive roles. The policies within the Editor role include the policies in those roles, but in a less restrictive / more permissive way. For example, a policy in the Editor role gives access to the "login" function in the "user" module without limitations, while the policies in the Anonymous role for the same function and module are siteaccess-specific.

One application of this inheritance hierarchy is to mimic security clearance levels. For example, a general "Employee" group for level "1" would be assigned the "Level 1 clearance" role. A "Grade 2" group would be created below the "Employee" group in the node tree and assigned the "Level 2 clearance" role. You could then add User objects for employees into the correct user groups. Continuing this pattern for four levels, a "Grade 4" employee would then have the access policies in roles for levels "1" through "4".

5.7.1.5. Disabling and enabling users

Associated with each user account is a status indicating whether the account is enabled or disabled. *Disabled accounts* cannot be used to log in to eZ Publish.

This feature is usually used for one of two purposes. First, newly created, self-registered accounts must normally be enabled (or activated) before they can be used, as was explained in Section 5.7.1.3.2, “Self-registered users”.

The second purpose is to provide an alternative to irreversibly removing users (described in Section 5.7.1.6, “Removing users”). By disabling the account, the user can no longer log in, but all relations the account has to existing content are preserved. For example, articles written by that user would still retain the correct author information.

The procedure for enabling and disabling an account is given below.

1. Access the **User accounts** tab and navigate to the user that you want to enable or disable. Make sure it is displayed in the main area and that the **Preview** window is fully expanded (click the **Preview** switch if it is not).

2. In the **Preview** window, locate and click the **Configure user account settings** link (see Figure 5.3, “User account "bergfrid"”).

   This will bring up the **User settings** interface:

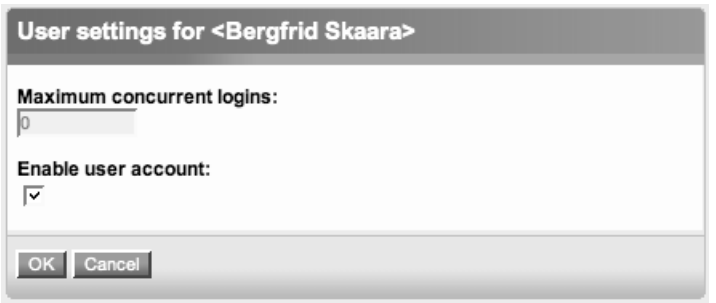


Figure 5.19. User settings interface

3. Mark the **Enable user account** checkbox to enable the account (or unmark it to disable the user account). Then, click the **OK** button.

   (The "Maximum concurrent logins" feature is not active in eZ Publish at the time of publication.)

5.7.1.6. Removing users

Removing a user means to delete the User content object. This will break all references to this user in eZ Publish. All content created or modified by this user will lose their creator / author information. Also, object relations involving this user will be broken. If you have links in content to this user, such a sidebar note that pulls information from a user account to display who is responsible for customer support, it will not display properly when the object is removed.

Because of the need to preserve content relationships, we strongly recommend that you do not delete users. Instead, you should disable the user account as described above. The only time that it is safe to delete a user is if you are absolutely sure that there are no references to the object. You can partially verify this by enabling the **Relations** window when viewing the User object:

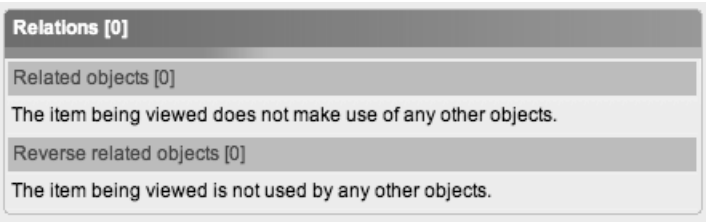


Figure 5.20. Relations window

As indicated above, there are no relations involving this particular user. However, this does not reveal whether this user has created any content, such as a forum post or article comment.

Warning

Never delete the Administrator user or the Anonymous user.

If you are sure that you want to remove a User object, follow the steps below.

1. Access the **User accounts** tab and navigate to the user you want to remove (or its parent node).

2. Click the **Remove** button at the bottom of the **Preview** window.

   Alternatively, bring up the pop-up menu and select "Remove", or mark the checkbox beside the user in the **Sub items**, then click the **Remove selected** button.

3. This will open a confirmation window:

Figure 5.21. Confirm removal of user object

Click the **OK** button to complete the delete operation. Otherwise, click the **Cancel** button.

Tip

User groups can be removed in the same way. However, you should never remove a group that has sub items, since that would remove the user accounts as well.

Tip

Removing users and user groups do not affect the roles that were applied to them.

5.7.2. Managing roles and policies

Recall that roles and policies are not content objects and are edited differently. Here, we will explain step-by-step procedures on how to manage them.

5.7.2.1. Creating, deleting, copying and editing roles

The **Role list** interface (see Figure 5.13, "Role list interface") provides access to role management operations. To access it, click the **Roles and policies** link in the **Access control** panel in the **User accounts** tab. The table below summarizes how to perform the various role management operations in the **Role list** interface.

Table 5.3. Role operations

Operation	Description
Create	Click the **New role** button. Enter a name. Add policies as needed. Then, click the **OK** button.
Delete	Mark the checkbox to the left of the role name(s) and click the **Remove selected** button. You will not be asked for confirmation, and the removal is carried out silently.
Assign	Click the **Assign** button to the right of the role name. See Section 5.7.2.2, "Assigning a role" for an illustrated procedure.
Copy	Click the **Copy** button to the right of the role name. You will be presented with the **Role editing** interface where you can update the name and policies in the duplicate role. Click the **OK** button when done.

Operation	Description
Edit	Click the **Edit** button to the right of the role name. Alternatively, when viewing a User or User group object that has been assigned the role, click the **Edit** button in the **Assigned roles** window (see Section 5.6.1, "Assigned roles window"). Or, click the **Edit** button at the bottom of the **Role management** interface (when viewing a specific role). You will be presented with the **Role editing** interface where you can update the name and policies. Click the **OK** button when done.

5.7.2.2. Assigning a role

Role assignment means to make a connection between access rules and user accounts (see Figure 5.1, "Access control components: Users, groups, policies and roles"). Assigning roles can be initiated from the **Role management** interface (for a particular role) or the **Role list** interface (for any role). Clicking the **Assign** button takes you to the **Browse** interface, where you select one user or user group to which to assign the role, then click the **Select** button:

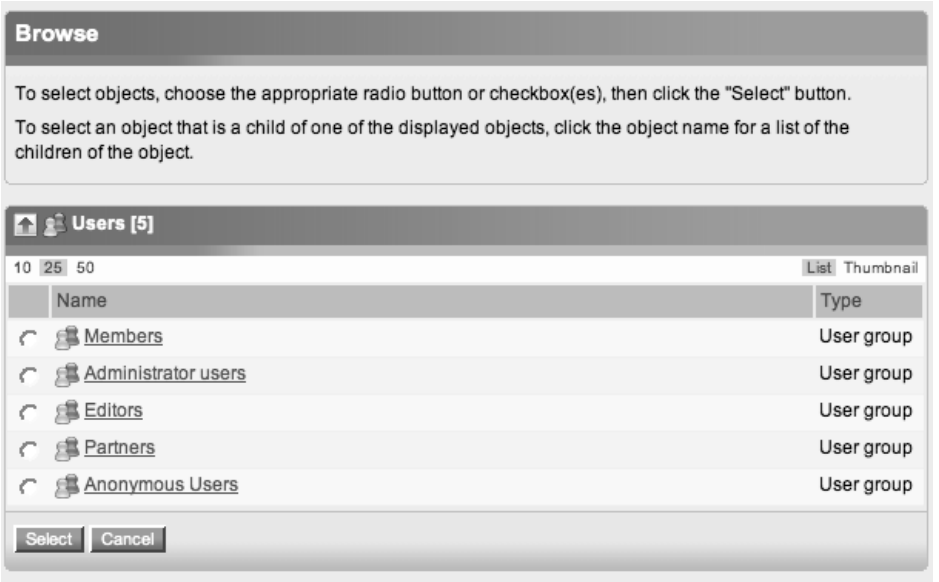


Figure 5.22. Browse interface for assigning a role

If you want to assign a role to multiple users or user groups, you must repeat the operation, as you cannot select more than one target for the assignment at once.

You can also assign a role with Subtree or Section limitations, similar to the limitations available for individual policies. This can only be done from the **Role management** interface. First, select the desired limitation in the dropdown list, then click the **Assign with limitation** button. For Subtree limitations, this will open the **Browse** interface, where nodes from the content structure will be shown. For Section limitations, the page will simply reload with a special-purpose **Select Section** window:

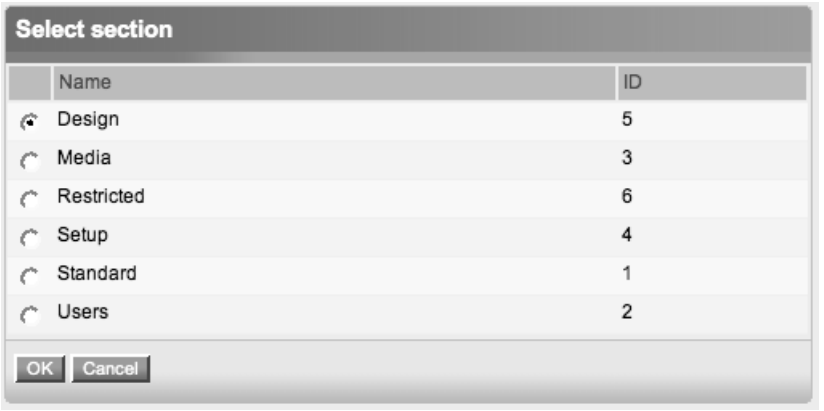


Figure 5.23. Select section window

This list will contain all available sections. Mark one of the radio buttons, then click the **OK** button. Each section is listed with both its name and ID.

After selecting a subtree or section, the assign process continues with the **Browse** interface where you select a user or user group to which to assign the role, as explained previously.

5.7.2.3. Working with policies

Recall that policies cannot be assigned directly to a user or user group. You have to first add the policy to a role, and then assign the role. Because of this, there is no separate "create / delete / copy / edit / assign" functionality for policies as there is for roles (see Table 5.3, “Role operations”). To make policy changes, you have to first edit the role that contains the policy, by bringing up the **Role editing** interface as described in Section 5.7.2.1, “Creating, deleting, copying and editing roles”. Note that the **Available policies** window (displayed when viewing a user or user group) does not provide access to any management tasks.

To remove one or more policies from a role, mark the corresponding checkboxes in the **Role editing** interface, then click the **Remove selected** button.

To create a new policy in a role, follow the steps below.

1. Click the **New policy** button in the **Role editing** interface.

This will open the **Policy** wizard:

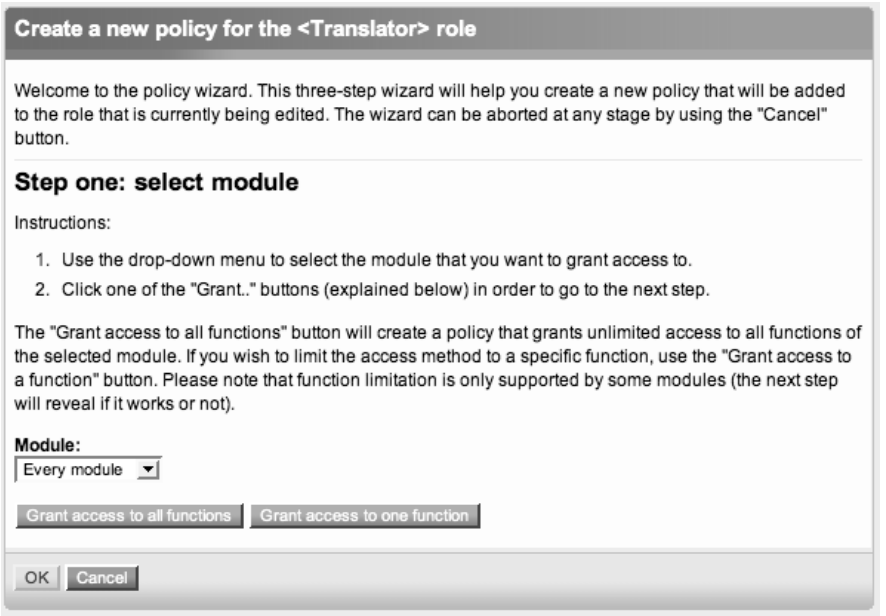


Figure 5.24. Policy wizard step one

The wizard contains three steps with instructions to help you create a new policy.

2. Select a module, such as "content", from the dropdown list.

3. Grant access to all or just one function of that module by clicking the corresponding button.

   If you click the **Grant access to all functions** button, the policy will be added to the role and the procedure is complete.

   If you click the **Grant access to one function** button, the wizard continues to step two. (Note that the numbering of steps in the wizard does not correspond to the numbering of steps in this procedure.)

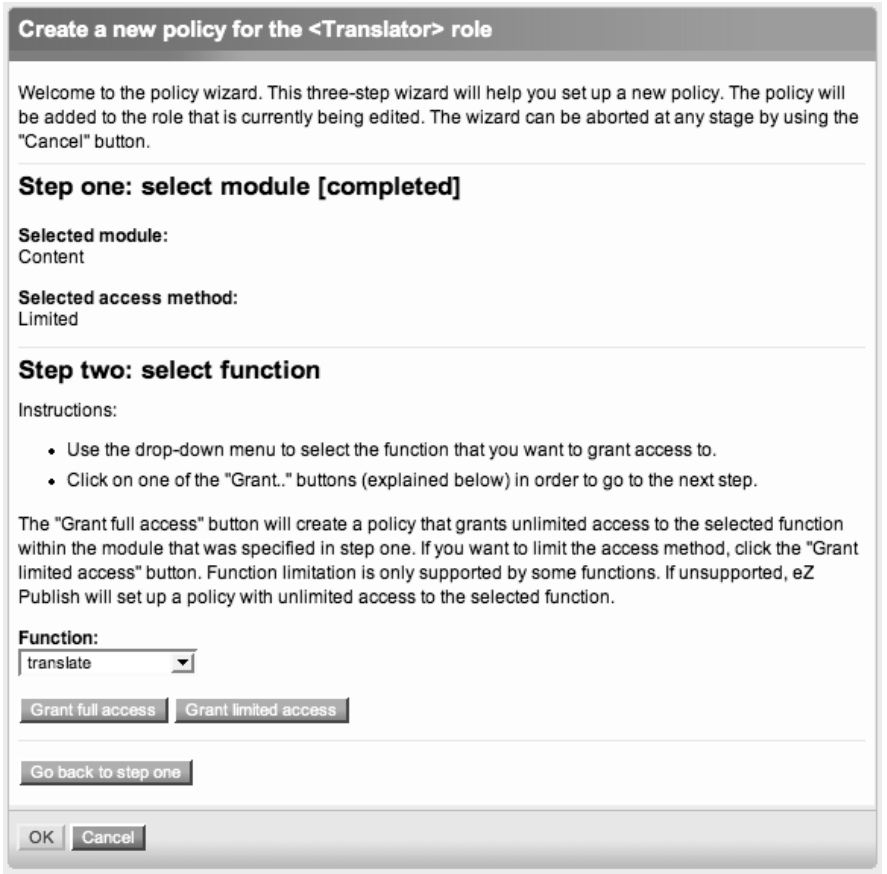


Figure 5.25. Policy wizard step two

4. Select a function, such as "translate", from the dropdown list.

   Grant full or limited access to the function by clicking the corresponding button. Some functions do not support limitations, such as when you grant access to use notifications. If this is the case, or if you grant full access to the function, the policy will be added to the role for the given module and function (and the procedure is complete).

   If you selected to grant limited access to the function, continue with step three of the wizard, where you select the function limitations:

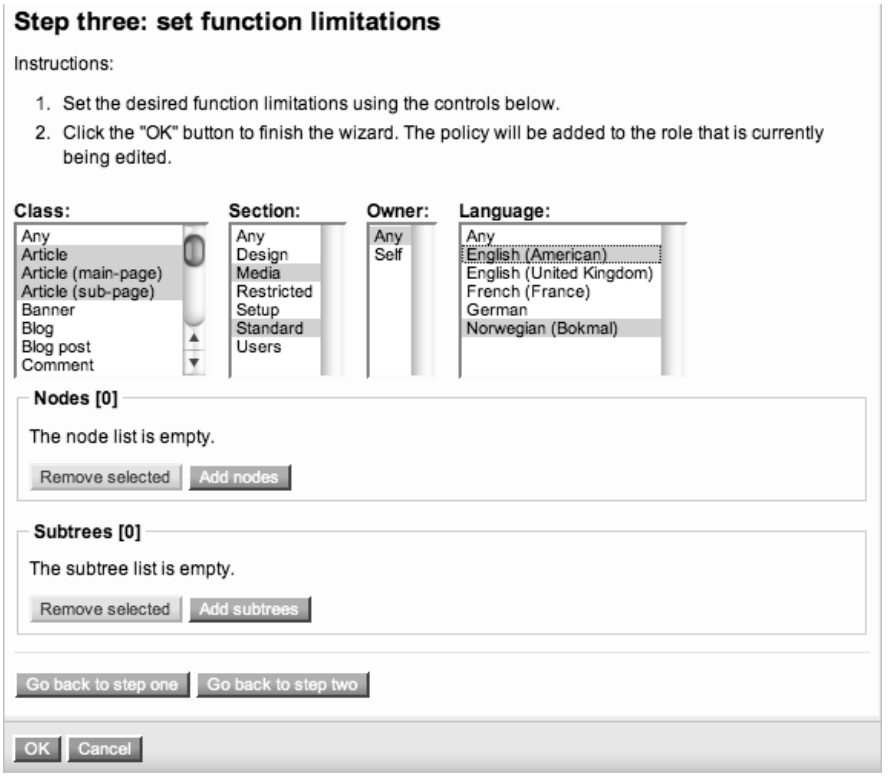


Figure 5.26. Policy wizard step three

5. Set the desired function limitations (as described in Section 5.5.1.1, “Function limitations in policies”) using the appropriate controls. For example, you could limit the policy to apply to articles within the Standard sections in English and Norwegian (excluding French and German).

   The function limitations vary, depending on the module and function previously selected. Keep in mind that limitations are applied together, making the resulting function limitation more permissive for each limitation you select within the policy.

6. Click the **OK** button to finish the wizard. The policy will be added to the role that is currently being edited.

Tip

You can select multiple items in each function limitation list view by holding down the Ctrl button on your keyboard while selecting the subsequent items. Alternatively, hold the Shift button on your keyboard and click again to select a range of items. To add multiple nodes of subtrees to their respective limitations, simply click the appropriate **Add** button after adding a previous node or subtree.

Tip

You can abort the **Policy** wizard at any time by clicking the **Cancel** button. If you need to go back to one of the previous steps, there is a button for each completed step at the bottom of the wizard.

To edit a policy of a role, access the **Role editing** interface and click the **Edit** button for the desired policy. This opens the **Policy editing** interface, which contains controls for updating limitations available for this module and function. For example, for the "user/login" policy, it is only applicable to set a siteaccess limitation as shown below. When finished editing, click the **OK** button.

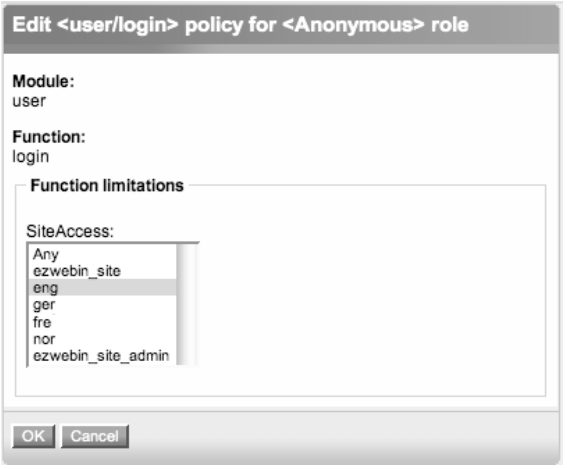


Figure 5.27. Edit "user/login" policy

5.8. Example: Creating a protected area

In this example, we will combine what you learned about sections in Chapter 4 with the access control concepts from this chapter to create a protected area of the site.

Recall that by default, the **Content structure** tab is associated with the Content top-level node and the Standard section. The out-of-the-box eZ Publish behavior is that everything published here constitutes a visible part of your site. This means that everybody, both logged in and anonymous users, is able to view all content located here. For example, if you create a new article in the **Content structure** tab, it will be visible to everybody.

A *protected area* is a part of your site that is only accessible to a certain group of users. For example, a company might have some employee-only material or annual reports available only to trusted clients.

In short, to make a protected area, you must segment the node tree by creating a new section and assigning it to some node(s). Then, you must use the permission system to grant access to the secret section for a specific group of users. The procedure below goes through the specific steps to create an example protected area:

1. Access the Administration Interface and navigate to the **Content structure** tab.

2. Create a folder called "Secret documents" somewhere under the Content top-level node. You can verify that it is in fact displayed on your public site at this time.

3. Bring up the **Sections** interface by navigating to the **Setup** tab and clicking the **Sections** link on the left menu.

4. Create a new section called "Secret section" by clicking the **New section** button and providing the name and navigation part in the **Section editing** interface.

5. Assign the newly created section to the "Secret documents" folder that you created in step 2. To do so, click the **Assign** button to the right of your new section in the **Sections** interface. Browse to the correct folder, mark its radio button, then click the **Select** button.

   You can now verify that the "Secret documents" folder is inaccessible from the public front-end site by bringing up your site in another browser window and attempting to access the folder.

6. In the Administration Interface, navigate to the **User accounts** tab and create a new user group called "Secret users", as described in Section 5.7.1.1, “Adding a user group”.

7. Create a new user within the "Secret users" group, as described in Section 5.7.1.2, “Adding users”.

8. Create a new role called "Secret role" as described in Table 5.3, “Role operations”.

9. Add a new policy to the role as described in Section 5.7.2.3, “Working with policies”.

   Grant access to the "read" function in the "content" module. During the final step where you add limitations, make sure that the policy grants access to the "Secret section" section.

10. Assign both the Anonymous role and the newly created role to the "Secret users" group as described in Section 5.7.2.2, “Assigning a role”.

    Bring up your site and attempt to log in with the user that was created inside the "Secret users" group. The user should be able to access the "Secret documents" part of the site, while anonymous users will still be blocked.

5.9. Tips and tricks for permission management

This section lists some practical solutions to problems related to the permission system. In addition, we provide some tips on best practices and on how to avoid common mistakes. Some of these tips were included within this chapter, but are also summarized here.

- Never delete the Administrator user or Anonymous user.

- Use the disable functionality (see Section 5.7.1.5, “Disabling and enabling users”) to prevent obsolete users from logging in. Do not use remove the User content object, which in most cases will leave your system in an inconsistent state.

- If a newly created user cannot log in and / or access content on the site, check that it is located in the correct user group and is assigned the correct role(s).

- A registered user must have access to "login" function of the "user" module in order to actually log in to the site. This can be limited to a specific siteaccess, or granted site-wide. Remember to update your self-registered users, and to add this access when you create new roles and users in the Administration Interface.

- Editors and content managers should have access to both the "create" and "edit" functions of the "content" module. In addition, they should be able to access the "ezdhtml" module in order to use the Online Editor. Access to the "admin" siteaccess is required for back-end editing. Likewise, access to the **Website Toolbar** must be provided to front-end editors.

Also, editors and content managers will need access to other functions and modules, depending on the tasks for which they are responsible. For example, using the show / hide feature (see Chapter 6, *Node visibility*) requires access to the "hide" function of the "content" module.

- If you have trouble with someone spamming or flaming your site (making unwanted posts to your community forums or blogs, for example) simply disable the user. This action will prevent the user from logging in, and will also keep the email address registered so that the offending person cannot create a new account with the same email address.

5.10. Summary

The permission system consists of two parts: a set of user accounts (with associated user profile) and access permissions. Both can be set up in the Administration Interface.

The permission system is further split into four access control components: user, user groups, roles, and policies. A policy is a rule granting access to a module, possibly limited to a specific function of it and with additional limitations. A role is a named collection of policies that can be assigned to an individual user account or a group of users.

Users and user groups are represented by content objects, while roles and policies are not. Editing User and User group objects is done in the same way as editing content objects. Editing roles and policies has its own special-purpose interfaces.

A protected area is a part of your site that is only accessible to a certain group of users. You can set this up by using sections and the permission system.

Chapter 6. Node visibility

By default, all nodes in the Content branch are visible and thus accessible on the front-end of a site. You can prevent the display of published objects by altering their *node visibility*. This chapter describes this concept and explains how to work with the *hide / show feature*.

The first part of the chapter explains the related concepts and applies to everybody who is interested in learning about node visibility. The second part walks you through how to hide and reveal content, applying only to content managers working in the Administration Interface.

Recall that:

- A content object is wrapped inside (or encapsulated) and structured using a *content node* (or "node" for short). Nodes are organized in a node tree.
- "Draft", "Published" and "Archived" are statuses indicating where an object is in its life cycle. A draft contains content that has been saved but not yet published.
- To "unpublish" content means to take content off the public site. This is usually achieved by deleting the content object (clicking the **Remove** button), or hiding the node (changing its visibility status).

In this chapter, you will find information about the following:

- Node visibility and the visibility statuses
- Requirements of the hide / show feature
- How to reveal and hide content

Locations and the structure of the content node tree play an important role in the hide / show feature. Review Chapter 1, *Introduction to Content Management Concepts* if you need to review node tree connectivity. Content managers who are going to be using this feature should study this entire chapter. If you only want an introduction to the concepts, read Section 6.1, “The hide / show feature” and Section 6.2, “Visibility statuses”.

6.1. The hide / show feature

We have previously seen in Section 5.8, “Example: Creating a protected area” that it is possible to create restricted areas by applying access permissions to sections. The *hide / show feature* also denies access, but is applied directly to nodes, without requiring section or permission management. Usually, it is used to prevent the display of content for all

users on a siteaccess, and is therefore most suitable for temporary unavailability of content (for example, if you are working on a new area of a site).

Node visibility (or "visibility" for short) refers to whether some published content can be viewed, generally on the front-end of a site. The effect of hiding a node is that the node in question, and all of its descendants, disappears from the front-end pages (including the Website Interface) while still being accessible from the Administration Interface. This behavior is regulated by an INI setting, as described in Section 6.4, “Requirements for working with hidden content”.

A visible node is only shown if it is allowed by the permission system. In other words, when the system determines whether or not to show a node, access control comes first, then node visibility.

The following illustration shows a part of a node tree with two main branches. The direct descendant of the left branch has been hidden (by a user), while the node of the right branch is visible.

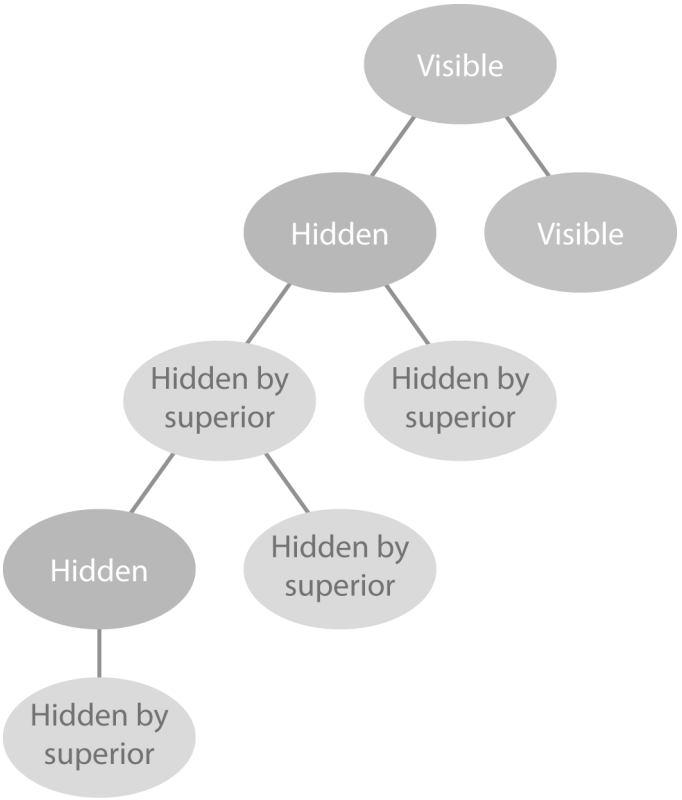


Figure 6.1. Example: node visibility

Suppose that you hide a folder containing all your products. Both the folder itself and the products will not be shown on the front-end of the site. Then you can, for example, update price information for the whole shop before revealing it to site visitors once again.

Tip

Remember that visibility is applied to nodes, not objects. If you hide one location of an object, its other locations might still be visible. See Section 6.2.1, “Status propagation”.

Next, we explain the three different visibility statuses, and show you the places in the Administration Interface where you can view (and in some cases change) this information. The actual procedures for hiding and revealing content are found in Section 6.5, “Usage of the hide / show feature”.

6.2. Visibility statuses

Visibility status refers to whether a node (and content located below it) should be displayed and if not, whether it has been explicitly or implicitly hidden. A node has one of the following three visibility statuses: "visible", "hidden", or "hidden by superior".

Nodes with the "hidden" or "hidden by superior" status are not displayed on the front-end of a site, although there are different reasons as to why they are not displayed (as explained in the table below). Because the end result is the same in both cases, the terms "hidden", "invisible", "not shown" and "not displayed" are all commonly used to describe nodes with the "hidden" and "hidden by superior" statuses.

The table below summarizes the visibility statuses:

Table 6.1. Overview of visibility statuses

Visibility status	Description
Visible	This is the default status. When the node is accessed, the object encapsulated by this node is displayed.
Hidden	The node is invisible and the referenced object is not displayed. This node was explicitly hidden by a content manager or another user with sufficient permissions. All nodes below it in the subtree are also invisible (with either a "hidden" or "hidden by superior" status).

Visibility status	Description
Hidden by superior	The node is invisible and the referenced object is not displayed. A node with this status has a node with the "hidden" status somewhere above it in the content node tree (this can be a direct parent node or one of its ancestors). In other words, it has been indirectly hidden by the system because of a hide operation further up in the tree.

In Depth: Visibility flags

The visibility status is actually a combination of two flags: the *Hidden flag* and the *Invisible flag*. The flags are handled at the system level, whenever you hide or reveal nodes in the Administration Interface. A node is flagged as "Hidden" only when it is explicitly hidden. A node is flagged as "Invisible" in two cases: when the node itself is explicitly hidden, and when a superior node gets explicitly hidden.

The table below shows which flags are assigned to nodes that have the different statuses. A hyphen indicates that the flag is not set, while an "x" indicates that the flag is set.

Table 6.2. Visibility status and flag combinations

Visibility status	Hidden flag	Invisible flag	Shown / not shown
Visible	-	-	Shown
Hidden	x	x	Not shown
Hidden by superior	-	x	Not shown

A node cannot have a "Hidden" flag without also having an "Invisible" flag.

6.2.1. Status propagation

When you alter the visibility of a node, this has an effect down the entire subtree. This is illustrated in Section 6.5, “Usage of the hide / show feature” later. For content managers, this means that you can quickly hide or reveal entire subtrees, although you must be conscious of whether nodes have "hidden" or "hidden by superior" statuses.

For example, when you hide a node, all of the node's visible children are automatically given the "hidden by superior" status (as shown in Figure 6.1, “Example: node visibility”).

Tip

By going through the example illustrations provided with the usage procedures later in this chapter, you will gain more insight into how visibility statuses and propagation work.

6.3. Viewing visibility information

You can view the visibility of a node in several ways in the Administration Interface. First, and without having to actually access the node, you can check its visibility by looking at the background of the node name in the left tree menu. The background is normally transparent, but turns gray when a node is invisible. However, do not confuse this with the currently selected node, which has white text and a bolder background. In the screenshot below, the "Boxes" folder and its sub-items are hidden.

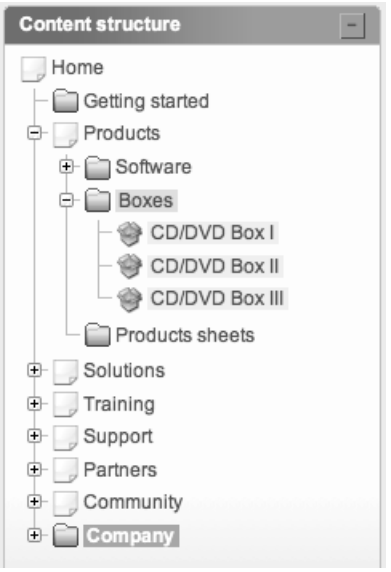


Figure 6.2. Hidden nodes with gray background in tree menu

You can view the visibility status of a node from its hover tooltip in the left menu or **Sub items** window. For example, hover the cursor over the name of a node in the **Sub items** window to show a tooltip with the node's ID and visibility status:

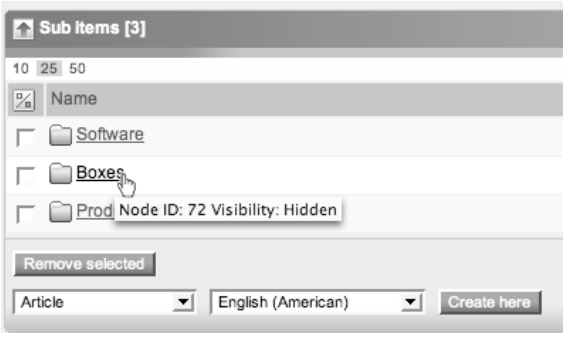


Figure 6.3. Hidden node indicated by tooltip

Alternatively, view the node in the main area and look at the title bar of the **Preview** window. Next to the content class name, you will see "Hidden" or "Hidden by superior" in parentheses, representing those statuses. If the node is visible, there will be nothing to the right of the class name.

Figure 6.4. Hidden node indicated by Preview window title bar

Lastly, you can examine the **Visibility** column of the **Locations** window when viewing a node. Here, you can change the visibility status as well (as explained in Section 6.5.1, “Altering the visibility status”).

Figure 6.5. Hidden node indicated by Locations window

6.4. Requirements for working with hidden content

If a user has the appropriate permissions in a given situation (see Chapter 5) to access visible nodes, he or she can also view hidden nodes if the current siteaccess is configured to show hidden nodes.

The *site.ini* configuration file contains a "ShowHiddenNodes" setting in the "[SiteAccessSettings]" block. Typically, this is set to "false" for all of the *site.ini* siteaccess overrides except for the Administration Interface siteaccess (where it is set to "true"). See Section 3.5.4, “Settings view interface” for information about how to view and edit INI files.

In order to be able to toggle the visibility status of a node, the user's account or user group must have been assigned a role with a policy granting access to "hide" function of the "content" module. Otherwise, you must modify or assign the appropriate policy, as described in Section 5.7.2.3, “Working with policies”.

The Administrator user is automatically configured to access hidden nodes in the Administration Interface. To check whether a particular user or group has the proper permissions, enable the **Policies** switch in the **User accounts** tab when viewing that user or group. Look for an entry in the **Available policies** window that says "content - hide". If it is there, the user is already authorized for this feature, possibly with some limitations. If the user is granted access to all modules, or all functions within the "content" module, he or she is also authorized.

6.5. Usage of the hide / show feature

A user can hide and reveal a node using the Administration Interface, given that the above described requirements are met. As you will see, your most important tools in this case are the **Locations** window and the context-sensitive pop-up menu.

Tip

The terms "reveal", "show" and "unhide" all refer to the same operation on a node.

The effect of hiding or revealing a node differs depending on whether its parent is visible or invisible. Recall that you can hover over a node in the left menu to show its visibility status. Also, the hide / reveal action will propagate down the subtree starting at the target node, and affecting its descendants, as we will demonstrate.

6.5.1. Altering the visibility status

To explicitly hide or reveal a node, locate it in the left tree menu, bring up the pop-up menu, and select the "Hide / unhide" item under the "Advanced" sub-menu. The system will change the visibility status of the target node. (If the node has the "hidden by superior" status, this will give it the "hidden" status.)

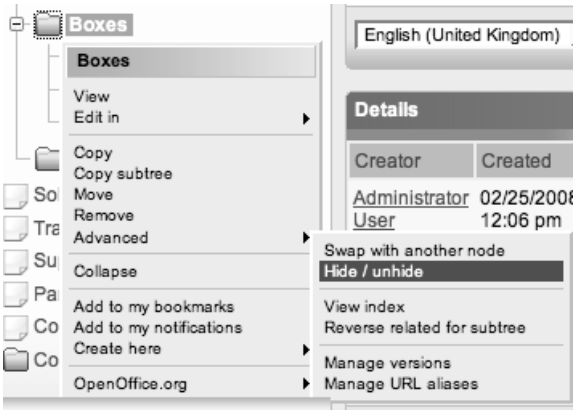


Figure 6.6. Altering a node's visibility status from the pop-up menu

If you cannot use the above approach (if your browser does not have JavaScript enabled or your left menu only displays a subset of content classes that excludes your target node), follow the procedure below:

1. Navigate to your target node and make sure it is displayed in the main area of the Administration Interface. Enable the **Locations** switch if the corresponding window is not shown.

2. Within the **Locations** window (see Figure 6.3, “Hidden node indicated by tooltip”), click the link in square brackets ("Hide" or "Reveal"). The current visibility status is shown in the **Visibility** column.

The following node tree examples are chosen to highlight different scenarios, and do not build on each other in sequence.

6.5.2. Hiding nodes

By default, all nodes are visible. When you hide a node for the first time, the target node (and the nodes below it, if any) become invisible. The following scenarios explain the effect of hiding a node with a visible or an invisible (having a "hidden" or "hidden by superior" status) parent.

6.5.2.1. The effect of hiding a node with a visible parent

The following illustration shows a node tree before and after hiding a visible node.

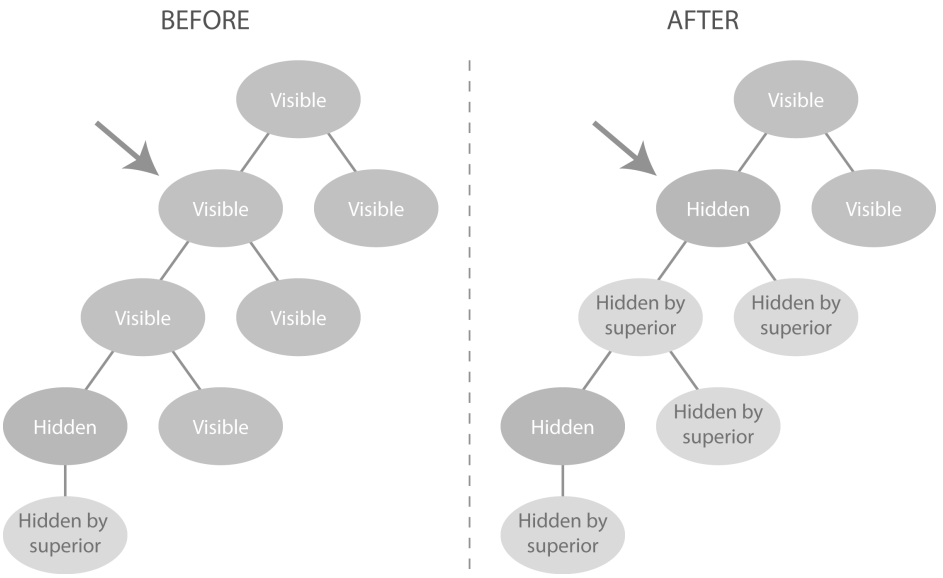


Figure 6.7. Example: hiding a node with a visible parent

In the left side, only one node has a "hidden" status, and its direct descendant has a "hidden by superior" status as a result. When the indicated node is hidden, it receives the "hidden" status and its descendants receive the "hidden by superior" status. However, nodes in this subtree that already have the "hidden" or "hidden by superior" status before the operation preserve their visibility status. In other words, only nodes in the subtree that have a "visible" status are affected.

6.5.2.2. The effect of hiding a node with an invisible parent

This second scenario is slightly different from the one above, in that the target node is already invisible (and has a "hidden by superior" status):

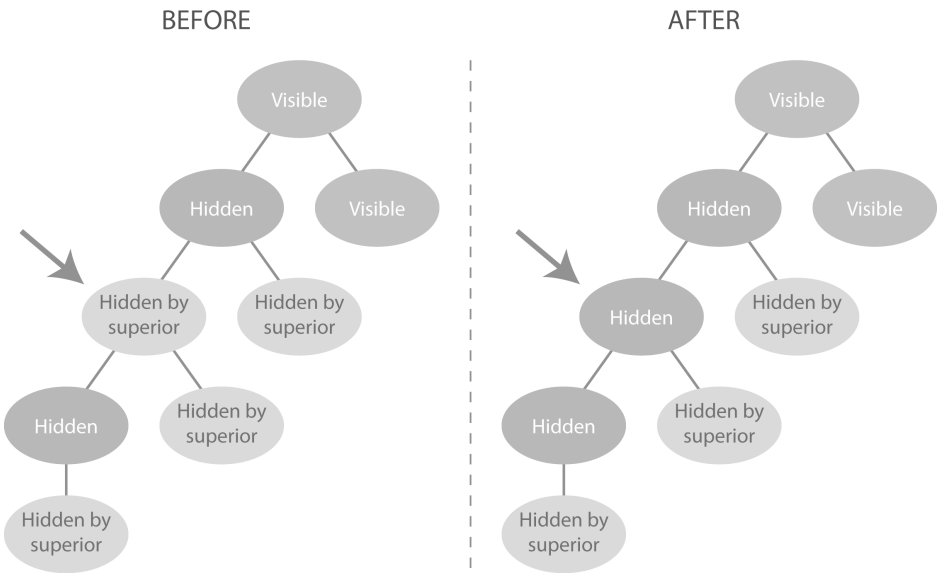


Figure 6.8. Example: hiding a node with an invisible parent

This operation is useful in cases where you want to hide a sub-branch before revealing a previously hidden larger part of the tree, since the sub-branch would remain hidden.

If a user tries to hide a node that is already invisible (and has the "hidden by superior" status), only the visibility status of that specific node will change. This is so because its subtree has already been affected by a previous hide operation higher in the tree. In the screenshot, the targeted node status changes from "hidden by superior" to "hidden", while all other nodes remain unchanged.

6.5.3. Revealing nodes

A hidden node (or subtree of nodes) can be revealed using the same techniques as described earlier: either access the context-sensitive pop-up menu or the **Locations** window. Note that you cannot directly reveal a node with a "hidden by superior" status, since it has inherited its invisibility from a hidden node higher in the node tree.

Revealing a node with a "hidden" status will make it visible if it has a visible parent. If it has an invisible parent (with a "hidden" or "hidden by superior" status), it will receive the "hidden by superior" status. The effect will also propagate to descendants of the target node.

6.5.3.1. The effect of revealing a node with a visible parent

The following illustration shows the case when a target node with a "hidden" status is revealed below a visible node:

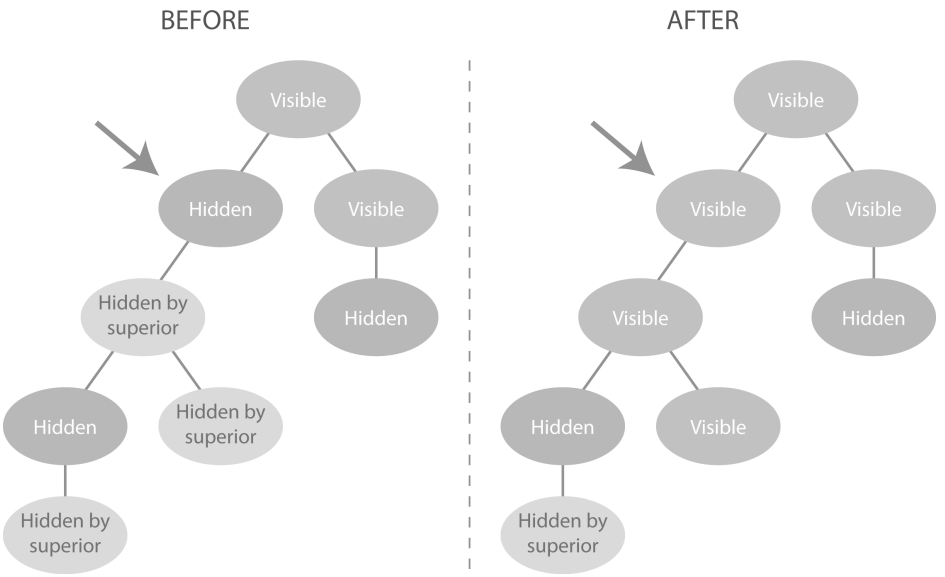


Figure 6.9. Example: revealing a node with a visible ancestor

Observe the change from being hidden to visible for the target node. Its descendants also become visible. However, notice in particular that the propagated effect stops when a node with the "hidden" status is encountered. If a node has a "hidden" status and one of its ancestors becomes visible, the node will remain hidden and its descendants will remain invisible.

6.5.3.2. The effect of revealing a node with an invisible parent

A node with an invisible parent means that higher up in the tree, a node (perhaps even its direct parent) has a "hidden" status:

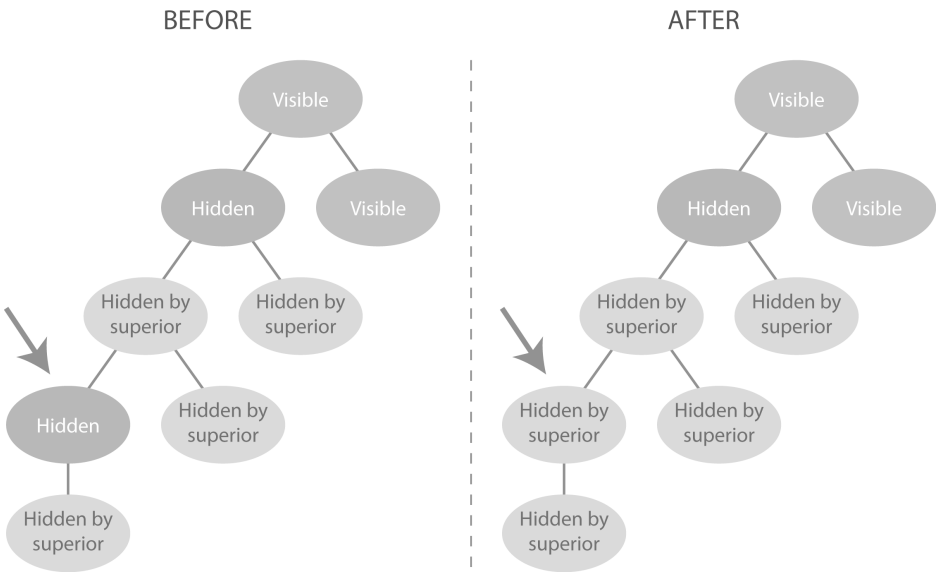


Figure 6.10. Example: revealing a node with an invisible parent

Although the target node remains invisible after the reveal operation, its status changes from "hidden" to "hidden by superior". This is because there is still a superior node that has a "hidden" status. In the right side of the screenshot above, one would need to also explicitly unhide the node with the "hidden" status in order to make all nodes in the example tree visible.

While it might not be apparent at first, this can be quite useful. For example, if you have an invisible subtree for some new product documentation in progress, that subtree might have several different sub-branches dedicated to different parts of the documentation. By first explicitly hiding these sub-branches, they can subsequently be revealed to internally signify that they have been completed. When the parent node for the product documentation is finally revealed on the product launch date, all of its descendants will become visible (since they all have the "hidden by superior" status). If any parts of the documentation are not yet ready during the launch date (in other words, one or more of the nodes has a "hidden" status), they will remain invisible.

6.6. Hiding before publishing

The previous hiding scenarios in Section 6.5.2, “Hiding nodes” deal with content that is already published. You can also mark content as hidden before clicking the **Send for publishing** button in edit mode. You can only do this for the first version of an object. If you are about to edit a visible object that you want to make hidden, simply hide it before entering edit mode.

Hiding a node before publishing it is typically used for preventing the display of unfinished work; otherwise, the node is visible for at least a few seconds (before you can hide it) after it is published. It can also be used as a manual delayed publishing mechanism. Here, we use the term "manual" because you have to explicitly unhide the node later - Chapter 7 describes the system's delayed publishing feature, which prevents the content from becoming published until a specified time.

In order for this feature to be available, the **Locations** window must be enabled for edit mode. You can toggle this in the **Edit mode settings** window on the left side of the **My account** tab.

Through the **Locations** window in edit mode, you can control where the object being edited should be published and how:

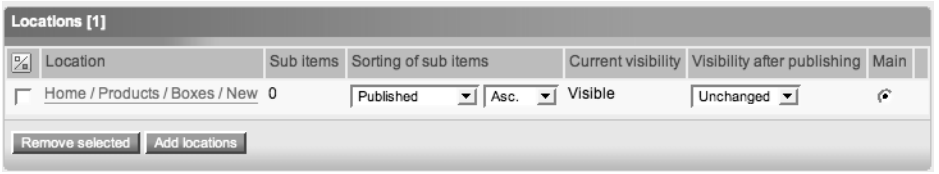


Figure 6.11. Locations window in edit mode

Once you have published an object (visible or invisible), it is part of the content node tree and thus you will be able to add sub-items to it. For example, you can publish an unfinished article as hidden, then create sub-pages for the article, and finally reveal the hidden article (and its sub-pages) when ready. This is handy since you cannot add sub-items to Draft objects (which do not yet have a location). The edit mode **Locations** window is disabled for all subsequent versions of the object.

You may also wish to stay in the **Object Edit Interface** after clicking the **Send for publishing** button in order to continue editing. This is useful to, for example, add related content. The "Re-edit" option must be enabled in the **Edit mode settings** window (accessed as described above). If it is enabled, you can mark the **Back to edit** checkbox at the bottom of the **Object Edit Interface**.

Remember that objects with the "Draft" status are not yet part of the content node tree, and visibility information is recorded within a node. Therefore, setting "Visibility after publishing" to "Hidden" in the **Locations** window in the **Object Edit Interface** has no effect when you click the **Store draft** button. For the same reason, the hide / show feature cannot be combined with delayed publishing (see Chapter 7) or the collaboration system (Chapter 8), since these features are based on postponing publishing (and thus the node assignment) until some event takes place.

6.7. Summary

Node visibility refers to whether or not some published content can be viewed on the front-end of a site. Hiding a node will render the target node and the entire subtree below it invisible. This is the correct way to "unpublish" content without deleting it or moving it to a restricted section.

The three visibility statuses are "visible", "hidden" and "hidden by superior". You can alter the visibility status of a node from the pop-up menu or the **Locations** window.

Chapter 7. Delayed publishing

Usually when you publish an object in eZ Publish, its status changes immediately from "Draft" to "Published", thus appearing on the front-end of the site. The *delayed publishing* feature gives content managers control over when content is published, so that you can specify a time other than the moment you click the **Send for publishing** button. You can also specify times for the system to automatically unpublish an object (essentially removing it) or to hide its node(s).

Delayed publishing is particularly useful when you want to synchronize or postpone content such as company press releases or news that might be written in advance of its intended release. This chapter applies to anyone who might use this feature, and especially to content managers who work with specific publishing dates. Our examples use the Administration Interface where some of the delayed publishing features are located. Content editing related to delayed publishing can also be done through the Website Interface.

Recall that:

- A content object has one or more versions. Each version has a status, and the object itself has a status.
- An eZ Publish *INI file* is a configuration file that controls the behavior of a specific part of the system. See Section 3.5, “The configuration model”. The main configuration file is `site.ini`. You can check the value of a particular setting within the **Settings view** interface as described in Section 3.5.4.1, “Viewing the settings of a configuration file”.
- A *workflow* is a sequential list of *events* that is initiated by a *trigger*. See Section 3.7, “The workflow system”. The initiation can occur before or after a function of a module completes, depending on the trigger. The details about a workflow can be viewed and modified in the **Workflow editing** interface, as described in Section 3.7.1.2.1, “Workflow editing interface”.
- *eZ Flow* is an eZ Publish extension aimed at media sites that make rapid and frequent content changes. A *layout* is a combination of *zones* placed on a frontpage. A *block* is an area within a zone that contains a particular type of content. The **eZ Flow** interface (see Section 2.5.3, “eZ Flow interface”) is a special-purpose edit mode for managing frontpages.

In this chapter, you will find information about the following:

- Requirements for the delayed publishing feature
- How to postpone publishing to a specific point in time

- How flow scheduling, automatic rotation, overflow, and timeline previewing work for eZ Flow frontpages

In this chapter, we assume that you are familiar with the default publishing process. Consult the *eZ Publish Content Management Basics* book if you need to review the basics. For details on INI files and the workflow system, refer to Chapter 3. Although we explain the requirements for the delayed publishing mechanism, we do not go into technical detail about how to set it up. This task is usually assigned to a site administrator.

For a conceptual introduction to delayed publishing, it is sufficient to cover Section 7.1, “What is delayed publishing?” and Section 7.2, “Requirements for using delayed publishing”. Otherwise, we recommend that you read the chapter in the order given. The last two sections only apply to frontpage scheduling with eZ Flow.

7.1. What is delayed publishing?

Normally, a draft version transforms into a published version when you click the **Send for publishing** button. Using the delayed publishing feature, the draft is given a "Pending" status, and remains so until it is automatically given the "Published" status at the specified publishing time. The delayed publishing feature can be used for new and existing objects. Regarding the latter, you might want to update an existing object but not display the new version until a specific date. For example, you might update a schedule that should only be revealed when some news is announced later. Therefore, while the updated draft version is given the "Pending" status, the previous version with the "Published" status is still displayed on the site. An example of this is illustrated later in Figure 7.9, “Version history interface - subsequent version pending”).

For content managers, the basic idea is to enter the desired publishing time in a designated attribute when creating or editing content. After you click the **Send for publishing** button, the workflow system handles the rest of the process. You can use this feature for both batch launches of objects and to specify a publication time for a single object. The time is specified per object, so a batch launch means to set the same future publication time for multiple objects.

Note that delayed publishing is not pre-configured by default, and has to be set up by a site administrator before it can be used. We outline the setup requirements next.

7.2. Requirements for using delayed publishing

There are some requirements in order for objects of certain content classes to work with the delayed publishing feature, as listed below:

- The content class must have an attribute of the "Date and time" datatype that specifies the publication time.
- There must be a workflow with an event of the "Wait until date" type, and an associated trigger (typically "content / publish / before").
- The workflow system must be running (see Section 3.7, “The workflow system”).

Consult your site administrator to check the configuration and operational status of the workflow system. Below we describe the datatype and event, in the case where the delayed publishing mechanism has been set up for objects of the `Article` class. This is for illustrative purposes only, as content managers usually do not have to set up the system.

7.2.1. Date and time datatype

To postpone the publishing of a content object, the system needs to know the time that publication should take place. This means that the object must contain an attribute of the "Date and time" datatype. Note that this datatype is not restricted for use with the delayed publishing feature. It is a standard attribute for storing time information.

The "Date and time" datatype stores a year, month, day, hour and minute value, as shown below for attributes in an article:

Figure 7.1. Attributes in an article, including the Publish date attribute

The input fields of the "Publish date" attribute (which is usually the attribute used to determine the launch date) may or may not contain default values, depending on how the attribute was specified in the class definition. For objects of the `Article` class, they are empty by default. If you leave them empty and click the **Send for publishing** button, the object is instantly published on the site.

7.2.1.1. Unpublishing content

Attributes of the "Date and time" datatype can also be used to set up timed unpublishing (putting the object in the trash when a date is reached), and for timed hiding (changing the node visibility to hidden when a date is reached). For example, the **Unpublish date** field in the previous screenshot represents an attribute that can be used to unpublish the

object at a certain time in the future. If eZ Publish is properly configured (as outlined in the in-depth block below), then the procedure for content managers is the same as for delayed publishing: you simply enter the date and time information (for when the content should be unpublished or hidden) into the appropriate field.

In Depth: Configuring timed unpublishing and timed hiding

Timed unpublishing and timed hiding are configured in INI file settings, rather than as standard workflow events. Timed unpublishing of objects is regulated by settings in the "[UnpublishSettings]" block in *`content.ini`* and the *`unpublish.php`* cronjob script must run periodically.

Timed hiding of nodes is regulated by settings in the "[HideSettings]" block in *`content.ini`* and the *`hide.php`* cronjob script must run periodically. Interested readers should refer to the technical manual at http://ez.no/doc/ez_publish/technical_manual for more information.

7.2.2. Wait until date event

Simply supplying a value in the "Publish date" attribute does not make delayed publishing happen. The second element that is required for the delayed publishing feature is the "Wait until date" workflow event. This is normally not set up by content managers, but learning about this workflow event will help you understand how the delayed publishing feature works.

In workflows set up for delayed publishing, the "Wait until date" event is commonly connected with the "content / publish / before" trigger. Triggers are viewed and configured by clicking the **Triggers** link in the **Setup** tab.

Let us look at the **Workflow editing** interface for an existing event to see exactly what the "Wait until date" event specifies. If you only want to review the current event options, simply click on the workflow name to view the workflow and its events.

Figure 7.2. Workflow editing interface

We have given this workflow the name "Press release launch" and it currently holds one item, the "Wait until date" event. In the middle of the screenshot you see a list titled "Class/attribute combinations", containing one entry. This simply tells the system to find the required information about the launch time in the "Publish date" attribute (which must be of the "Date and time" datatype) of the `Article` class.

By marking the **Modify the objects' publishing dates** checkbox, when the object is finally published, the timestamp value of the object will be set to the actual launch time. Otherwise, the time when the user clicked **Send for publishing** is used for the object's timestamp. Note that this checkbox setting applies to all content affected by this event, and is not a class- or attribute-specific setting. Timestamps are explained later in Section 7.6, “Launch time and object and version timestamps”.

In Depth: Delayed publishing sequence of events

When delayed publishing is properly set up on your site, the following is an outline of the sequence of events that takes place in a typical delayed publishing scenario for an article. Note that this is a slightly less technical version of what occurs, and that the article is not the only type of object that can be used with the delayed publishing mechanism.

1. A content manager enters edit mode for a new or existing article and fills in the desired content. At the moment, this version has the "Draft" status. If this is a new object, the object has the "Draft" status as well. In an attribute of the "Date and time" datatype, the content manager fills in a future publication date. Note that this attribute and content class must be in the **Class/attribute combinations** list of the "Wait until date" event. The content manager then clicks the **Send for publishing** button.

2. The "content / publish / before" trigger starts the specified workflow with a "Wait until date" event. The workflow is placed in a list of workflow processes (which can be viewed by clicking the **Workflow processes** link in the **Setup** tab). The version is given the "Pending" status.

3. A cronjob script (see "In Depth: Cronjobs" in Section 3.7.2, “Workflow-related requirements”) periodically checks the workflow processes. When the publish date is reached, the version is automatically published. Both the object and the version now have the "Published" status.

7.3. Pending items

Recall that versions that are waiting to be published by the workflow system are assigned the "Pending" status. A pending item can be either the first version of a new object, in which case the object is a draft; or a subsequent version of an existing object, in which case the object is published (and displays a previous version). A draft object with a pending version can only be viewed on the **My pending items** page in the **My account** tab by the creator of the pending version. If you have pending versions of existing objects, they are accessible here and also in the **Version history** interface for the object.

Figure 7.3. Pending items list

Note that the pending list will also contain pending items associated with the collaboration system (see Chapter 8).

Click on the name of the object to bring up the **Version preview** interface for that version. The screenshot below shows an object that has not yet been published.

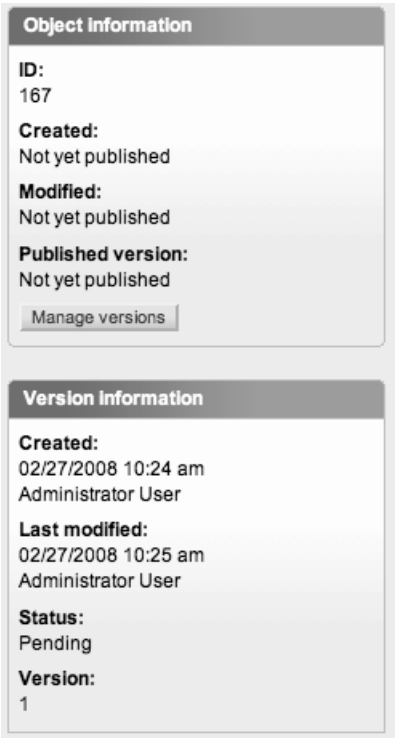


Figure 7.4. Version preview interface - object not yet published

Since it is not yet published, this object is not yet part of the node tree. Unlike drafts, pending versions cannot be edited, not even by their creators. You also cannot move or remove the object. In other words, you cannot do anything but view a pending version (and only if you are the creator or approver) until it is given the "Published" status at the

launch time. Therefore, be sure that a version is ready to be published before submitting it for delayed publishing.

Pending versions of existing objects can be copied in the **Version history** interface, becoming drafts that can be edited. You can also click the **Edit** button for the object. In that case, the new draft is a copy of the published / current version.

It is possible to have multiple pending versions of the same published object at the same time, given that they all have a publishing date in the future. Once the time is reached for one of the pending versions, that version is published and set as current.

Also, if you publish a more recent version of an object while an old version is pending, the newer version will still be replaced by the older one on the front-end of a site when the publish time for the older pending version is reached. This is shown in the screenshot below, where version "4" was published when version "3" was still pending. Once the publish time for version "3" was reached, version "3" was published and version "4" was archived.

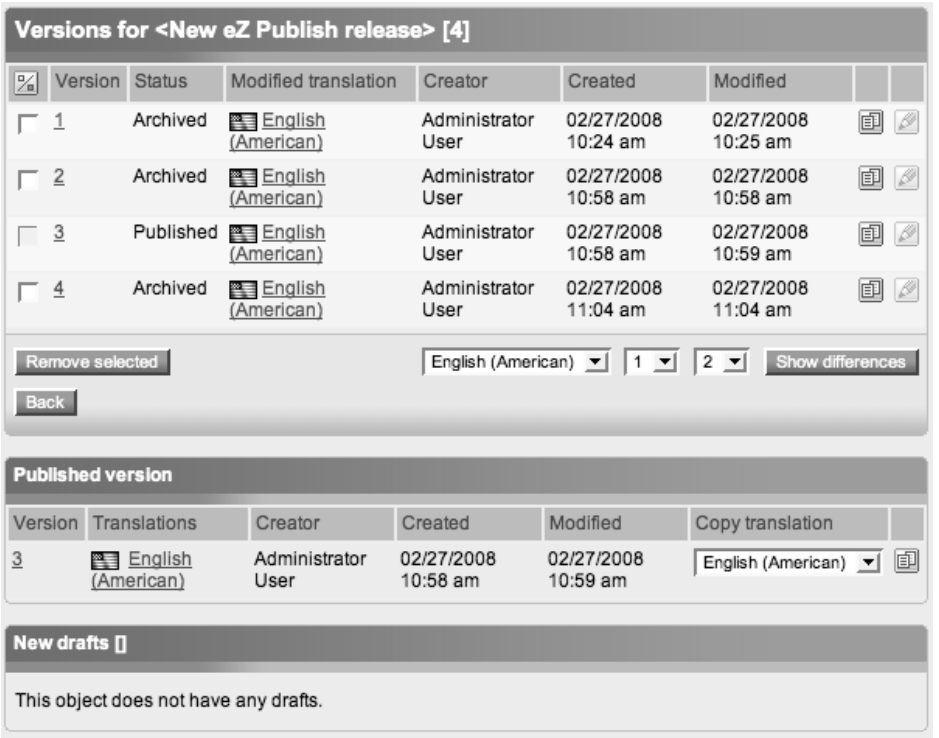


Figure 7.5. Version history interface - older version published at a later time

This is not an error, since the system simply does what it was told: to publish an object's version at the time specified. However, we recommend that you do not schedule more than one version per object at the same time, since it usually results in a rollback and is

rarely what you wanted. There is no way to cancel a pending item. Instead, you have to set a publish date for another version to be directly after the publish date of the original pending item, or manually edit the object the moment after the pending item is published.

7.4. Specifying the date and time values

Recall that attributes of the "Date and time" datatype have five input fields. The "Publish date" attribute of the `Article` class is not required by default. If you do not enter a value for this attribute, the object will be published directly when you hit the **Send for publishing** button. In other words, the delayed publishing feature is used only when you fill in the fields representing the appropriate attribute.

When you use the attribute, you must enter a value (or use the initial ones) for all five input fields. For example, if you attempt to save or submit the object with values for only the month and day, the datatype input validation process will prevent the operation and display an error message similar to the one below:

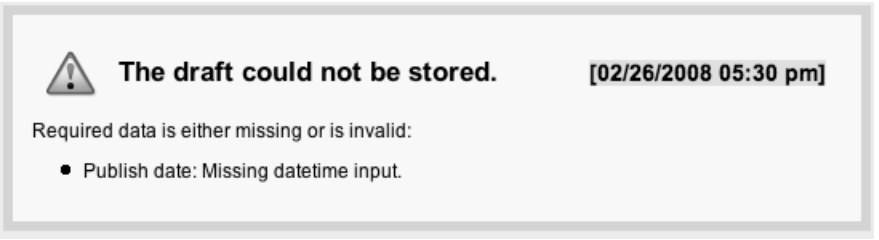


Figure 7.6. Date and time input validation error

If you set a time in the past, such as the current time minus five minutes, the object is published instantly. This is the same result as when you do not enter any values for the publish date.

Tip

If you use delayed publishing to synchronize the launch of multiple objects or an entire part of a site, you should make sure that all the objects within a launch scenario share the exact same value in this attribute.

If you wish to do a batch launch but the actual timing is of minor importance, consider using the hide feature rather than delayed publishing. In that case, you would publish a container node as hidden (see Section 6.6, “Hiding before publishing”), then send the sub-items for publishing as normal. These sub-items would not be revealed until you change the visibility status of the hidden ancestor. This saves you from the tedious and repetitive work of ensuring uniform timestamp values and also enables you to create content at several levels of the node hierarchy (since you cannot add sub-items to an object that is not yet published).

The following section goes through a detailed example on how to use the delayed publishing feature.

7.5. Example: Issuing a press release at a given time

The following procedure illustrates how the delayed publishing feature can be used, assuming that the requirements listed in Section 7.2, “Requirements for using delayed publishing” have been met.

1. Select a location for the new content, and enter edit mode to create an object of the desired class. In our example, bring up the pop-up menu for the "Company" folder in the left tree menu. Then, select "Create here - Article".

2. Enter the desired content for the object, including a date and time in the future in the **Publish date** field. (In our example, we entered the date for February 27, 2008 10:30, which was a future date at the time the example was created.) Click the **Send for publishing** button.

3. Provided that the workflow and trigger are set up properly, you will now see exactly the same contents in the tree menu as before. This is correct, since the node tree only holds published content, and the launch time you entered is some time in the future.

   Optionally, go to the **My account** tab and click the **My pending items** link in the left menu. You should see the newly created object in the list of pending items, as shown in Figure 7.3, “Pending items list”. Once the object to be published in the future has been registered by the workflow system, it will show up in the list of

workflow processes in the **Setup** tab (shown when you click the **Workflow processes** link in the left menu).

4. When the launch time is reached, the article is published, showing up in both the node tree and on the front-end site. The result is shown in Figure 7.7, "Published press release when publish date is reached" below. The object will also disappear from the list of pending items and the list of workflow processes.

Figure 7.7. Published press release when publish date is reached

7.6. Launch time and object and version timestamps

Each content object, and all of its versions, have two timestamp values maintained by the system. The *Created* timestamp indicates the time during which editing was initiated for the version or object. It corresponds to the time when the **Create (here)** button was clicked. The *Modified* timestamp indicates when it was last modified. This is updated every time changes are made, such as when you click the **Store draft** button or publish a new version.

These timestamps can be seen when viewing a published object in the **Details** window and the **Preview** window in the main area. In addition, timestamps are included in the following interfaces:

- The **Version history** interface shows timestamps for all versions of an object.
- The **Object information** and **Current draft** windows within the **Object Edit Interface** (see Section 2.3.5, "Object Edit Interface") show timestamps for the object and the version being edited.

- The **Version preview** interface shows timestamps for the version being previewed and the object itself.

The Created and Modified timestamp values of an object and version should not be confused with the value submitted in the "Data and time" attribute used for delayed publishing. The former is handled exclusively by the system, whereas the latter is a user-controlled content object attribute.

If the workflow event is set to modify the object's publication time (see Figure 3.25, “Workflow editing interface”), you may observe that some of the timestamp values change during the transition from draft to pending to published. We illustrate these changes here, building on the press release shown in Figure 7.4, “Version preview interface - object not yet published”. After this object was published, the timestamps were updated as shown below.

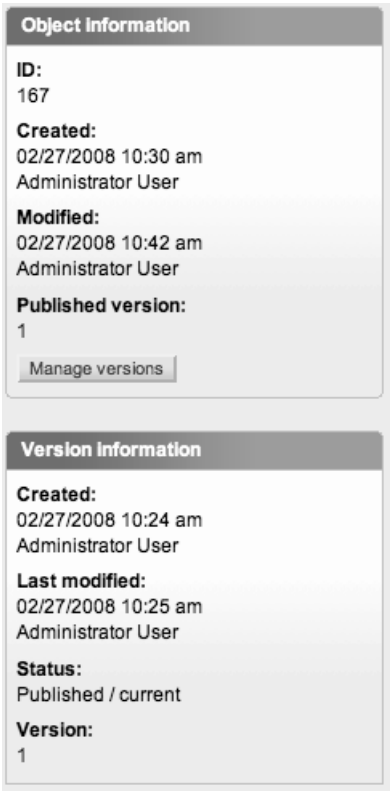


Figure 7.8. Version preview interface - object after publishing

If you compare this screenshot to the previously shown press release, you can verify that it is the same object with an Object ID of "167". The **Version information** window contains the same timestamp values between screenshots for when the version was created and last modified. The version number is "1" in both cases, but the version status has

changed from "Pending" to "Published". This difference in the **Version information** window applies regardless of whether the **Modify the objects' publishing dates** checkbox is marked for the workflow event.

The changes shown in the **Object information** window are more radical. Before the launch, both timestamps and the version number display as "Not yet published". After the launch, the published version is set to "1", and the Created and Modified timestamps are set. In this example, the time value given when editing the object was 10:30. Because the **Modify the objects' publishing dates** checkbox is set for the "Wait until date" event in our case (see Section 7.2.2, “Wait until date event”), the object's Created timestamp equals this value. Otherwise, it would get the value from the modification time of the published version.

Another difference in Figure 7.8, “Version preview interface - object after publishing” is that the object's modification time is 10:42, 12 minutes after the specified launch time. The reason for this has nothing to do with the checkbox, but rather with the workflow system itself. The system was unable to carry out the event at the exact time. Remember that workflow scripts are run by cronjobs at periodic intervals. In our case, the given time fell in between two checkpoints. The pending item, which was ready to be processed, was not processed until the script ran. Once the request was serviced and the object published, the system used the current time for the last modified timestamp.

Subsequent edits of an object that was initially subject to delayed publishing are published instantly, unless you alter the appropriate attribute of the "Date and time" datatype (the "Publish date" attribute in our examples) to be some time in the future. In that case, you may have a scenario like this when looking at the **Version history** interface:

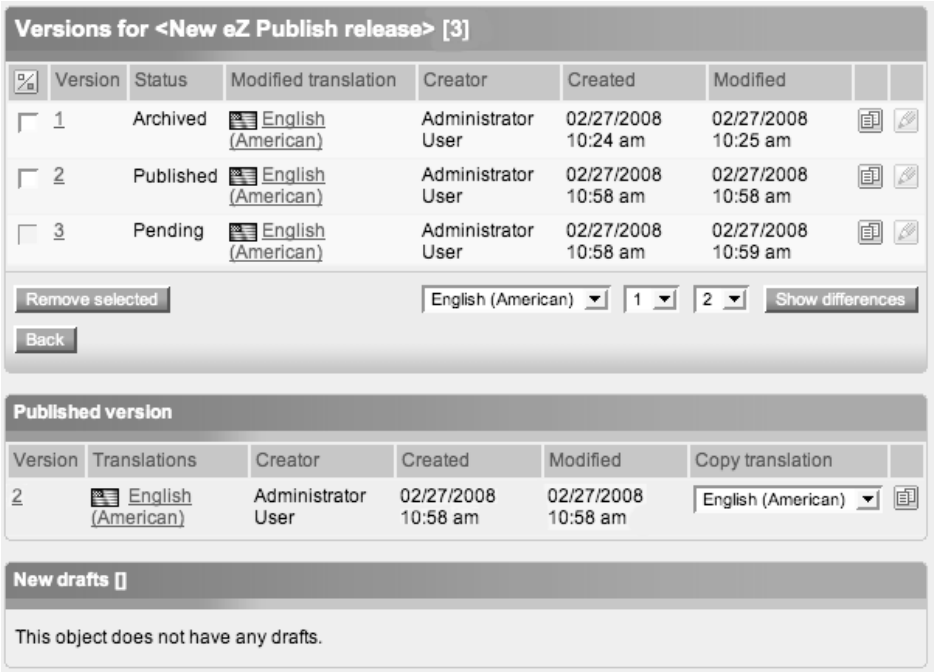


Figure 7.9. Version history interface - subsequent version pending

In this case, version "2" was created and published without modifying the "Publish date" attribute, and got instantly published. In version "3", however, we altered the publishing time, and as a result, its status is "Pending" and version "2" is still the published one. Note also that the draft list at the bottom of the screen is empty. This is because pending versions are different than draft versions, even though both statuses are assigned to content that is not yet published.

7.7. Delayed publishing and node visibility

It is not possible to combine delayed publishing with hiding an object before it is published. This is because node visibility works on published objects (as explained in Section 6.6, “Hiding before publishing”), while delayed publishing delays objects (or versions) from becoming published.

That said, delayed publishing has some similarities with node visibility and hiding content before it is published. In both cases, display is prevented (until some future event / action), but there is a difference in how the object and version statuses are set. When marking an object as hidden before it is published, both the object status and version status will be "Published", but it will have a "hidden" visibility status (or "hidden by superior" status

for its sub-items). The object is part of the node tree in this case. With delayed publishing, the object will have the "Draft" status (for existing objects, the object will still have the "Published status"), and the version will have the "Pending" status (Section 7.3, “Pending items”) until the launch date. In this case, the pending content is not yet part of the node tree.

Depending on your needs, you should select the feature that best matches what you want to achieve. If you need to prevent content from being displayed publicly, but it needs to be in the node tree and editable in the Administration Interface, you should hide it. However, keep in mind that it will affect the entire sub-branch. Also, a content editor has to explicitly alter node visibility for it to become available on the front-end. With delayed publishing, the launch process is handled automatically by the system. If you want to ensure that it is revealed at the appropriate time, use delayed publishing. Also, you should use delayed publishing to postpone the display of subsequent versions of a published object, since hiding it would render even the existing published version inaccessible.

7.8. Delayed publishing and notifications

The notification system (see Section 1.4, “Notifications”) is usually not combined with delayed publishing, since it would be redundant to alert the user that scheduled the delayed publishing about the pending version. However, once the launch time is reached and the object is published, users subscribing to subtree notifications on the parent node would receive notification emails.

7.9. Content flow with eZ Flow

We now turn our attention to the more sophisticated content scheduling of eZ Flow frontpages. This includes the automatic overflow of items between blocks, as well as the scheduled flow and automatic rotation of items within blocks.

7.9.1. Overflow

Recall from Section 2.5.1, “eZ Flow concepts”, that each block can only display a predefined number of items at a given time, and that an overflow rule determines what happens when an item is no longer displayed within a block.

When an item flows from the "Queue" list to the "Online" list, this usually pushes an item from the "Online" list to the "History" list. The **Overflow** dropdown list in the top right of a block in the **eZ Flow** interface lets you select another block within the same zone. When another block is selected, items that are no longer displayed can be automatically moved to that other block, instead of simply staying in the "History" list.

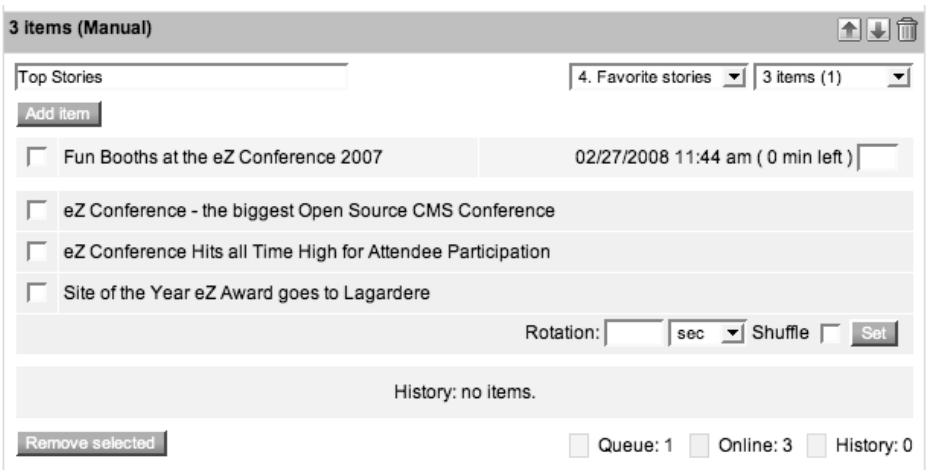


Figure 7.10. Block with three items online and one item in the queue

The above screenshot shows a block of the "3 items" type that currently has one item in the queue and three items online. With an overflow rule, when the item in the queue goes to the "Online" list, the oldest item from the "Online" list is added to the queue of another block, and indicated as being moved in the "History" list of the original block:

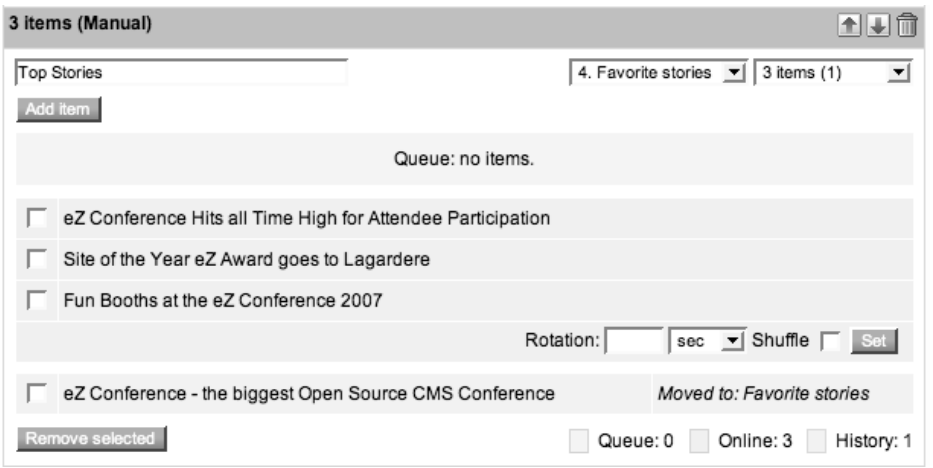


Figure 7.11. Item moved according to overflow rule

7.9.2. Flow scheduling

By default, when you add items to a block and publish the object, those items are instantly moved from the "Queue" list to the "Online" list and thus displayed on the site. However, with *flow scheduling*, you can time the flow of each item in the queue, specifying in how

many minutes an item from the queue should move to the "Online" list. This enables a frontpage to be updated with fresh, pre-created content on a pre-determined schedule.

To specify flow schedules, first add some items to the block, as was described in Section 2.5.6, “Adding content to a block”. These items will be added to the queue. To the right of each item in the queue is a **Flow** input field:

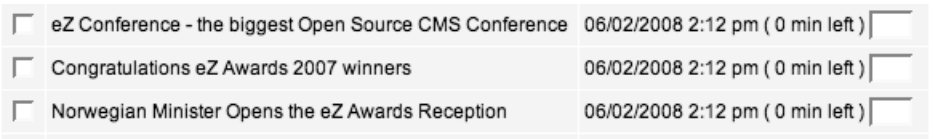


Figure 7.12. Flow input field for queue items

Enter a time in minutes in the **Flow** input field for each queue item for which you want to time the flow, then click the **Set** button at the bottom of the "Online" list. After you publish the object itself, an item will remain in the queue until the time specified has elapsed, at which point the item will move to the "Online" list.

7.9.3. Automatic rotation

Automatic rotation works within a single block, rotating items between the "Queue" and "Online" lists. With automatic rotation, when items are flowed through the block, items from the "Online" list are rotated back through the queue, instead of being pushed to the "History" list or to another block. This is useful when, for example, it is the weekend and no news editors are working. Even though no new content has been pre-created, the existing items that are displayed can dynamically rotate.

The **Rotation** interface is located at the bottom of the "Online" list of a block. Here, you can specify the frequency with which items are moved online and through the queue, either sequentially or randomly (randomly rotating the order of the items):

Figure 7.13. Rotation interface

To set up automatic rotation, first add some items to the block, which will add them to the queue. Then, select a time unit in the **Rotation** dropdown list, specify the frequency in the **Rotation** input field, and click the **Set** button. If you mark the **Shuffle** checkbox, a random item will be moved online from the queue during each rotation. Otherwise, the rotation of items through the queue and the "Online" list is done in order. Do not forget to click the **Send for publishing** button to save the changes to the frontpage.

Tip

For flow scheduling and automatic rotation to work on the site, a cronjob (see "In Depth: Cronjobs" in Section 3.7.2, “Workflow-related requirements”) for eZ Flow needs to run periodically. Contact your site administrator for more information.

7.10. eZ Flow timeline preview

The eZ Flow *timeline preview* is a front-end tool accessed from the **Website Toolbar** that enables you to view what the frontpage will look like to site visitors at any given time. This is particularly useful because the contents change over time due to flow scheduling, and the standard preview feature only captures the current state of the front-page.

To access the timeline preview, log in to the front-end site and navigate to the frontpage you wish to preview. Click the **Timeline** button at the right of the **Website Toolbar**:

Figure 7.14. Website Toolbar with Timeline button

The page will automatically reload to include the timeline preview at the top:

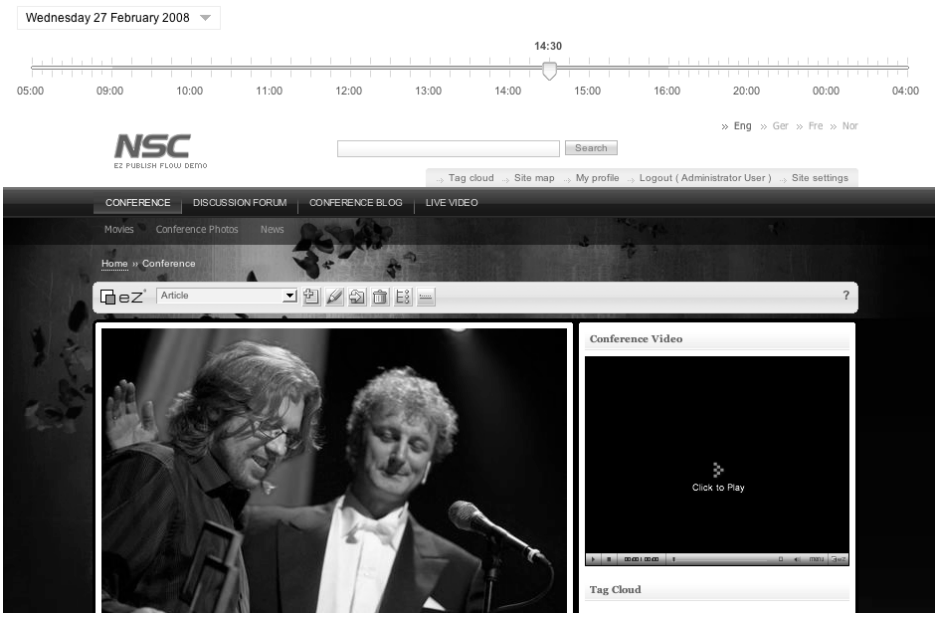


Figure 7.15. Timeline preview

By default, the slider enables you to preview the frontpage for times within a 24 hour period. Simply drag the slider left or right to the desired point in time. Once you let go of the slider, the page will automatically reload with the content that was or will be presented at that time, due to flow scheduling, overflow rules and automatic rotation.

You can also get previews for dates and times before or after the 24 hour period by using the **Date** dropdown list in the top left. After selecting the date, use the slider as normal. This is useful for checking that the flow is correct for nights, weekends, holidays and so on. By tracking back in time, you can confirm reported display problems in order to possibly correct them for the future.

Note that the timeline preview only applies to the frontpage as a whole. If you click on any links to other pages on your site, that page will load without the timeline preview.

7.11. Summary

The delayed publishing feature enables editors to control when content is launched. Assuming that the requirements for the delayed publishing feature are met, you simply have to specify the publish date in the appropriate attribute when editing an object. The workflow system takes care of postponing publishing from when the **Send for publishing** button is clicked until the given time is reached.

Delayed publishing is not limited to the first version of a content object. If you specify a future publish date for a subsequent version of a published object, the object will remain published, but with an older, existing version. When the publish date is reached, the pending version becomes the published version.

With eZ Flow frontpages, more sophisticated timed publishing is available through overflow rules, flow scheduling and automatic rotation. You can use the timeline preview to view the past and future display of an eZ Flow frontpage.

Chapter 8. Collaboration system

The eZ Publish *collaboration system* enables you to require that certain content to be approved by an editor before it is published. For example, you could have multiple editors submit material for final approval by an editor-in-chief, or you could make sure that all content created by site visitors (like comments, forum messages and blog posts) is checked before it is published.

This chapter outlines the details of the collaboration system and explains the relevant procedures and interfaces for content managers who submit content for approval or who approve content.

Recall that:

- An eZ Publish *INI file* is a configuration file that controls the behavior of a specific part of the system (see Section 3.5, “The configuration model”). The main configuration file is `site.ini`. You can check the value of a particular setting within the **Settings view** interface as described in Section 3.5.4.1, “Viewing the settings of a configuration file”.

- A *workflow* is a sequential list of *events* that is started by a *trigger*. The initiation can be done before or after a function of a module completes, depending on the trigger. See Section 3.7, “The workflow system”.

- The *notification system* enables users to be informed via email about certain events, such as content being updated or added to the site; and collaboration actions, such as when an article is waiting for editorial approval. See Section 1.4, “Notifications”.

In this chapter, you will find information about the following:

- How the collaboration system works, including collaboration notifications and the "Approve" workflow event

- How the "Pending" status is used by the collaboration system

- How different users work together when approving and denying content

For information on technical configuration of the collaboration system, refer to http://ez.no/doc/ez_publish/technical_manual.

We recommend that everyone who is working with the collaboration system should read the entire chapter, in order to understand how the different parts of the system work together. The collaboration system interacts with the workflow system, so we encourage you to first review Section 3.7, “The workflow system”.

8.1. Introduction to the collaboration system

With the collaboration system, a user first creates an object or a new version of an object, then clicks the **Send for publishing** button as usual. However, the object or version is not published immediately. An approver (such as an editor-in-chief) has the power to approve or reject the content or changes to the object. As you will learn in Section 8.1.1.1, “Approve workflow event”, the collaboration system can be applied under different circumstances, based on sections, languages, and more. Also, there can be multiple approvers, of which only one is needed to approve or deny a specific piece of content.

If the content is approved, the object is published. If the content is denied, the creator of that particular version can edit the content and re-submit it for approval. The approver (and the author, depending on his or her user account) also has the ability to comment about the object in order to provide feedback to the author or other approvers.

8.1.1. Collaboration system requirements

As with the delayed publishing feature described in the previous chapter, the collaboration system interacts with the workflow system. The collaboration system requires a workflow with an "Approve" event, associated with the "content / publish / before" trigger. The workflow system must also be running. See Section 3.7, “The workflow system” for details.

In order to access the pending items in the **My account** tab, authors and approvers need to be assigned a role that includes a policy granting access to the "pendinglist" function of the "content" module. In order to access the **Collaboration** interface (described below), a policy granting access to the "collaboration" module is required. Refer to Section 5.7.2.3, “Working with policies” for more information on how you can check and grant the access permissions for relevant users and user groups. By default, the Editor role includes access to the pending list but not to the **Collaboration** interface.

8.1.1.1. Approve workflow event

The "Approve" workflow event is an important and necessary part of the collaboration system. Setting it up is normally not handled by content managers, but this material will help you understand how the collaboration system works.

Let us look at the **Workflow editing** interface for an example workflow with the "Approve" event, in order to see exactly what this event specifies. You can access this as described in Section 3.7.1.2.1, “Workflow editing interface” by clicking the **Workflows** link in the left menu of the **Setup** tab. Then, click on the workflow group that contains the desired workflow and finally click the **Edit** button beside the workflow. If you only

want to review an event's options, simply click on the workflow name to view the workflow and its events.

Figure 8.1. Workflow editing interface showing an "Approve" event

Some of the options within the "Approve" event represent approval criteria. This set of criteria determines whether content needs to be approved before it is published:

- Section: specified in the **Affected sections** selection list (see Chapter 4 for more information about how nodes are grouped into sections). For example, you can specify

that all objects in the Standard section, which represents the Content branch, must go through the approval process.

- Language: specified in the **Affected languages** selection list. This means that you can have designated editors responsible for approving content in their native languages.

- Version: specified in the **Affected versions** selection list. You can specify that new objects, new versions of existing objects, or both need to be approved.

In the **Users who approve content** and **Groups who approve content** areas, content approval privileges can be assigned to both individual users and user groups. Note that only a single user needs to do the approval. In the above screenshot, both the Administrator user and the entire Editors user group have been assigned as approvers.

By default, content that meets the section, language and version criteria must be approved. In the **Excluded user groups** area, you can specify that some user groups can create and edit content that does not need to be approved. Content created by approvers is also excluded from the approval process.

Tip

When you need to have content approved that is also scheduled to be published at a later date, the collaboration system must be combined with the delayed publishing mechanism. This is commonly done by having a workflow with two events: one "Approve" and one "Wait until date". The order of these events depends on the desired behavior. Either you delay, then approve (content appears in **Collaboration** interface at the time specified in the "Publish date" attribute). Or, you approve, then delay (content appears instantly ready for approval, then remains pending until the publication date arrives. Ask your site administrator for in-depth information.

8.2. Pending items in the collaboration system

Once a piece of content that is subject to an approval event has been sent for publishing, it receives the "Pending" status. This status is also used for content subjected to delayed publishing, as described in Section 7.3, “Pending items”. In both cases, it is used to denote that the content version is waiting for some event to take place - either a specific point of time to be reached, or an approval action to take place.

Pending collaboration items will show up in the list of pending items in the **My account** tab of the user who submitted them:

Figure 8.2. Pending list for an author

They will also show up in the **Collaboration** interface (see below) in the **My account** tab of the user who submitted it and of the approvers.

If the pending version is approved, it receives the "Published" status. It will also be removed from the list of pending items and from the **Collaboration** interface.

If the pending version is denied, it receives the "Draft" status and the workflow process stops. The author can subsequently edit the draft and publish it again, in which case approval is required once again. Remember from Section 7.3, “Pending items”, that pending versions cannot be edited and also that drafts can only be edited by their owners. Therefore, approvers cannot edit objects in the **Collaboration** interface; if any changes need to be made, the approval request must be denied and then the object can be edited by the creator of the denied version.

8.3. Collaboration interface

The *Collaboration interface* is a special-purpose interface available to approving editors. To access it, click the **Collaboration** link in the **My account** tab. Here you can review, approve and deny content subject to be approved.

If you are working with content that often requires edits by the author, the **Collaboration** interface is important for authors as well. Here, authors can view messages about their approval items, as well as view approver comments and add comments. Note that the **Collaboration** interface is not for editing. Editing has to be done in the **Object Edit Interface**.

The illustration below shows the main page of the **Collaboration** interface for an approver.

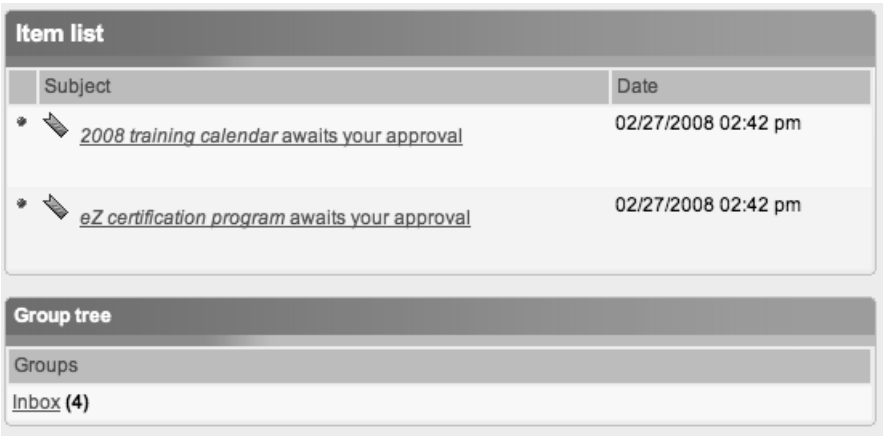


Figure 8.3. Collaboration interface main page

The main area holds two items: the **Item list** and **Group tree** windows. For approvers, the **Item list** window contains a list of object versions that are waiting to be approved. For authors, it shows all of their approval items, no matter whether they are pending or have been denied or approved.

The **Group tree** window provides a link to the Inbox group, which archives all of the collaboration requests. For authors, the contents of the **Item list** and **Inbox group list** windows are identical.

Clicking on the link for the Inbox group will reload the page, replacing the **Item list** window with the **Group list** window:

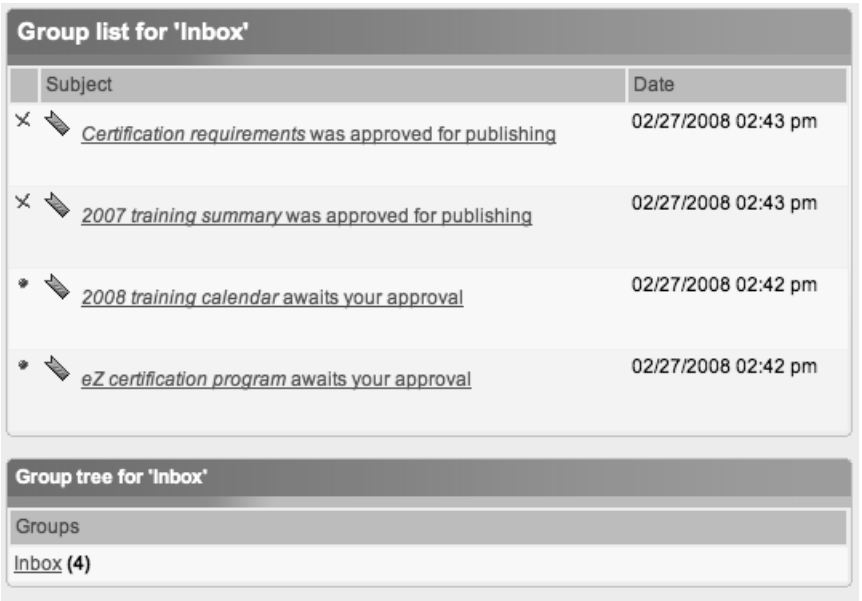


Figure 8.4. Collaboration interface Group list window

Here, our example shows messages for two new pending items, as well as for items that have been denied or approved. The list is sorted with the most recent entries at the top. If any items have associated comments, the number of comments is listed in parentheses for each item. A bullet indicates that the item is unread. The bullet will change to a hyphen when the item has been read and to a cross when the item has been addressed by being approved or denied. The icon to the right of the bullet, hyphen or cross simply represents a collaboration item, in the same way as you have folder and image icons.

Clicking on one of the entries (or the icon left to it) brings up the **Approval** interface, with the **Approval** window on top, followed by the **Preview** window and the **Participants** window. If the item has any comments, you will also see the **Messages** window at the bottom.

8.3.1. Approval window

The **Approval** window contains text describing whether the content has been approved, denied or is waiting to be approved, and a link to the **Version preview** interface so that you can see how the object would appear on the front-end of the site.

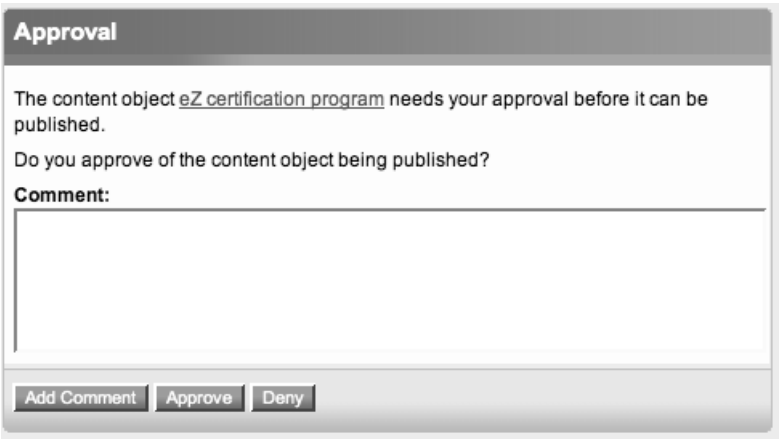


Figure 8.5. Approval window (approver) - item pending

If the content was rejected, authors will be shown a link to the **Object Edit Interface**. This provides easy access to correct problems pointed out by the approver in the **Messages** window.

The **Comment** input field is only shown for pending versions. Authors and approvers can relay messages to each other by typing text into the input field, then clicking the **Add comment** button to append the text to the list of messages associated with this item. This is useful for both approvers and authors to provide feedback to each other.

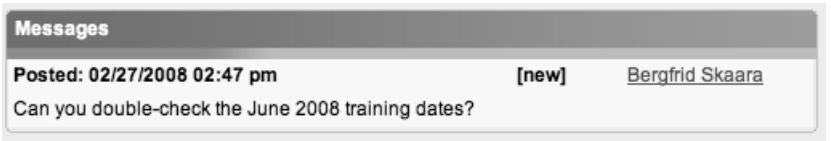


Figure 8.6. Messages window

Next to the **Add comment** button in the **Approval** window are the **Approve** and **Deny** buttons for accepting or denying the publish action for the current version. These apply only to approvers and are disabled for authors.

8.3.2. Preview window

The **Preview** window simply shows the contents of the attributes, without the front-end design markup. This is quite similar to the **Preview** window in the three left-most tabs.

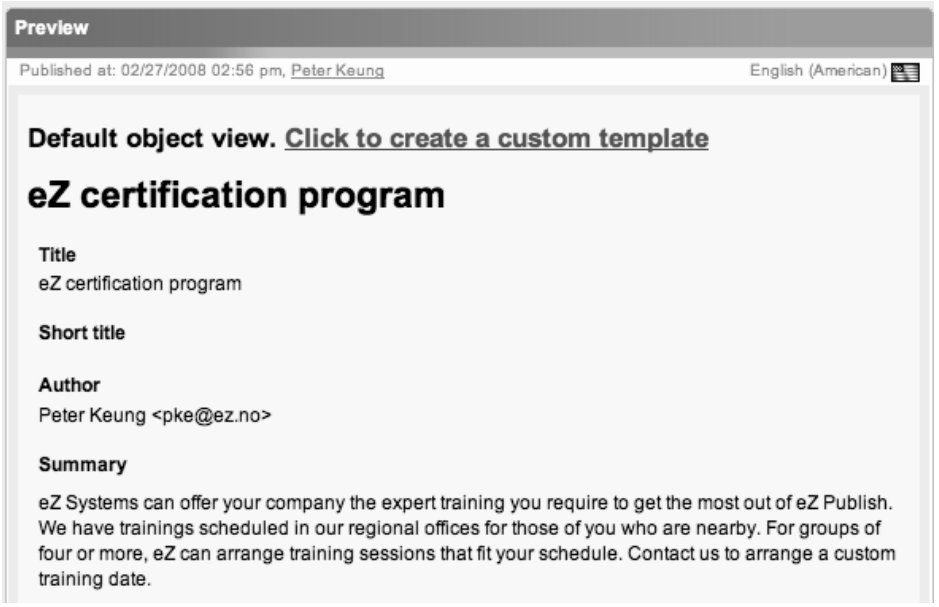


Figure 8.7. Preview window

8.3.3. Participants window

The **Participants** window lists who can approve the version and indicates who is the author of the version.

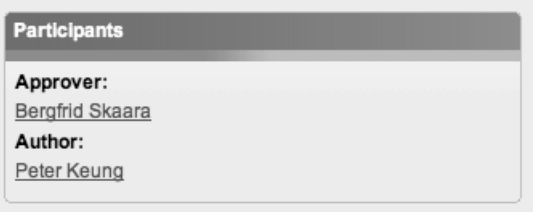


Figure 8.8. Participants window

8.4. Collaboration notifications

The notification system can be used to send email notifications about new collaboration messages. This means that approvers and authors can receive emails whenever there is some content waiting to be approved. At the time of writing, "Approval" is the only type of collaboration notification supported. It sends an email to the approvers whenever content has been submitted for approval, and it also sends an email to the author to ac-

knowledge the content submission. The following procedure illustrates how to sign up for collaboration notifications.

1. Access the **Notification settings** interface by clicking the **My notification settings** link in the left menu of the **My account** tab.

   This interface can also be reached from the user profile interface on the site's front end.

2. If this is the first time you sign up for notifications, you also need to specify how frequently you want to receive notifications, as described in Section 1.4.1, "Signing up for notifications".

3. At the bottom of the **My notification settings** window, mark the **Approval** checkbox.

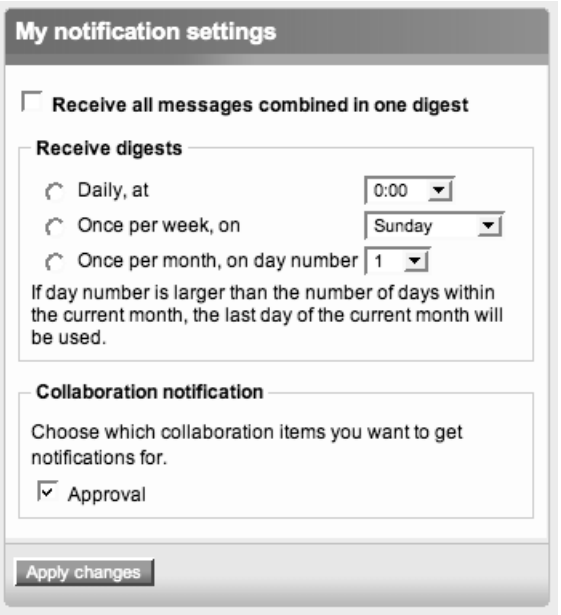


Figure 8.9. Sign up for collaboration notifications

4. Click the **Apply changes** button.

Note that you will not receive any notifications unless the notification system is running (see Section 1.4, "Notifications"). Also, the **My notification items** area at the bottom of the **Notification settings** interface applies to subtree notifications only.

Alerting the content author when content has been approved or rejected is currently not supported. Content authors can regularly check the **Collaboration** interface (if they have

access to it) to see whether their content has been approved or denied and also to read any relevant comments.

8.5. Example: approving content

This section explores an example scenario that outlines each step of the publishing process within the collaboration system. The example assumes that the technical requirements have been met (see Section 8.1.1, “Collaboration system requirements”) and that the content meets the criteria defined in an "Approve" event (see Section 8.1.1.1, “Approve workflow event”).

1. The author selects a content location, then creates (or edits) the desired piece of content, and clicks the **Send for publishing** button. The author can verify that the submission for approval is listed in his or her pending list (either through the Administration Interface or the Website Interface, depending on his or her access permissions), as illustrated in Figure 8.2, “Pending list for an author”.

   Alternatively, this step can be carried out by site visitors submitting comments, forum messages, blog posts and so on.

2. If signed up for collaboration notifications (and the notification system is running) the author will get an email saying something like "Article A awaits approval by editor", and the approving editor will get a message saying something like "Article B awaits your approval".

3. The approving editor then accesses the **Collaboration** interface. If you are not using the notification system¸ you should make it a habit to regularly check for new items in this interface. Note that items pending approval are only listed in the author's pending list, not for the approver(s).

4. The approving editor opens the pending item by clicking the subject name to review it. He or she can add a comment for other approvers or the author to read.

5. To approve and publish the object and version, the approving editor clicks the **Approve** button. The **Approval** window now indicates that the pending item has been approved:

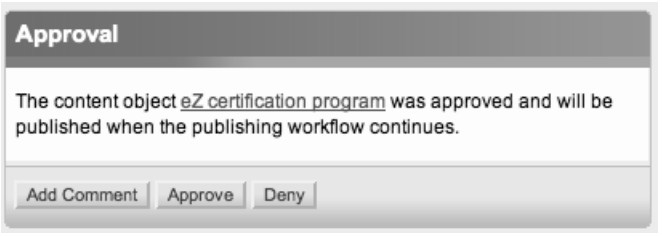


Figure 8.10. Collaboration interface - content approved

To reject the item, click the **Deny** button. The **Approval** window will display the following text to the approver:

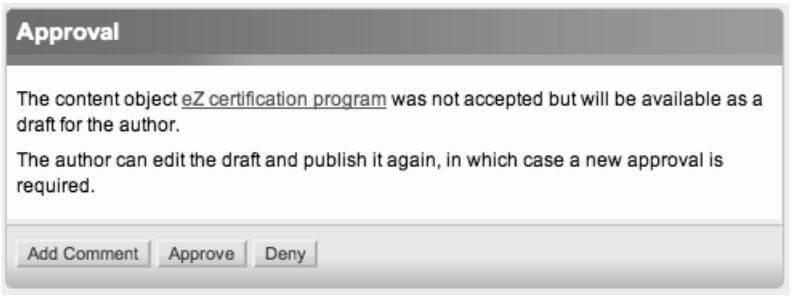


Figure 8.11. Approval window (approver) - content not approved

The corresponding message to the author is as shown in the screenshot below:

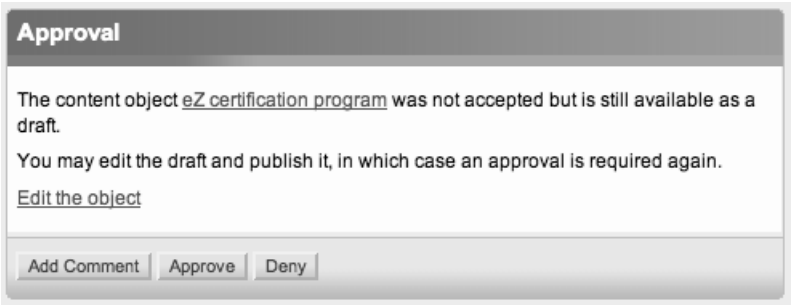


Figure 8.12. Approval window (author) - content not approved

6. Depending on the outcome of the approval process, the content is either approved to be published (and will be published when the appropriate cronjob script is run; see "In Depth: Cronjobs" in Section 3.7, “The workflow system”), or available for further edits by the author (at which point this procedure can be repeated).

If the author clicks the **Edit the object** link in the **Approval** window, he or she will be presented with an editing conflict warning ("the object is already being edited by you"). Simply click the **Edit selected** button to resume editing the rejected draft.

In Depth: Bypassing the collaboration system

If you are an approver and need to make minor changes to a new version of an existing object, you can use a workaround technique to edit and publish the object yourself. First, deny the approval request and let the author know that

you will handle it. (You can submit a comment in the **Collaboration** interface or alert the author directly.) Then, access the **Version history** interface for the object, make a copy of the denied draft (since drafts can only be edited by their creator), edit the copy and publish it. Note that this is only possible for subsequent versions of existing objects, since Draft objects are not part of the node tree and are thus inaccessible in the **Version history** interface.

8.6. Tips and troubleshooting

This section lists a few common problems related to the collaboration system, along with tips on possible solutions.

8.6.1. Approval step skipped

If you submit some content that you know should be subject to approval before publishing, but that content is instead published immediately, check the following:

- Check with your site administrator that the collaboration system is correctly set up.
- Does the content you submitted meet the approval criteria? For example, only new objects might need to be approved, and you might have simply edited an existing object. If you have the necessary access permissions, check the approval workflow in the **Setup** tab and review the configuration options in the "Approve" event (as was described in Section 8.1.1.1, “Approve workflow event”):

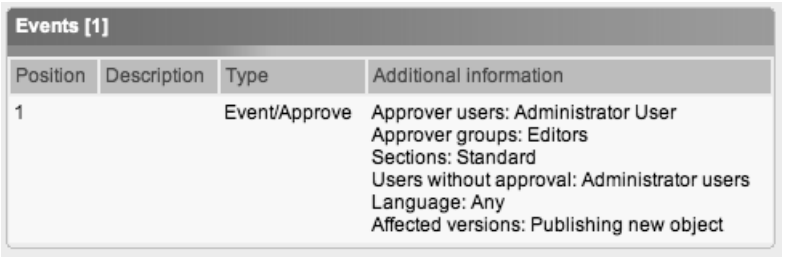


Figure 8.13. Workflow - check event details

- Is your user, or the user group you belong to listed as an approver or someone that can create and edit content without having to get it approved? Check the options for the "Approve" event.

8.6.2. Item has been approved but it is still not published

If the content has been approved, and this is verified by the **Approval** window as shown in Figure 8.10, “Collaboration interface - content approved”, the problem is most likely that the required cronjob script is either not running, or will soon run in its periodic cycle. Remember that there are settings for how often cronjobs run. Consult your site administrator.

8.6.3. You do not get collaboration notifications

Try checking the following to see why you do not get approval request notifications:

- Have you signed up for approval request notifications? Check the notification settings in the **My account** tab or in your user profile on the front-end of the site.
- Do you use the email account associated with the user account with which you signed up for notifications? Check the value of the email attribute in your user account settings (see Section 5.3.2.1, “Accessing your personal space”). Also, check to see whether the email has been filtered out by a spam filter or if your inbox is full.
- Are notifications processed frequently enough on your site? Ask your site administrator.
- Does email sending work correctly on your site? Ask your site administrator.

8.7. Summary

The collaboration system makes it possible to have content approved by an editor before it is published. When a draft is submitted, it will be pending until an approver has accepted or rejected it.

Authors can access submitted content from the pending list in their personal space if their user accounts or user groups have been granted access to the "pendinglist" function of the "content" module. Approvers can view, comment on, approve and reject content through the **Collaboration** interface if access is granted to the "collaboration" module.

Chapter 9. Web 2.0 - users as contributors

The progression of web technology and interactive websites has led to the term "Web 2.0". Websites have outgrown their role as static information providers. *Web 2.0* refers to enabling users to interact with and even create content, in order to customize and enrich the user experience. RSS feeds, polls, forums, wikis, comments, blogs, and tags are all Web 2.0 features that are built into eZ Publish.

This chapter is relevant to content managers working with blogs and tagging, as well as those working with optimizing content for search engines. Both blogs and tag clouds are front-end features, and editing is mostly done through the Website Interface. In contrast to most of the previous chapters, this chapter takes a front-end perspective.

Recall that:

- A *module* offers an interface for web-based interaction, providing access to eZ Publish core functionality. Each module has a set of *views* that enable you to reach the functions provided by the module. For example, the "edit" view of the "content" module is used for functions such as creating, editing and translating content.

- The *collaboration system* makes it possible to get content approved by an editor before it is published. When a draft is submitted, it will be pending until an approver has accepted or rejected it.

- The *delayed publishing* feature enables editors to control when content is launched. When a future publish date is entered into the appropriate attribute in an object, publishing is postponed from when the **Send for publishing** button is clicked until the given time is reached.

- The *permission system* consists of two parts: a set of user accounts (with associated user profiles) and access permissions. A policy is a single rule that grants access to specific or all functions of a module. A role is a named collection of policies that can be assigned to an individual user account or a group of users.

In this chapter, you will find information about the following:

- How tags apply to objects of certain content classes

- The utility of blogs as well as the related content classes, attributes, and datatypes

We will focus on tags and blogs, although they represent only a portion of the available Web 2.0 features. See *eZ Publish Content Management Basics* for information about

forums. *RSS feeds* are beyond the scope of this book, although they are mentioned in relation to blogs in Section 9.2.2.3, “RSS feeds”. *Wikis* are usually intended to let a group of people create and maintain a body of information. They are typically created using Documentation page objects for content, and use access control (see Chapter 5) to provide users with appropriate editing permissions.

For a conceptual introduction only to tags and blogs, we recommend that you read Section 9.1.1, “Keywords datatype” and Section 9.2.1, “Blog-related content classes”.

9.1. Content relevance with tag clouds

Tags represent categories, or keywords, that relate to specific content objects. This helps to create interfaces where a user can quickly browse other pages with similar or related content (for example "see also" or "related pages" lists of hyperlinks, and tag clouds). *Tag clouds* are a quick way for visitors to get a visual idea of the most popular topics on a site, and to quickly focus on topics of interest:

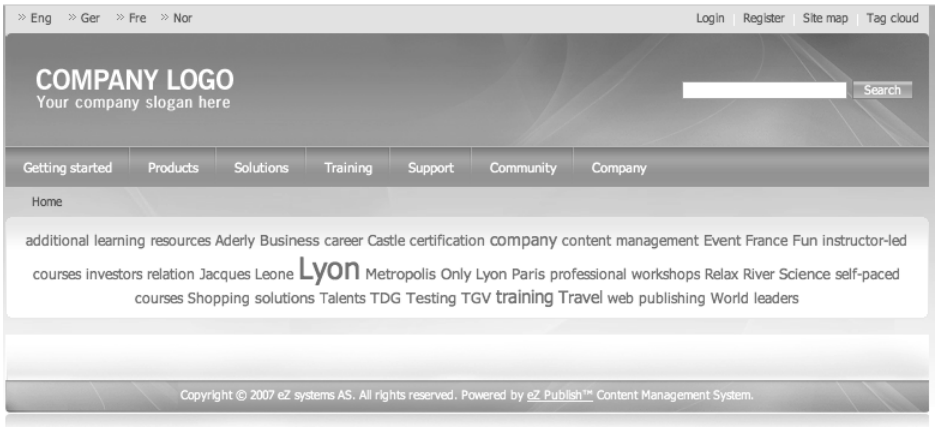


Figure 9.1. Tag cloud

9.1.1. Keywords datatype

The *Keywords* datatype enables you to store keywords for a content object as a comma-separated list, such as "Telemark, Skien, Norway, eZ Systems". This datatype is commonly used to build connections within your published content. Tag clouds are one of the applications of the "Keywords" datatype.

Many of the built-in content classes of sites using the Website Interface 1.2 and higher have a "Tags" attribute, which is of the "Keywords" datatype:

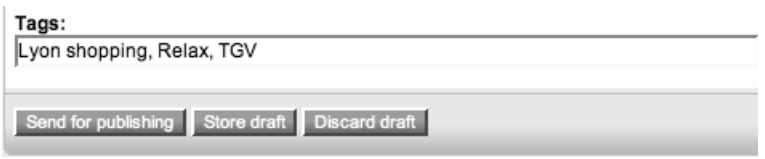


Figure 9.2. Tags attribute

The supported classes include the `Article`, `Blog`, `Blog post`, `Documentation Page`, `Folder`, `Product`, `File` and other multimedia classes. You can enter multiple tags into the corresponding input field shown above, separating each tag with a comma. Each tag can contain multiple words, so be sure to position your commas correctly when entering the keywords.

Tip

Remember to update your keywords when editing content. Keywords that have lost their relevance reduce the usefulness of the tag cloud.

By default, the "Keywords" datatype automatically connects objects, given that two objects have at least one common keyword. The connections are handled by the system by special-purpose database tables and the display is dictated by templates. In other words, this is not the same as object relations (see the "Object relation" entry in the Glossary), which you can view through the **Relations** window of the Administration Interface.

The next section describes how tag clouds are built using keywords.

9.1.2. From keywords to tag clouds

The tag cloud feature is implemented through the template system (see Section 1.1.3, “Templates”). This means that you cannot simply edit a tag cloud - it is built based on the keywords you enter for the "Tags" (or similar) attributes when editing content. Used on their own, the words "tag" and "keyword" both refer to the same thing.

On the front-end of a site, visitors can view the tag cloud for the entire site by clicking the **Tag cloud** link in the top right corner. The cloud is built starting from the Content top-level node, using a special-purpose view ("tagcloud") of the "content" module. In other words, it shows the tag cloud for all of the nodes in the Content branch.

Tip

If you look at the latter part of the URL for the site tag cloud, it reads "content/view/tagcloud/2". You can manually force a tag cloud to display from another origin by replacing the Node ID "2" with another ID. This is particularly useful when working with improving your keywords. For example, choosing the ID "137" in our demo installation will generate a tag cloud for the "Company" folder and all of the nodes below it. By narrowing the cloud to a specific sub-branch, it is much easier to spot missing or incorrect keywords.

As shown previously in Figure 9.1, “Tag cloud”, tags are displayed in relative font sizes according to their frequency of use. In other words, if one tag has been used in more objects than another tag, the first tag will be displayed in a bigger font. Click on a link for a particular tag to bring up a list of the content objects that have that tag:

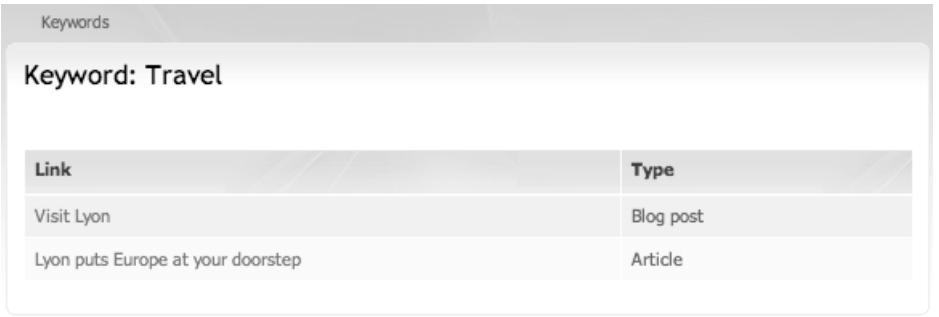


Figure 9.3. Keyword list

This list of objects can be used to access all content that shares the selected keyword by simply clicking on the names. A similar page is generated when you submit a search query, as described in Chapter 11, *Search engines and finding content*. The difference is that a search result matches the search term(s) against all published content, whereas the tag cloud only contains references to objects that have been explicitly tagged.

9.2. Blogging

The term "blog" is short for "web log" and refers to a collection of posts displayed in descending order by date (reverse chronological order). *Blogs* provide visitors with a personal and informal information channel from blog authors and the ability to comment and interact on posts. The screenshot below shows the typical blog features:

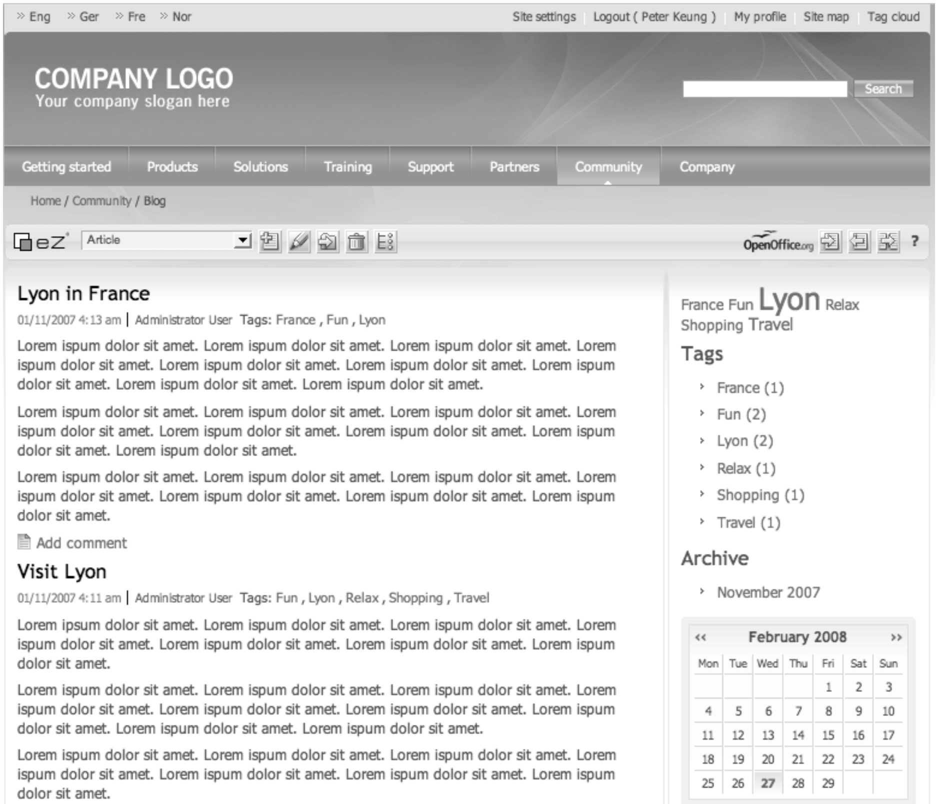


Figure 9.4. Blog page

As shown above, the default eZ Publish blog page has a main area containing blog posts, and a sidebar on the right with additional information such as tags and an archive.

Blogs have been used in many ways, such as online diaries, a channel for company commentary about an industry, or simply as an easy setup for employees or even site visitors to share information. Visitors can also be given the opportunity to have their own blogs and blog posts can be written by multiple authors.

9.2.1. Blog-related content classes

Blog-related classes include the `Blog`, `Blog post` and `Comment` classes. Whereas blogs are containers, blog posts and comments hold the actual content. Blogs are commonly set up by content managers, while blog posts can be written by a multitude of users. Blogs and blog posts are created in the same way that you create other content. For example, in the Website Interface, select "Blog post" from the dropdown list on the **Website Toolbar**, then click the **Create** button. Comments are submitted via the front-end of the site by visitors.

9.2.1.1. Blog class

The `Blog` class is a container class like the `Folder` class, and objects of this class are typically shown as in Figure 9.4, “Blog page”. Blogs are intended to hold objects of the `Blog post` class, in the same manner as an image gallery holds images.

By default, the `Blog` class consists of three attributes: "Name" ("Text line" datatype), "Description" ("XML block" datatype), and "Tags" ("Keywords" datatype). All of these datatypes have been previously described; see Chapter 1, *Introduction to Content Management Concepts* and Section 9.1.1, “Keywords datatype”.

Tip

You can have multiple blogs on a site by creating multiple objects of the `Blog` class (usually within a folder or frontpage) and populating each blog with blog posts.

9.2.1.1.1. Blog page sidebar

The sidebar shown in Figure 9.4, “Blog page” is automatically generated by the template system based on the contents of "Tags" attributes within the blog (in the blog itself, blog posts and comments) and the publication timestamps of the blog posts.

At the top of the sidebar is a tag cloud (see Section 9.1.2, “From keywords to tag clouds”) originating from the blog object. In other words, it is a blog-specific subset of the site-wide tag cloud. Beneath the cloud is a listing of the same tags as in the cloud, but showing, in parentheses, the number of posts that use each tag. Clicking on the name of a tag leads to a list of blog posts that have that tag.

At the bottom of the sidebar is a list of each month in which there were posts, followed by a visual calendar, which displays links to specific days when blog posts were made. Clicking on a month filters the list of blog posts to the specified month, while clicking on a day filters the list of blog posts by the specified day.

9.2.1.2. Blog post class

The `Blog post` class is used to create the individual entries in a blog. It has many of the same attributes as the `Article` class, and you may think of blog posts as simplified articles. The difference is that a blog post lacks a short title, summary and specific author and image input fields. By omitting the "Author" attribute, your blog posts cannot be compromised by someone entering false author information. Instead, blog posts are shown as authored by the user that created the object. Images can be used, but only within a post's body. As with articles, blog posts can have comments as sub-items.

Tip

On the front-end of a site, between the title and body of each blog post are a timestamp, the name of the author and a list of tags. Clicking one of the tags will reload the blog page showing only the posts that have this particular tag. It is the same as clicking one of the entries in the tag list in the sidebar.

The following list summarizes the attributes of the `Blog post` class, with their datatypes in parentheses:

- *Title* ("Text line")
- *Body* ("XML block"): contains the main content for the post. You can embed images, link to other objects and websites, and use formatting features such as tables, bold and italic text spans, custom tags and so on.
- *Publication date* ("Date and time"; see Section 7.2.1, “Date and time datatype”): the default value corresponds to the draft's Created timestamp. This attribute can be used with the delayed publishing feature (see Chapter 7, *Delayed publishing*).
- *Unpublish date* ("Date and time"): blank by default, this attribute can be used to schedule automatic object deletion with the workflow system. For more information, see Section 7.2.1.1, “Unpublishing content”.
- *Tags* ("Keywords"; see Section 9.1.1, “Keywords datatype”): These tags make up the blog tag cloud and other blog tag displays as illustrated in Figure 9.4, “Blog page”.
- *Enable comments* ("Checkbox"): if this is marked, users are allowed to submit comments about the blog post.

9.2.1.3. Comment class

The `Comment` class was covered in *eZ Publish Content Management Basics*, thus we will only do a quick review here, relevant to blogs.

Visitors usually enter a title ("Text line" datatype) and a message ("Text block" datatype) for comments, through the front-end of a site. Note in particular that neither of these attributes allow rich text formatting. Some sites do not allow anonymous users to add comments, in order to reduce the risk of having spam or other undesirable comments.

Comments are normally stored as sub-items of the piece of content to which they were added. For example, when you add a comment to a blog post, it is stored beneath the blog post node. In other words, comments should not be seen as isolated pieces of content.

By default, comments are displayed below the body of the content they apply to, ordered by publication date with the newest comments at the top. This way, visitors can publicly see all of the comments on a post.

A new comment is generated by clicking the **New comment** button at the bottom of a blog post (and below previous comments if there are any), then filling in the attributes for the comment. If the system is configured to require that people must be logged in to post comments (and you are not currently logged in), the page will read "Log in or create a user account to comment."

Comments include author information, but this can only be entered explicitly if the current user is anonymous. For logged-in users, the author attribute is automatically set to the user's name.

9.2.2. Enhancing your blogs with other eZ Publish features

The built-in RSS feature, as well as the permission and collaboration systems, enable you to enhance your blogs and to have more control over user-contributed content.

9.2.2.1. Multiple blogs, multiple authors

By using the permission system (see Chapter 5), you can let different users create, edit, or view blog posts under the different blogs. This requires users to log in with their personal user accounts. For example, you can have a blog where only the CEO writes posts, visible to all site visitors. Another blog would allow members of a certain user group to write and edit their own posts, visible only to members of that same user group.

9.2.2.2. Editorial approval of blog contents

You can use the collaboration system (described in Chapter 8) to require posts or comments to be approved by certain users before they are published on the site. This is useful to add editorial control or fact-checking to the blog post publishing process. Comments can be screened for spam or offensive comments before being publicly viewable.

This is a trade-off, as you lose the ability for visitors to instantly read each other's comments (when the comments are posted on the site as soon as they are created).

9.2.2.3. RSS feeds

An *RSS feed* is a structured XML file containing the content from a particular part of a site. RSS feeds follow well-known web standards, making it easy to exchange information across sites. Other sites can grab the content from your RSS feed to automatically display that content on their site. Then, whenever your content is updated (when you edit or create an object), the display of your content on their site is also updated.

You can do the same on your site by importing RSS feeds from other sites. Also, with RSS readers, site visitors can get automatic updates whenever content on an RSS feed has been updated. Using the built-in RSS feature (which is not restricted to blogs), you can create RSS feeds for specific blogs, or even combine multiple blogs into one feed. The exact procedure for importing and exporting RSS feeds is beyond the scope of this book. Consult your site administrator for more information.

9.3. Summary

"Web 2.0" refers to interactive website features that enable user participation. "Keywords" attributes can be used to generate tag clouds to show content relevance. Blogs can serve as online diaries or as channel to share company information, and also provide users a way to share thoughts and ideas.

A keyword list resembles a search result page in that you can access all objects that have a common word. The difference is that while a search result matches the search term(s) to any of the published content, a tag cloud only contains references to objects that have been explicitly tagged.

Blogs can be used in conjunction with other features and sub-systems of eZ Publish such as access control, delayed publishing, the collaboration system, and RSS feeds.

Chapter 10. Structuring content

A good content structure is one of the ways to achieve a compelling website. This chapter discusses some structuring principles and explains some practical tools in the Administration Interface that help you to organize content.

Some content managers and webmasters rarely get involved with defining the content structure, while others are responsible for making structural decisions from the very beginning. Consequently, this chapter will have varying levels of relevance, depending on your particular responsibilities.

Recall that:

- A *module* offers an interface for web-based interaction, providing access to eZ Publish core functionality. Each module has a set of *views* that enable you to reach the functions provided by the module. For example, the "edit" view of the "content" module is used for functions such as creating, editing and translating content.

- Node visibility refers to whether or not some published content can be viewed on the front-end of a site. Hiding a node will render the target node and the entire subtree below it invisible. The three visibility statuses are "visible", "hidden" and "hidden by superior". You can alter the visibility status of a node from the pop-up menu or the **Locations** window. The "hidden" status can also be set before the object is initially published.

In this chapter, you will find information about the following:

- How sitemaps can be used to get an overview of your content structure

- How the node swapping feature can be used to change the object a node encapsulates while preserving the structure below it

- How to create and maintain a good content structure

The sections in this chapter can be read separately and in any order, depending on which topics interest you.

10.1. Viewing a sitemap

A *sitemap* is a visual representation of your site content, or more specifically, a visual representation of the node tree structure. It is quite similar to a table of contents in a book. In eZ Publish, the sitemap is displayed and behaves slightly differently depending on whether it is accessed through the front-end of a site or through the Administration Interface. The Administration Interface sitemap enables you to access more functionality for

the listed nodes. In both cases, it displays two levels of depth, as defined in the respective templates. It is possible to override this limitation by creating custom templates.

The following screenshot shows the sitemap as it is displayed to site visitors, accessed by clicking the **Site map** link in the top right of the front-end site.

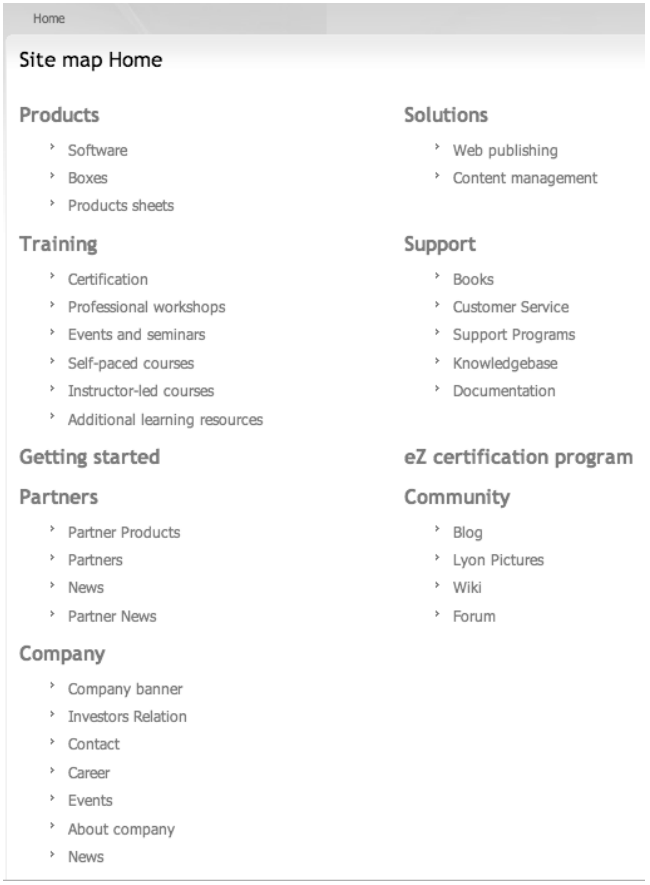


Figure 10.1. Front-end sitemap

It lists the names of all nodes on the first level of the Content branch, as well as their direct sub-items. In other words, the front-end sitemap provides you with a two-level structure of the nodes within the Content branch. Note that only nodes accessible to the current user are included. For example, in the above screenshot, the "Partners" frontpage and its children are not shown since they are not available to anonymous users. The nodes are listed in the same order as defined by the sort order within the **Content structure** tab.

By default, the URL of the front-end sitemap is the URL to the index page of one of your siteaccesses, appended by "content/view/sitemap/2". As the string reveals, "sitemap" is

a special view of the "content" module (see Section 1.3, “Modules and views”). It displays nodes below the Content top-level node, which has a Node ID of "2" (see Section 4.1.3, “Top-level nodes”). You can force the system to use another node as the starting point for the sitemap by changing the last number in the URL to a valid Node ID. For example, replacing "2" with "86" in our example URL yields a two-level sitemap starting from the "Training" frontpage node:

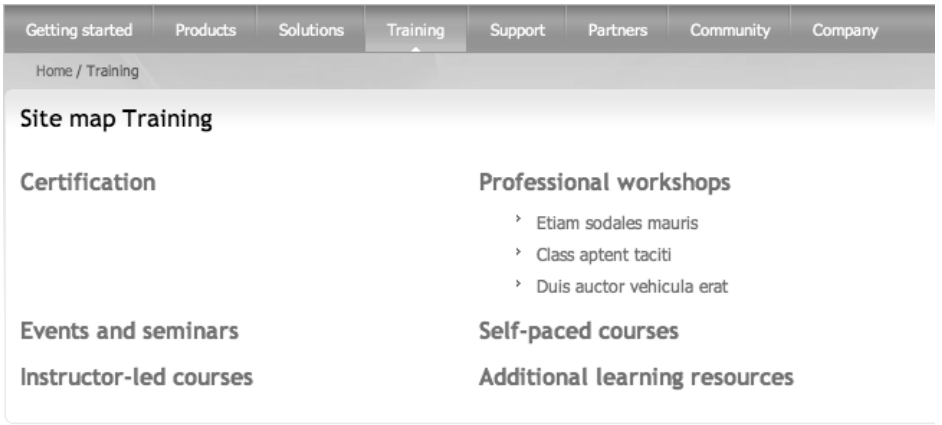


Figure 10.2. Front-end sitemap - custom starting point

Clicking one of the names in the sitemap opens the corresponding page as if you had navigated to it through the menus.

The Administration Interface sitemap is similar to the front-end sitemap, but has a few more features:

- You can open the sitemap from the context-sensitive pop-up menu from any selected node. The selected node will then be the starting point of the sitemap. In other words, you do not need to manipulate the URL to bring up sitemaps with different starting points.

  Choose a node for the starting point and bring up the pop-up menu for it. Then, click the "View index" item under the "Advanced" sub-menu:

Figure 10.3. Accessing the sitemap from the pop-up menu

- The sitemap will contain hidden nodes as long as their display is enabled for the Administration Interface (see Section 6.4, “Requirements for working with hidden content”).

- An icon representing the encapsulated object's class is displayed to the left of the node name.

The screenshot below shows the sitemap in the Administration Interface with the Content top-level node as the starting point.

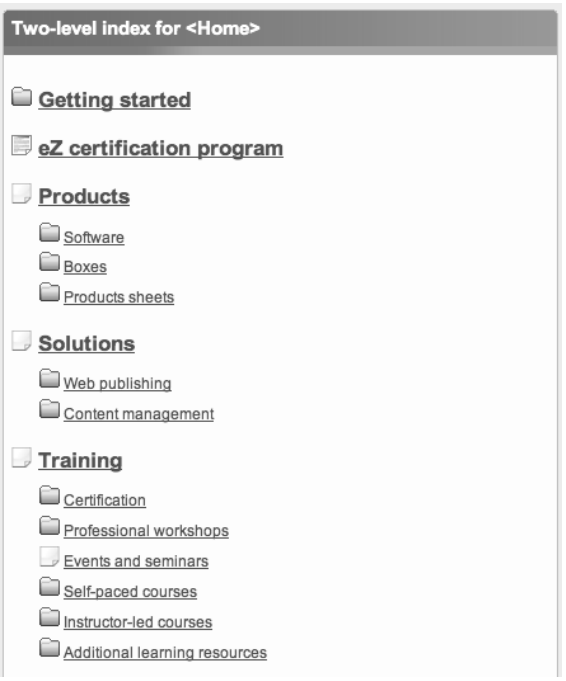


Figure 10.4. Administration Interface sitemap

Tip

Opening a sitemap for the Media top-level node can give you a good overview of the media library. This is especially useful if your left menu only shows the most significant container nodes.

10.2. Swapping nodes

The most obvious way of reorganizing content is to *move* a node or subtree of nodes. Sometimes, however, you might want to change the type of a node while preserving the structure below it. For example, you might want to replace a folder with a frontpage or documentation page, or maybe a custom-made folder (with some customized attributes) in order to better suit the site. In such cases, the best solution is to use *node swapping*.

This feature is particularly useful for changing a top-level node. For example, the Content top-level node used to encapsulate a Folder object by default, but it now encapsulates a Frontpage object in sites using the Website Interface 1.2 and higher. If your site has been upgraded and you want to take advantage of the features of objects of the `Frontpage` class, swapping is the only way you can alter the node type.

The following procedure explains how to perform node swapping. Note that it is very thorough for illustrative purposes. On a real site, the operation usually consists of only 3 steps: creating the replacement object, performing the actual swapping through the pop-up menu, and archiving or deleting the replaced object.

Consider the following scenario. When setting up the community area of your site, someone incorrectly used a Folder object as the container for your individual Forum objects, instead of using a Forums object. Because of this, the starting page for your discussion forums does not display as intended. It will most likely use a Folder template rather than a Forums template.

Tip

Note that it is not required that the container for objects of the `Forum` class is a Forums object. If your site has only a single forum, it could very well be located within a folder. However, for sites with multiple forums, it is recommended to use the special-purpose **Forums** class for the container in order to use the proper template.

For example, the **Last reply** column would show all of the folder's sub-items instead of just the latest reply:

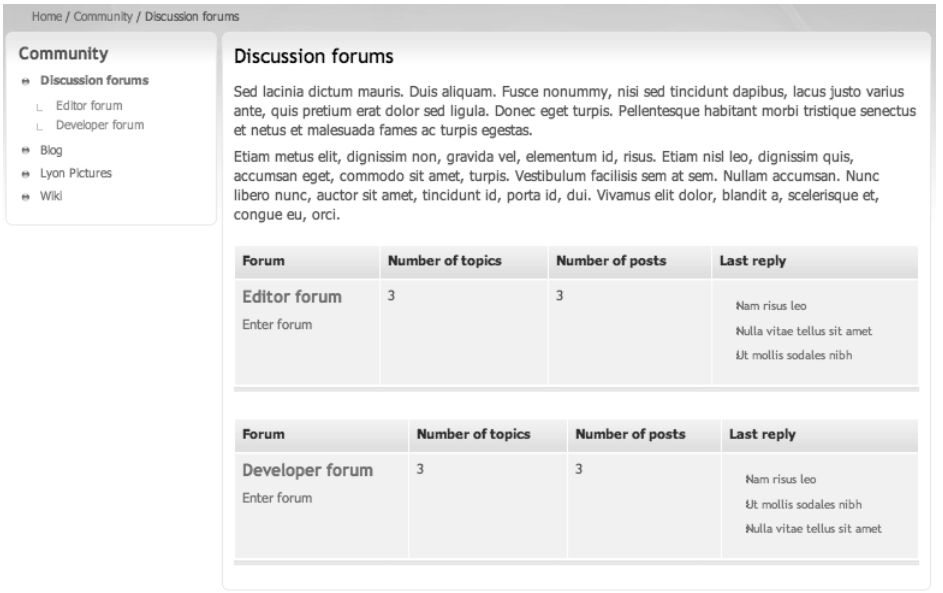


Figure 10.5. Forums page before the swap

This would be particularly troublesome if there are many sub-items.

Therefore, that node should encapsulate a Forums object. As preparation for the swap, create the replacement object somewhere in your content structure (in our example, we create it below the "Community" frontpage).

Tip

If you do not want anyone to see the replacement object while you are working on it, alter its node visibility (mark it as hidden before clicking the **Send for publishing** button, as described in Chapter 6, *Node visibility*). Note that your user account must have access to the "hide" function of the "content" module.

The following screenshot shows the initial state of this scenario, as illustrated by the tree menu:

Figure 10.6. Folder with forum-related sub-items before the swap

Once you have created the Forums object, follow these steps:

1. Locate the node you want to replace within the left tree menu or the **Sub items** window. In this case, this is the "Discussion forums" folder under the "Community" frontpage.

2. Open the pop-up menu by clicking on the icon for the "Discussion forums" folder, then select "Swap with another node" under the "Advanced" sub-menu:

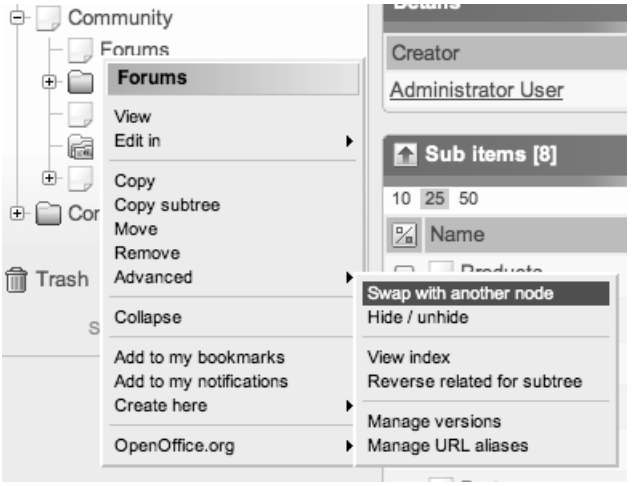


Figure 10.7. Pop-up menu - Swap with another node

3. This will open the **Browse** interface:

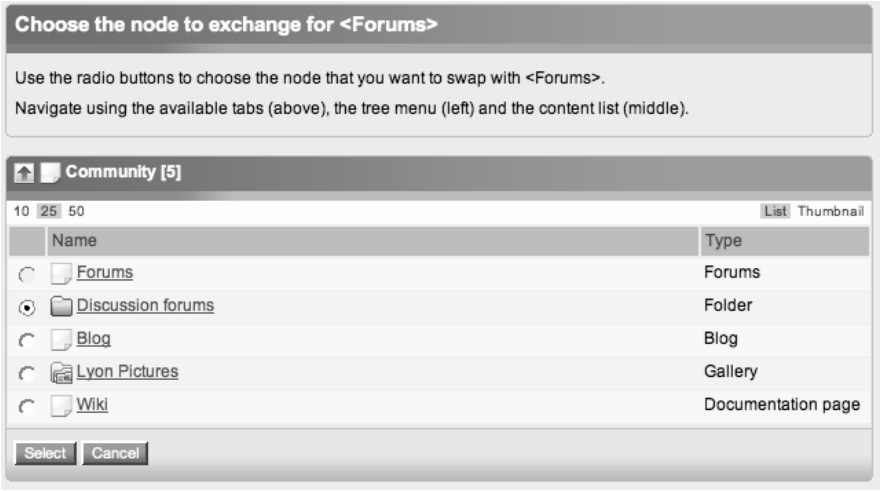


Figure 10.8. Browse interface for swapping two nodes

Use the **Browse** interface to locate the node with which to swap the current node by clicking the corresponding radio button. In this case, select "Forums".

4. Click the **Select** button to execute the swap. The result is shown in the following screenshots.

5. Perform some clean-up operations: Edit the replacement object if needed, and archive or remove the old object (which should not have any sub-items). Update node visibility if you used the hide feature before performing the swap.

Figure 10.9. Forums page after the swap

Compare the above screenshot with Figure 10.5, “Forums page before the swap” to see the changed display of the forums page. The following screenshot of the left menu shows the change from Figure 10.6, “Folder with forum-related sub-items before the swap”. Notice the new name and icon for the parent node of the "Editor forum" and "Developer forum" objects:

Figure 10.10. Forums container with forum-related sub-items after the swap

In order to understand what really happens here, look at the Node and Object IDs. You can find this information by enabling the **Details** window. The following sequence shows the **Details** window for both nodes, before and after the swap.

Figure 10.11. Node encapsulating Folder object before the swap

Figure 10.12. Node encapsulating Folder object after the swap

Figure 10.13. Node encapsulating Forums object before the swap

Figure 10.14. Node encapsulating Forums object after the swap

Let us start by establishing that we have two objects: one of the `Folder` class, and one of the `Forums` class. An object's ID is fixed for the entire time that the object exists. Here, the Folder object (named "Discussion forums") has an Object ID of "175", while the Forums object (named "Forums") has an Object ID of "180".

Second, we have two nodes, with Node IDs "167" and "172". Recall that a node is simply a location within the tree; a node references exactly one content object; and it has a reference to its superior node in the tree. The latter is important for preserving the sub-items after the swap. Also, keep in mind that the name of a node is pulled from the object it encapsulates.

Observe the following changes between the screenshots:

- The Folder object (Object ID "175") is first referenced by node "167", and later by node "172". This transition is seen from Figure 10.11, “Node encapsulating Folder object before the swap” to Figure 10.12, “Node encapsulating Folder object after the swap”.

- The Forums object (Object ID "180") is first referenced by node "172", and later by node "167". This transition is seen from Figure 10.13, “Node encapsulating Forums object before the swap” to Figure 10.14, “Node encapsulating Forums object after the swap”.

- The node with ID "167" first references a Folder object, and later a Forums object. This transition is seen from Figure 10.11, “Node encapsulating Folder object before the swap” to Figure 10.14, “Node encapsulating Forums object after the swap”.

- The node with ID "172" first references a Forums object, and later a Folder object. This transition is seen from Figure 10.13, “Node encapsulating Forums object before the swap” to Figure 10.12, “Node encapsulating Folder object after the swap”.

With node swapping, two nodes mutually exchange their encapsulated objects. Literally, you take an object out of its node and put it into another one. Technically, the Object ID

value kept in the node is simply updated with a new value. The illustrations below show what happens from a content node tree perspective:

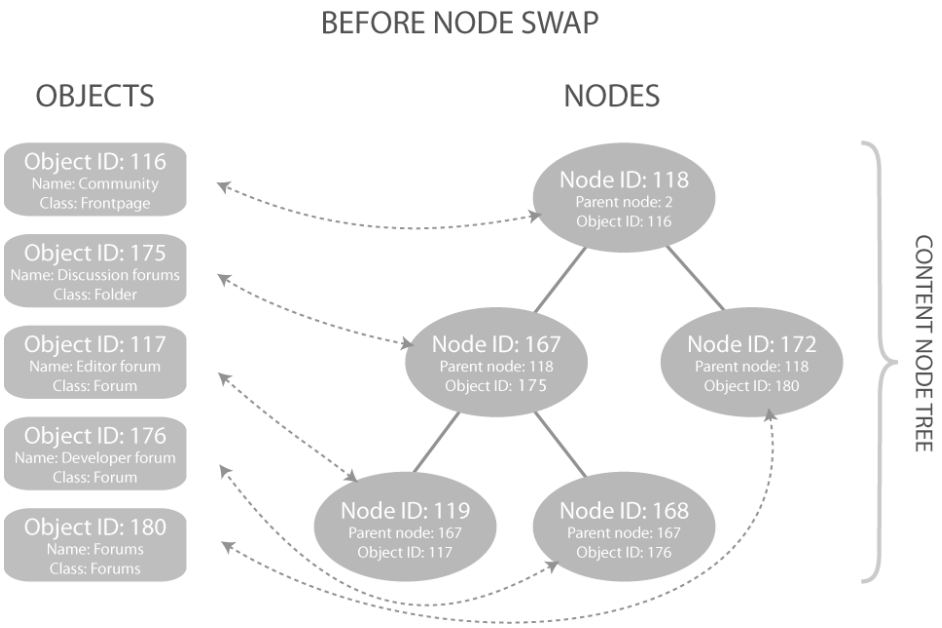


Figure 10.15. Object-node relationships before the swap

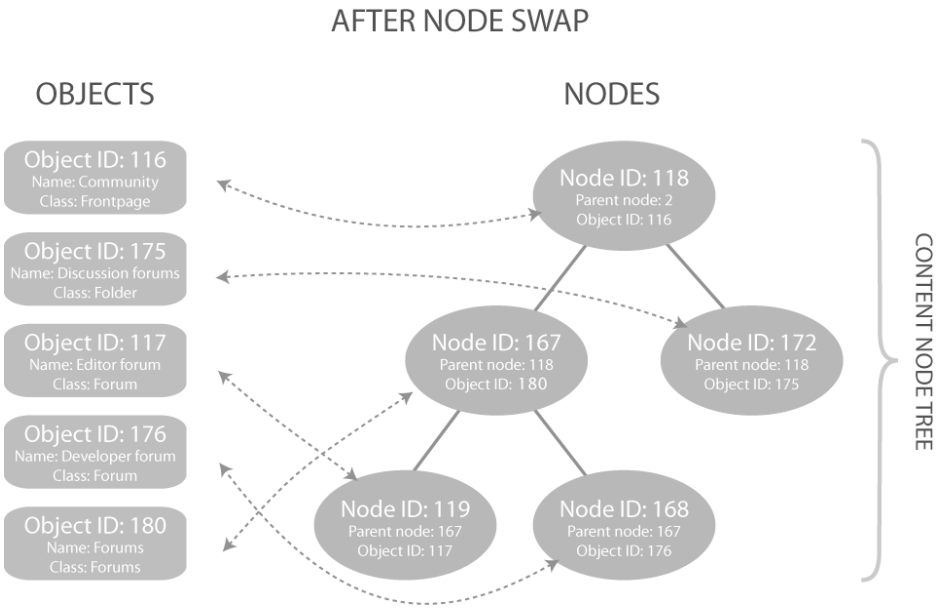


Figure 10.16. Object-node relationships after the swap

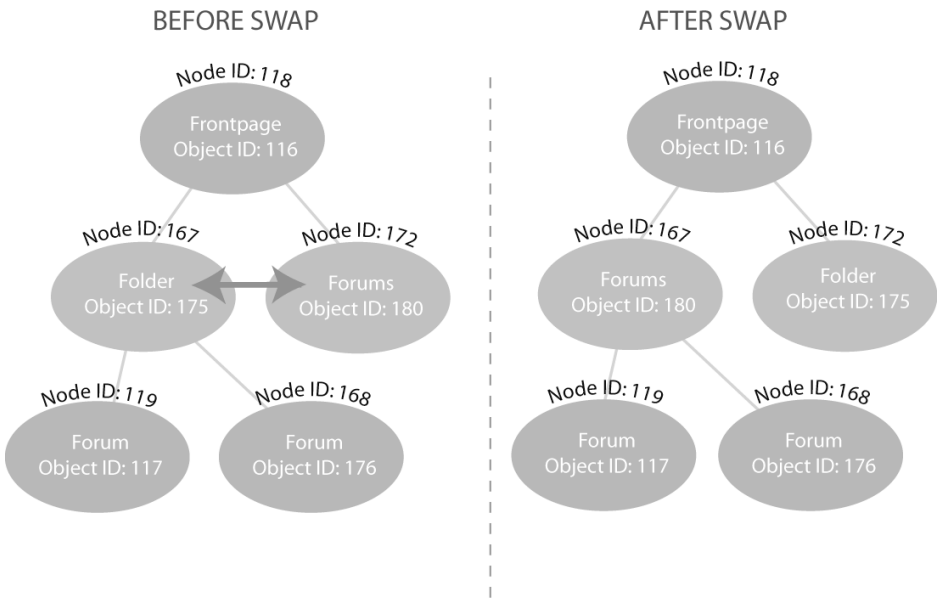


Figure 10.17. Node tree and encapsulated objects before and after the swap

Note the position within the tree and the children of the nodes being swapped. In this case, only node "167" had sub-items. This is still true after the swap. The reason is that the "Editor forum" and "Developer forum" objects had node "167" as their parent node, and this value is not affected by the swap. Likewise, node "167", despite the fact that it now references another object, is still a direct descendant of the "Community" frontpage (node "118").

10.3. Principles of information organization

In this section we provide a few tips on how you can enhance your site by better organizing its content structure. This is by no means a complete guide, and interested readers should look for other books on the topic of information architecture.

10.3.1. Organize content according to user needs

Have your users and their needs in mind when setting up and maintaining a structure. The more you know about them, the better. For example, if visitors come to your site strictly to buy products, then you should probably have a dedicated webshop (see Chapter 15, *Webshop*) instead of products buried within other content.

Also keep in mind that you may have multiple visitor groups (not to be confused with objects of the `User group` class), each one with separate needs. It might be best to divide your site into group-specific areas, such as for communities, partners and customer support. Take advantage of the cross-publishing feature to have content appear at multiple locations, and build special-purpose structures for each user group's area. For common areas, think about how each group uses the content and try to come up with a solution that works for all of the groups.

10.3.2. Be familiar with your content node tree

If you do not have an awareness of the current state of your node tree, additions, removals and restructuring may gradually degrade the quality of your structure. For example, you might unnecessarily have multiple news folders, or content might not have the right context for its location. Use the sitemap feature to get a quick overview of your content node tree.

10.3.3. Hide content you do not want to have on display

Acknowledge that content may have a limited lifespan and interest to your visitors. Delete or move old stuff to an archive area or use the hide feature (see Chapter 6, *Node visibility*) in order to ensure that you provide users with material with high relevance.

10.3.4. Facilitate clear and easy navigation

Quite often, and in particular with Website Interface sites, the main menu reflects the items on the first level of the Content branch. Use priority-level sorting to take full control over how the items are sorted (as was explained in the *eZ Publish Content Management Basics* book). Use good, meaningful and descriptive names for your objects so users get an idea of what each of the direct sub-items (and their sub-nodes) of the Content top-level node contains. If you decide that you can improve on existing titles, simply edit the object and publish a new version.

10.3.5. Use metadata and keywords

Make use of (but do not overuse) the "Tags" / "Keywords" attributes (see Section 9.1.1, “Keywords datatype”). Use them to emphasize the most important terms in your content, to ensure that the search engine will return a match for a specific word to that object, and to build the foundation for an accurate tag cloud. For example, adding "eZ" as a keyword for content on `http://ez.no` about or written by eZ makes little sense.

10.3.6. Use appropriate classes for your objects

Take advantage of the various container classes that are more or less tailored toward a specific use. Each class comes with its own set of templates dictating layout. By using folders for almost everything, you miss out on a lot of possibilities. For example, frontpages enable you to better customize the presentation and are not limited to a plain listing of sub-items within a folder. On the other hand, do not use a particular content class just because it has more features. It might not add value to your content. A frontpage may be unnecessary for your Company section if all you will ever have there is a short article and a contact form.

If you change your mind about a container object's class, consider using the node swapping feature (see Section 10.2, “Swapping nodes”).

10.4. Summary

A sitemap is a two-level visual representation of the node tree structure. It is commonly used for overview and navigation purposes.

Any node, including top-level nodes, can be swapped with another node. It is commonly used to change the object that a node encapsulates while preserving the structure below it. The operation is initiated from the context-sensitive pop-up menu in the Administration Interface.

When structuring your content tree, keep the needs of your users in mind and make use of built-in eZ Publish features (such as hiding, swapping, sitemaps and tags) to optimize your site's usefulness.

Chapter 11. Search engines and finding content

Website content is difficult to work with if you cannot find what you are looking for. Previously, we looked at how good content structure can facilitate the user experience (see Section 10.3, “Principles of information organization”). This chapter deals with the interfaces and functionality of the search feature.

Until recently, the only publicly available alternative for integrated content searching on eZ Publish sites was the built-in search engine. For eZ Publish version 3.10 and higher, a new extension called *eZ Find* is available. Among other things, eZ Find provides more search term options and relevancy ranking of results. Here, we cover both the built-in and eZ Find search engines.

It is useful for content managers to be aware of the search options in order to use them to assist their daily work.

Recall that:

- An eZ Publish *INI file* is a configuration file that controls the behavior of a specific part of the system. See Section 3.5, “The configuration model”. The main configuration file is `site.ini`. You can check the value of a particular setting within the **Settings view** interface as described in Section 3.5.4.1, “Viewing the settings of a configuration file”.

In this chapter, you will find information about the following:

- The principles of the built-in search engine, the search statistics and how the **Advanced search** interface works
- The advantages of the eZ Find search extension and how it compares to the built-in search engine
- Available search term options and search filtering

We recommend that you read the material in the order it is written. In particular, we cover the fundamentals of a search engine in Section 11.1, “Built-in search engine”. The description of eZ Find is based on this and shows how it differs from the standard / built-in search engine. For more detailed information about eZ Find, refer to the online manuals at http://ez.no/doc/extensions/ez_find.

Also, some readers may find parts of Section 11.1, “Built-in search engine” and Section 11.2, “eZ Find search extension” a bit too technical. We have isolated the most advanced information into in-depth blocks for interested readers.

11.1. Built-in search engine

A *search engine* is a system for information retrieval on a computer system. In other words, its purpose is to help you to find stuff. To illustrate how important a well-functioning search engine can be, think of how the Google search engine affects the daily web experience for many people. Within the eZ Publish context, the search engine finds the requested content on your site, helping visitors, editors and administrators to browse and work more efficiently and effectively.

Typically, user interaction with a search engine goes through a special-purpose interface where you submit a search query (consisting of search terms and / or phrases) and receive the results. In other words, this is a request - response scenario. In order to shorten the response time, many search engines build an index over the stored information. The following section explains how this is done in eZ Publish.

11.1.1. Content indexing

In eZ Publish, the search engine only looks for matches to the search query within the *search index*. This implies that at some time, the engine has to go through all the published content of your website and build such a index. Before we look deeper into how this is done, let us quickly explore what an index is.

An index is best illustrated by a technical book, like the one you are currently reading. Usually you can find out what you need to know by reading the entire book from beginning to end. However, this is not very efficient if you are looking for specific information on, for example, workflows. The shorter route is to turn to the book's index and look for the term "workflow". There, you would find one or more page references, depending on whether there is any material on that topic.

Warning

Do not confuse an index with a table of contents in a book or a sitemap on a website.

In eZ Publish, you do not have to build the index yourself; it is taken care of by the search engine. All you have to do is ensure that what you want to be searchable is in fact indexed by the engine.

Recall that a content class is the definition of a data structure, and that it consists of attributes of certain datatypes. Each class attribute has a set of generic controls, one of these being "searchable". By enabling or disabling the **Searchable** checkbox in the **Class**

edit interface (see Section 3.2, “Viewing and editing content classes”), it is possible to specify which attributes are indexed by the search engine. Setting the "searchable" control is typically done during the development phase of a site, but it can also be changed after a site has gone live.

Tip

To find out if a specific attribute is searchable, go to the **Setup** tab, click the **Classes** link in the left menu, then bring up a view of the desired class by clicking on the class name (commonly within the Content group). Alternatively, select "View class" from the developer pop-up menu, accessed by clicking the class icon in the **Preview** window title bar when viewing a content object. In the **Flags** column on the right for the attribute in question, you will see an entry that says "Is searchable" or "Is not searchable".

Indexing is supported for many but not all of the built-in datatypes. Attributes of datatypes for numbers, such as prices, VAT percentages and dates, are usually not searchable. Refer to http://ez.no/doc/ez_publish/technical_manual for exact details.

In eZ Publish, indexing happens silently when a content object is published. What happens is that the contents of attributes that are marked as searchable are examined by the search engine and relevant metadata is added to the search index. For the built-in search engine, the index is kept in the database, and it is updated every time a new version of a content object is published. This is the default behavior. Alternatively, indexing can be postponed to be processed later, making the publishing process faster. Delayed search indexing is described in the in-depth block below.

In Depth: Delayed search indexing

It might take a few seconds from when you click the **Send for publishing** button in edit mode until the page reloads and the new content becomes available. The *delayed search indexing* feature enables indexing to be handled at a later time, decoupled from the publishing process. This will speed up publishing a bit; therefore, applying delayed search indexing is recommended for performance reasons. The drawback is that content is not immediately searchable once published.

The requirements for delayed search indexing include enabling the feature under the "[SearchSettings]" block in the `site.ini` configuration file, and having the `indexcontent.php` cronjob script (see the in-depth section in Section 3.7.2, “Workflow-related requirements”) run periodically. Consult your site administrator about setting this up.

The built-in search engine also supports the indexing of binary files such as PDF and Microsoft Word documents, in addition to indexing text-based attributes. The in-depth block below provides an introduction to binary file indexing.

In Depth: Binary file indexing

By default, eZ Publish is only capable of indexing searchable attributes and plain text / ASCII files. The *binary file indexing* feature makes it possible to also include contents of uploaded files like PDFs or word processor documents in the information searched by the default engine. Note that these files are usually stored in attributes of the "File" datatype.

This feature requires external programs to provide keywords and / or convert the binary file to plain text recognizable by the built-in search engine. This is set up by your site administrator.

11.1.2. Features of the standard search engine

The built-in search engine provides basic search functionality and some advanced search term options and filtering. The list below highlights some of its important properties and limitations. You will learn more about these throughout this chapter.

- Search for individual terms, for an exact phrase, or for multiple terms. See Section 11.4, “Available search term options”.
- Limit a search by applying filters (advanced). See Section 11.6, “Advanced search filtering”.
- Wildcard searching (disabled by default). See Section 11.7, “Wildcard searching”.
- Possible to create custom templates for search interfaces and result pages.
- Requiring and excluding terms not supported.
- Search index stored in database.

The standard eZ Publish search engine works out of the box on all installations. There is no setup required to use this search engine, as it is already enabled in the configuration files and the technical requirements are a subset of what your eZ Publish installation needs. If you are going to use delayed search indexing, binary file indexing or wildcard searches, some configuration and additional programs are needed. Refer to the relevant descriptions in this chapter for more information.

11.2. eZ Find search extension

eZ Find is a freely available search extension (also called "ezfind" for system reference) for eZ Publish sites that provides enhanced functionality and better search results compared to the standard search engine. It has more available search term options, enabling you to better customize your queries. Also, the result page is more sophisticated, including keyword highlighting and relevance ranking. At the time of publication, the latest version of eZ Find is 1.0.

The key technical difference is that eZ Find makes use of an external Open Source search server, *Solr*, which is introduced in the in-depth block below.

In Depth: Solr, behind the scenes of eZ Find

The Solr enterprise search server (see http://lucene.apache.org/solr/) is an Open Source project based on the Apache Lucene Java search library. This library is what makes the new and enhanced functionality of eZ Find possible.

The search index is handled by Solr in a special-purpose file that enables the index to be searched and updated at the same time, thus increasing performance. Note in particular that the more the search engine is used, the faster it becomes. The reason for this is Solr's ability to learn from queries.

Only a subset of Solr's capabilities are used in eZ Find 1.0; therefore, the opportunity for new and improved functionality in future versions is promising.

11.2.1. Requirements for eZ Find

The following eZ Find requirements are usually handled by a site administrator.

- eZ Publish version 3.10 (with PHP 4.4.x) or eZ Publish version 4.x (with PHP 5). Note that there is a different eZ Find release for each of these two versions.
- Your web server must have access to the Java Runtime Environment (JRE), version 5 or higher. To download it and for more information, visit http://java.sun.com.
- eZ Find must be installed on your site as an extension (see Section 3.6, “Extension management”).
- The Solr back-end application must be running. Refer to the eZ Find manual at http://ez.no/doc/extensions/ez_find.
- The search index of your site has to be updated, by having the extension re-index all your existing content after you first install eZ Find.

11.2.2. Features of eZ Find

eZ Find has, with a couple of exceptions, the same features as the standard search engine (see Section 11.1.2, "Features of the standard search engine") plus some additional features, as listed below.

- Search for individual terms, for an exact phrase, and for multiple terms. See Section 11.4, "Available search term options".
- Limit a search by applying filters (advanced). See Section 11.6, "Advanced search filtering".
- Wildcard searching not supported.
- Possible to create custom templates for search interfaces and result pages.
- Requiring and excluding terms supported.
- Relevancy ranking and keyword highlighting in results.
- Possible to index and search multiple eZ Publish sites (limited to published content available to anonymous users).
- Search index stored in a special-purpose file, for better search performance.

11.3. Search interfaces

We now turn our attention to the interfaces for inputting search queries. By default, the basic **Search** interface is located in the top right of your site, both via the front-end of the site and in the Administration Interface. The **Advanced search** interface provides additional search term options and filtering.

11.3.1. Basic Search interface

The basic **Search** interface generally only consists of a text input field and a **Search** button:

Figure 11.1. Basic Search interface (front-end)

Within the Administration Interface, the **Search** interface also has radio buttons to toggle between searching either all content or searching just the node tree from the current location and below.

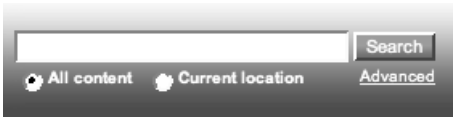


Figure 11.2. Basic Search interface (Administration Interface)

11.3.2. Advanced search interface

The **Advanced search** interface supports a wider range of search term options, and also enables you to filter the search by class, section and more. By default, it is accessed in the Administration Interface by clicking the **Advanced** link in the **Search** interface. In the front-end of the site you have to access it via the result page (see Section 11.8, "Search result pages") after submitting a query in the basic **Search** interface. The following illustrations show the **Advanced search** interface. Note that the front-end version only provides access to a subset of the Administration Interface options, as is described in the next section.

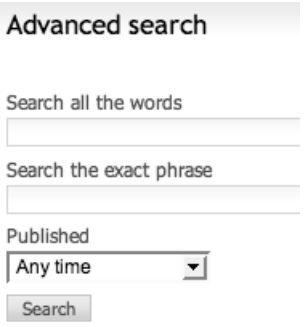


Figure 11.3. Advanced search interface (front-end)

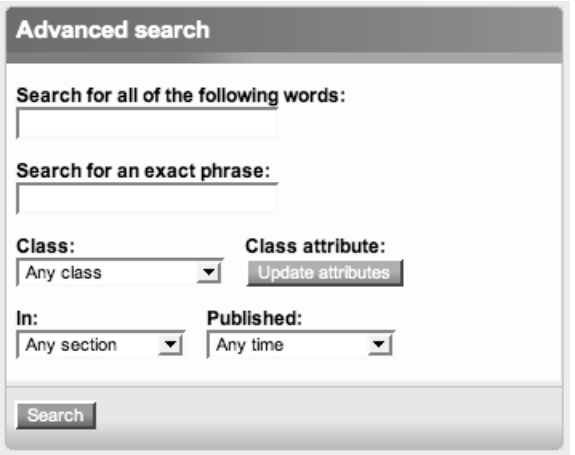


Figure 11.4. Advanced search interface (Administration Interface)

11.4. Available search term options

Five different search term options are available to at least one of the two search engines. eZ Find is generally more permissive. Four of the search term options are described next, while wildcard searching is currently available only in the built-in search engine and is described in Section 11.7, “Wildcard searching”.

All terms are case insensitive. Recall that the search engine only indexes the published versions of content.

11.4.1. Search for individual terms

This is the simplest search option, in which the query consists only of a single word. It is also known as a *simple query*. All search text input fields support this. Example individual term queries include "download", "support" and "Paris".

11.4.2. Search for an exact phrase

Searching for an exact phrase will only match a specific sequence of words. *Phrase queries* can be submitted in the basic **Search** interface with quotation marks surrounding the phrase. If you omit the quotation marks, the query will be treated as a multiple term query. Alternatively, use the **Search for an exact phrase** input field of the **Advanced search** interface (do not use quotation marks in that case).

Note that the order of the words is relevant. Consider the phrases "pen and paper" and "paper and pen". A search for one phrase will not return the same matches as the other.

11.4.3. Search for multiple terms

It is possible to search for multiple terms in the same query. You can also combine single terms with phrases into one query, such as 'Lyon "city center" development'. In other words, this search option allows for various combinations of the other options.

Multiple term queries can be submitted directly in the basic **Search** interface, or in the **Search for all of the following words** input field of the **Advanced search** interface for both search engines.

11.4.4. Excluding and requiring terms

This option is only supported by eZ Find. Inline logical operators like "AND" and "OR" are not supported. Rather, the term to be required or excluded should be prefixed by a plus or minus symbol. For example, "Lyon +France" requires that "France" is found in each search result item, while "Lyon -French" means that "French" must not appear in any search result item.

Tip

In the built-in search engine, you can use the **Search for all of the following words** field in the **Advanced search** interface to require all terms to be found in the search results.

11.5. Invalid search input

If you enter invalid input in one of the search fields, it is simply discarded. For example, if you enter inline logical operators using the built-in search engine, the query processed will ignore those. "France -Lyon" is thus treated as a multiple terms query in the built-in search engine. A search using this query would return all matches that have both the terms "France" and "Lyon", rather than all matches for "France" excluding those containing "Lyon", as was intended.

11.6. Advanced search filtering

This section explains the filters you can apply in the **Advanced search** interface to tweak or narrow a search. The filters can be combined and are represented by corresponding dropdown lists. In the front-end of a site, the only filter available is the Publication time filter.

11.6.1. Filter by content class

The *Class filter* limits the search to objects of a certain class. When a class is specified, you can further filter results by attribute, as described next.

11.6.2. Filter by class attribute

The *Class attribute filter* is used to search in a specific attribute, such as the "Title" attribute of articles, or the "Name" attribute of users. After you select a class in the **Class** dropdown list, click the **Update attributes** button to load the corresponding attribute list.

11.6.3. Filter by section

The *Section filter* limits the search to objects belonging to a certain section (see Chapter 4). For example you might want to search within the Restricted section, to separate the results from those in the Standard section.

Tip

By default, filtering searches by the Media section in the **Advanced search** interface is the same as setting the **Current location** radio button in the basic **Search** interface when the top folder of the **Media** tab is selected. However, the **Current location** radio button can be selected when you are navigating in any branch and depth of the node tree, and does not necessarily correspond to a specific section.

11.6.4. Filter by publication time

The *Publication time filter* narrows search results to objects that were published within a specific time period. As is the case for all searches, this applies to the object and the published version that is represented in the node tree. In other words, you can only search among versions that have the "Published" status. Drafts, pending items and archived versions are not searchable.

11.7. Wildcard searching

Wildcard searching is currently only available with the built-in search engine, and must be enabled in the configuration files. Consult your site administrator for information about setting this up.

By default, searches return only exact matches on terms. For example, if you search for the word "demo", the system will not return objects that contain words like "demolition"

or "demonstration". However, eZ Publish supports searches that include a wildcard, allowing for partial matches.

Warning

Wildcard searching requires a lot more processing power than exact-match searches. Some servers may not be powerful enough to support this kind of searching, or the response time might be so slow that it noticeably affects the user experience. Therefore, it is disabled by default, and should only be enabled by the site administrator if the server is sufficiently powerful.

When the wildcard search feature is enabled, the asterisk character acts as a wildcard - that is, the search will return both exact matches and matches that contain the letters before the asterisk and any combination of letters in the same word after the asterisk. For example, searching for "demo*" returns a list of objects that contain words starting with "demo", including "demo", "demonstration" and "demolition". Note that the asterisk wildcard can only be used at the end of a search term, not at the beginning or somewhere in between. For example, the following search queries are invalid: "*demo" and "demo*ion".

11.8. Search result pages

The *result page* displays differently depending on whether you are using the front-end or the Administration Interface, and depending on which of the search engines are in use (the built-in or eZ Find search engines). The information shown for each match is pulled from the content object. This might be the contents of the "Short description" attribute, the first paragraph of an article body, or simply the title or name of the object. The information displayed is determined by the templates. The following screenshot shows a search results page for the built-in search engine, listing a frontpage, a folder and an article:

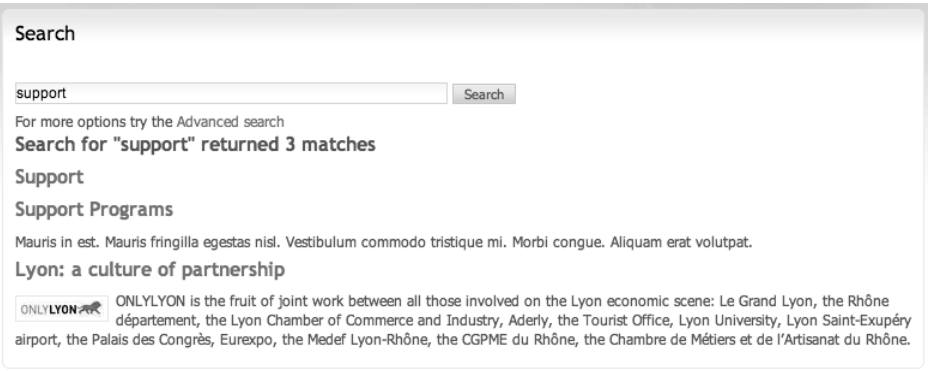


Figure 11.5. Search results - different display for various types of content

11.8.1. Search results with the built-in search engine

The following two screenshots show the result page generated by the built-in search engine:

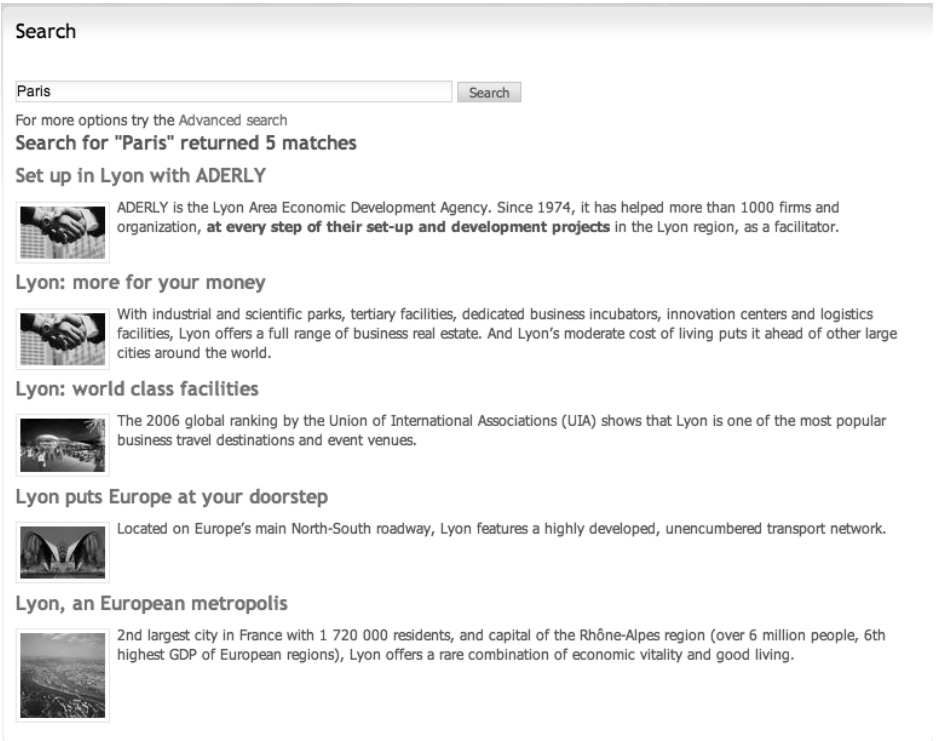


Figure 11.6. Search results - standard front-end page

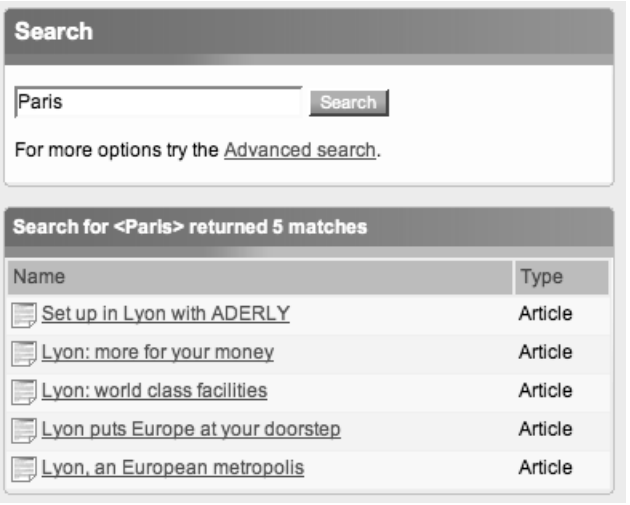


Figure 11.7. Search results - standard Administration Interface page

The two screenshots above show the result of searching for "paris" in the front-end of the site and the Administration Interface. Both result pages contain a duplicate of the **Search** interface, with the previous query entered, and a link to the **Advanced search** interface. The results of the search are listed below it. If no results were found, you will find some hints on the result page on how to better specify your search.

In both the front-end and the Administration Interface, the titles or names of the objects are listed as links. Clicking a link will open a full view of the target content, in the same way as if you had navigated the content structure and selected the node from a menu. In the front-end, a short text description is displayed along with each title, whereas the Administration Interface lists the objects' content classes. Depending on the type of content, an image might be included, as in Figure 11.6, “Search results - standard front-end page” above.

11.8.2. Search results with eZ Find

The following two screenshots show the result page generated by eZ Find:

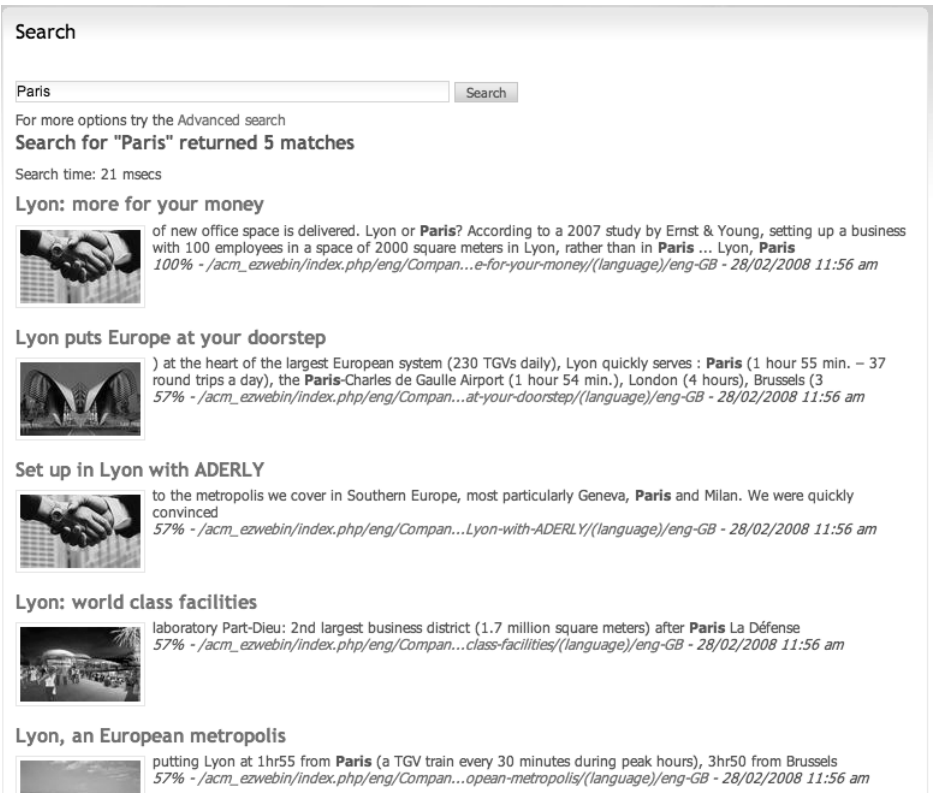


Figure 11.8. Search results - front-end eZ Find page

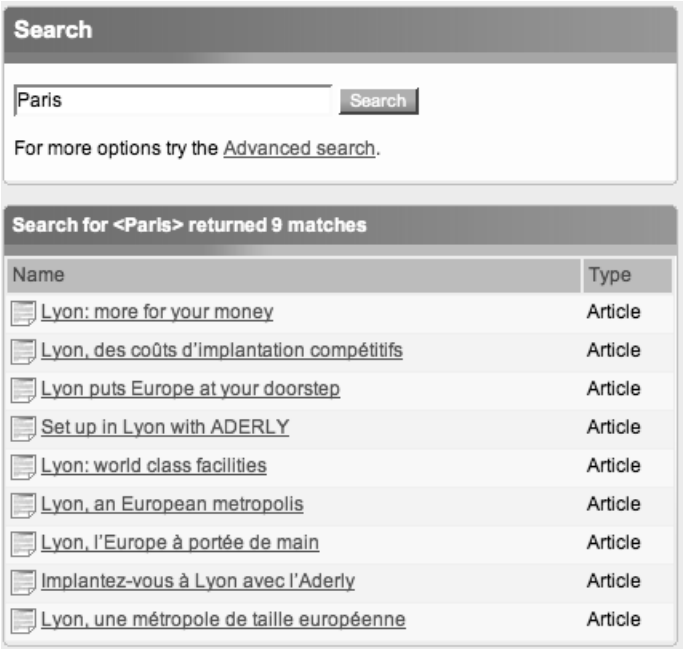


Figure 11.9. Search results - Administration Interface eZ Find page

The top of the front-end result page in eZ Find shows the amount of time taken to perform the search operation. For each result, the search term is highlighted in bold in a text extract from the object (this is often referred to as "keyword highlighting", and there is a percentage indicating the relevancy of the result. Relevancy ranking is based on advanced heuristics, which take into account factors such as the attribute(s) in which the search query matched and the languages used in translations in multilingual sites. For example, a French translation will never get 100% relevancy in a siteaccess where English is the primary translation.

The eZ Find result page for the Administration Interface is similar to that of the standard search engine, except that an object might appear multiple times if it has multiple translations that match the search query.

11.9. Search statistics

The *search statistics* show a list of words and phrases used in actual searches on your site, as well as the average number of returned results. These statistics help you to learn what topics are more popular than others; to gather information on which content you might want to make more easily accessible; and to identify common misspellings in search queries. The statistics are accessed by clicking the **Search statistics** link in the left menu of the **Setup** tab. You need an editor account with extra permissions (in addition

to the default permissions) or an administrator account for this. Example search statistics are shown below:

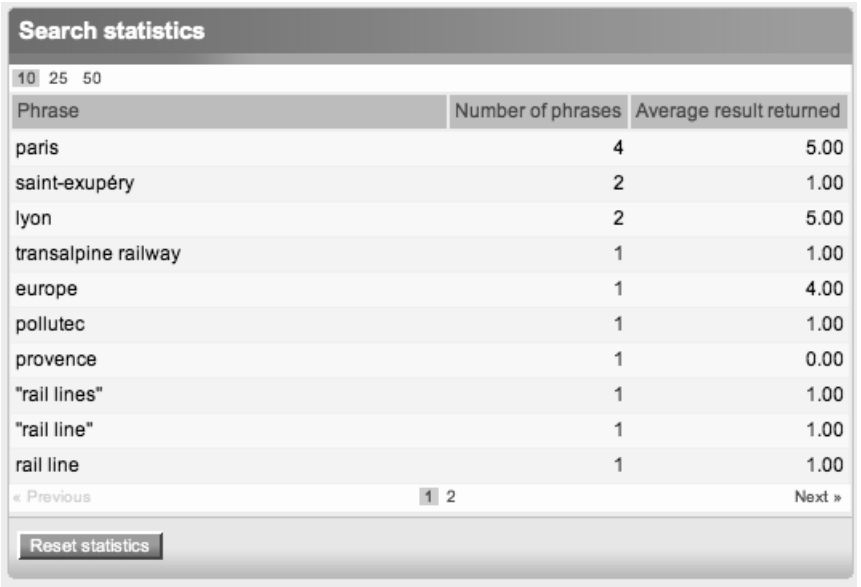


Figure 11.10. Search statistics window

In the top left corner of the **Search statistics** window, you can select how many queries to display per page. If the search statistics contain more entries than can fit on one page, there is a paginator at the bottom of the window. The only actions you can do within the search statistics are to view the list and reset the statistics. The latter is done by clicking the **Reset statistics** button at the bottom left of the window.

The queries are listed from the most recent to the oldest search query. Each query phrase is listed only once. Note that it is not possible to view or access the specific search results from the search statistics, although you can simply run those searches again to view the results for the current state of the site.

11.10. Summary

A search engine is an information retrieval system. In eZ Publish, all content marked as searchable is indexed by the built-in search engine. Users can submit queries to this engine through the **Search** interface.

The eZ Find extension provides improved and new features compared to the built-in search engine, such as the ability to require and exclude terms in a query, and displaying relevancy ranking percentages in search results.

Chapter 12. Working with multilingual sites

This chapter gives a comprehensive explanation of eZ Publish's built-in multilingual support and how to manage multilingual sites. A site is considered *multilingual* if it has more than one language installed and has multiple, language-specific public siteaccesses.

This material applies to all content managers and webmasters working with multilingual sites. It is advanced in the sense that it focuses mostly on management, rather than the day-to-day content editing that was covered in the *eZ Publish Content Management Basics* book. For those interested in only a conceptual introduction, Section 12.2, “Language and translation-related concepts” can be read separately.

Recall that:

- A *translation* is a representation of content in a specific language. The one-to-one translation mechanism makes it possible to represent the same content in multiple languages. Each content object version consists of at least one translation.

- A *Uniform Resource Locator* (URL), also commonly referred to as a "web address", tells a browser what you want to view and where to find it. *System URLs* are the raw identifiers used internally and normally not displayed to site visitors. *Virtual URLs* are more user friendly and are simplified versions of system URLs. A *URL alias* is an alternative, custom virtual URL.

- A site interface refers to the visual presentation, and is recognized through the address in the browser. There is a one-to-one relationship between a site interface and a *siteaccess*. A siteaccess is a collection of (override) configuration settings. The Administration Interface corresponds to the *admin* siteaccess, and the public front-end interface (possibly containing the Website Interface) corresponds to the *public* siteaccess.

- An eZ Publish *INI file* is a configuration file that controls the behavior of a specific part of the system. See Section 3.5, “The configuration model”. The main configuration file is `site.ini`. You can check the value of a particular setting within the **Settings view** interface as described in Section 3.5.4.1, “Viewing the settings of a configuration file”.

- A *module* offers an interface for web-based interaction, providing access to eZ Publish core functionality. Each module has a set of *views* that enable you to reach the functions provided by the module. See Section 1.3, “Modules and views”.

In this chapter, you will find information about the following:

- Language concepts and features
- Translation mechanisms and multilingual node URL aliases
- Object availability and the language fallback system
- How to manage language settings for your site
- How user management works for multilingual sites

eZ Publish URLs in general were covered in Section 3.3.1, “eZ Publish URLs”. Multilanguage considerations when importing and exporting content via WebDAV and the eZODF extension are discussed in the next two chapters.

This chapter should be read in the order in which it is written, but some sections can be skipped, depending on your experience. If you have previous experience with content editing operations involving multiple translations, you may skip Section 12.3, “Translation-related interfaces and windows” and Section 12.4, “Working with object translations”. If you are not going to handle language-related configuration, you may skip Section 12.6, “Configuring the site locale and languages” and Section 12.8, “Language-based permissions”. Note that Section 12.2, “Language and translation-related concepts” is a prerequisite for all subsequent material in this chapter.

12.1. Features of multilingual sites

An eZ Publish site is an expandable system. You may start with a single language site and add more languages and siteaccesses along the way. You may have a multilingual site, but initially only one translation for your content objects; over time, you translate the site's contents into more languages as resources permit. The following list highlights some of the built-in multilingual features:

- Display content according to locale-specific settings
- Have a one-to-one translation of content objects, representing the same content in different languages (except for non-translatable attributes)
- Display attribute labels and URL aliases according to the language used for content translation
- Have multiple concurrent translators working on different translations of the same object

12.2. Language and translation-related concepts

In this section, we provide an extensive description of eZ Publish's language-related concepts. Note that the covered concepts are at the object and class level, as well as the site and siteaccess level. We encourage you to pay special attention to the difference between translation languages, the main language and site languages.

12.2.1. Locale

A *locale* is a set of country-specific settings like language, character set, number format, currency format, data and time format, abbreviation for months and weekdays and more. These settings are defined in configuration files, and have standards-compliant identifiers like "eng-US" (English, USA) or "nor-NO" (Norwegian, Norway), which are further explored next. To view locale-specific settings, access the **Language management** interface (click the **Languages** link in the **Setup** tab) and click on the language name you want to explore. The following screenshot shows part of the language-specific locale settings for the "eng-GB" locale:

Figure 12.1. Language settings for "eng-GB"

12.2.1.1. Locale identifiers

A *locale identifier* like "eng-US" consists of a three-letter lowercase *language code* ("eng") and a two-letter uppercase *country code* ("US"). This identifier is used for several of the regional settings, including "Locale", "ContentObjectLocale" and "SiteLanguageList", which are described later.

You may say that a language is represented by a locale. For the "ContentObjectLocale" and "SiteLanguageList" settings, the system only uses the language information specified in a locale, not the number formats, time formats, and so on.

12.2.1.2. Locale-specific INI files

Each locale has a corresponding configuration file, named with the locale identifier, such as `eng-GB.ini`. These are not stored within the `settings` directory, unlike the configuration files we have been dealing with so far. Instead they are found in the `share/locale` directory (see Figure 1.2, “Directories below the eZ Publish root directory”).

You can view the contents of a locale configuration file (corresponding to one of your translation languages) as described above. However, the Administration Interface does not provide an interface for editing such files. You should never change an original locale configuration file. You can add files for missing locales to this directory, or edit copies of the original ones to make custom locales (as explored further in Section 12.2.10, “Translatable country names”).

12.2.2. Site locale

The *site locale* refers to the value of the "Locale" setting within the "[RegionalSettings]" block of `site.ini`, for a siteaccess or globally. The default site locale is "eng-GB". The specified locale will affect how some content is displayed on the site. It is common to use different site locales in your different language-specific siteaccesses. For example, the override file for the English (American) siteaccess would have "Locale=eng-US", the override file for the Norwegian siteaccess would have "Locale=nor-NO", and so on. This would, for example, result in your timestamp values for a blog post to be shown differently in the Norwegian and English siteaccesses, as illustrated below with the English siteaccess on the left and the Norwegian siteaccess on the right.

Figure 12.2. Blog post timestamp in "eng-US" and "nor-NO"

Section 12.6.1, “Configuring the site locale” describes how to edit this setting.

12.2.3. Default language

The *default language* replaces the deprecated notion of *primary language*, and it corresponds to the "ContentObjectLocale" INI setting. The settings value (in the "[RegionalSettings]" block in `site.ini`) is one of the locale identifiers, for example "ContentObjectLocale=eng-GB".

The default language is used in one aspect of the language fallback system (described in Section 12.2.7, “Availability and fallback system”), as well as by developers. What content managers might describe as the "default" language is more likely either the highest prioritized site language for a siteaccess, or the main or initial language of some content object. These concepts are described below.

12.2.4. Translation languages

The term *translation languages* denotes the set of languages in which you can create and translate content on your site. In other words, this is a site-wide list. The entire set of translation languages is commonly referred to as the *global content translation list*.

Each of the translation languages are identified by a locale identifier, such as "eng-GB". In the Administration Interface, they are often represented by more human-readable names, such as "English (United Kingdom)".

For example, the translation languages are listed in the dropdown list in the **Create here** interface at the bottom of the **Sub items** window.

Translation languages are specified in the database; consequently, there is no corresponding INI file setting. The database limits the number of languages that can be used simultaneously to 30. You may add and remove languages within this limit as required. The translation languages are managed through the Administration Interface as described in Section 12.7, “Managing translation languages”.

Translation languages simply define availability. In other words, you can have translation languages for which there are no content objects. However, if a language is not in the set, content cannot exist in that language, you cannot create new content in it, nor can you translate content to it. It simply does not exist within the context of your site. The site languages described below must be part of the translation languages list.

12.2.5. Site languages

Site languages, represented by locale identifiers, define a subset of the translation languages. This set is actually a prioritized list defining which translation to select from a multilingual content object within a given siteaccess.

Site languages can be specified globally or per siteaccess in the "[RegionalSettings]" block of the `site.ini` configuration file, as described in Section 12.6.2, “Configuring

site languages". Priorities correspond to the order in which the locale identifiers are listed in the "SiteLanguageList" settings value, and are handled by eZ Publish according to the *language fallback system* (described later in Section 12.2.7, "Availability and fallback system").

12.2.6. Main language

So far we have seen that the translation languages list specifies which languages are supported by your site. Also, the site languages list is a prioritized subset of the translation languages from which the displayed translation can be picked by the language fallback system for a given siteaccess (or globally). None of these concepts say anything about what translations an existing content object actually has.

The *main language* refers to a single, specific language for one particular translation of one particular content object. It is therefore object-specific, and automatically set by the system when the object is created, based on the language selected. For example, if an article is created in French, its main language is set to French. It must belong to the translation languages, but not necessarily to the site languages. The main language is also of importance for the availability and fallback system described below.

The terms "initial language" and "main language" mostly refer to the same thing. The former is used at the system level whereas the latter is used in the Website Interface and the Administration Interface. When you create an object, its initial and main language are the same, but you can change the main language later if the object contains multiple translations.

12.2.7. Availability and fallback system

The translation languages and site languages provide great flexibility when it comes to the languages in which a content object can exist and be displayed. Consequently, a mechanism is needed to handle language filtering and prioritizing in order to ensure that requested content is displayed. The integrated *language fallback system* provides this. We first describe the normal filtering process, then we explain the possible overrides.

12.2.7.1. Translation filtering

The language fallback system is used every time a content object is requested, to select one of the object's translations to display. *Filtering* is based on the site languages for a particular siteaccess according to the "SiteLanguageList" configuration setting (see Section 12.6.2, "Configuring site languages"). The language that is listed first gets the highest priority. The closer a language is to the bottom of the list, the lower is its priority.

Language filtering works as follows. The language fallback system first checks the list of site languages. In the case where the "SiteLanguageList" setting is empty (no site languages are listed) the language fallback system picks the translation corresponding to the default language (the "ContentObjectLocale" setting) and ignores all other transla-

tions. Otherwise, if the requested content object has a translation in the highest prioritized language, the translation is selected and displayed and the filtering process terminates. If an object is not available in this language, the check is repeated for the second prioritized language and so on. In other words, each time a particular translation is not found, the system checks for a translation of lower priority.

If the process terminates without a match, this means that the requested content object does not have a translation in any of the site languages (even if it has other translations), and is thus not available. You may say that translations other than those in the site languages are filtered out. For example, when viewing a news folder, the unavailable articles are not listed. This is a more severe problem if your frontpage is subject to filtering and its display is prevented in some siteaccesses. Even worse, if content objects representing user accounts do not produce a match, those users will not be allowed to log in to or access the siteaccess. Because of this, several override mechanisms exist for the language fallback system. You may think of them as safety nets below the lowest prioritized site language.

12.2.7.2. Overriding translation filtering

If an object does not exist in any of the site languages (filtering terminates without a match), it will not be shown unless at least one of the following is true:

- The siteaccess is configured to display untranslated content.
- The object is marked as *always available*.

12.2.7.2.1. Untranslated content

Untranslated content is a content object that does not have a translation in any of the site languages for a siteaccess, and is excluded, by default, by the language fallback system. By enabling the "ShowUntranslatedObjects" setting, located in the "[RegionalSettings]" block of `site.ini`, untranslated content is shown in the object's main language. Note that the fallback system goes through the site languages as usual; this setting simply treats the main language as an additional option (that will always match) at the bottom of the list. In other words, all objects will be displayed regardless of which translations they have, and objects that exist in a language specified in the priority list are still displayed using the prioritized language.

This setting is disabled by default for all siteaccesses except the admin siteaccess.

12.2.7.2.2. Always available object

Object availability is the second safety net in place for objects that do not have a translation in any of the site languages. This is typically the case for objects of the `User`, `User group`, `Image`, `File` and other multimedia classes, as well as containers such as folders.

Object availability is a property granted to individual objects. For simplified administration, an initial value for this property can be set on a class by class basis, affecting all

new objects created based on that class. As was the case with untranslated content described above, the object's main language is then treated as an additional option (that will always match) at the bottom of the site languages list.

At the class level, this property is granted or removed for all new objects by toggling the **Always available** checkbox. To do this, bring up the **Class edit** interface for the content class in question as described in Section 3.2, “Viewing and editing content classes”. Changes to this property have no effect on existing content objects, as it is only an initial value.

At the object level, object availability is specified by toggling the **Use the main language if there is no prioritized translation** checkbox within the **Translations** window and clicking the **Update** button:

Figure 12.3. Translations window - toggle object availability

If the **Always available** checkbox for the class of this object is marked (at the time the object was created), the **Use the main language if there is no prioritized translation** for the object is automatically marked. You can unmark it for specific objects, update old objects after modifying the class, or simply grant availability to selected objects.

The diagram below summarizes the availability and fallback system:

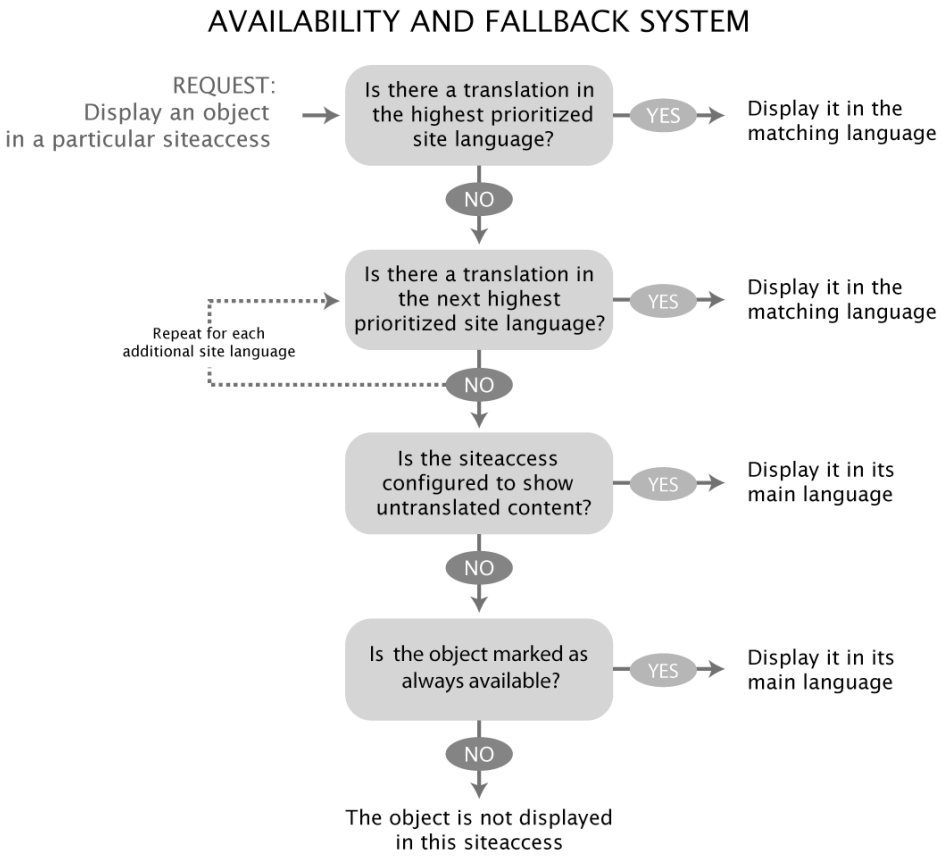


Figure 12.4. Availability and fallback system

In Depth: Differences and similarities between filter overrides

Both of the language filtering overrides make use of an object's main language to display it when it does not have translations that match any of the languages on the site languages list. The difference is the granularity of the overrides.

The "ShowUntranslatedContent" setting affects all untranslated content objects, either for a specific siteaccess or the entire site. It is a configuration setting, viewed and edited in the **Settings view** interface of the **Setup tab** (or manually through the file system).

Object availability is, in contrast, a property assigned to individual objects, and is independent of the siteaccess. It is viewed and edited in the **Translations** window, or alternatively by clicking the **Classes** link in the **Setup** tab.

If "ShowUntranslatedContent" is enabled for a siteaccess, objects are shown regardless of whether they have the object availability property on or off. It is quite normal that both safety nets are used in combination, with object availability enabled for important objects site-wide, and having all untranslated content shown for specific siteaccesses (such as in the Administration Interface).

12.2.8. Non-translatable attributes

It is convenient to translate the contents of an attribute when the attribute contains text, such as the title of an article or the text of a documentation page. However, some attributes are non-translatable by nature, such as numbers, dates, email addresses and images without text. Such attributes can be made non-translatable by toggling a generic control when editing the class. As a result, when you translate an object, contents of these attributes are simply copied from the main translation. Editing the copied values is prohibited. To change the value of a non-translatable attribute, you have to edit the translation corresponding to the object's main language.

12.2.9. Translatable class attributes

Multilingual support was extended in eZ Publish version 3.9 to also apply to content classes and their attribute names. The system can display the attribute labels in the correct language when users are editing or viewing the different translations. Class editing and translation is done in the **Class edit** interface (see Section 3.2, “Viewing and editing content classes”). Class management is normally handled by developers and is beyond the scope of this book. Interesting readers should refer to http://ez.no/doc/ez_publish/technical_manual.

12.2.10. Translatable country names

eZ Publish 3.9 and above also enables you to translate the names of countries to different languages, and have the translated name appear in the list of countries whenever it is displayed, for example for attributes of the "Country" datatype (see Section 15.5.4, “Country datatype”) based on the locale. Therefore, in the Norwegian siteaccess, "Norway" could be shown as "Norge". To use translated country names, you must create a custom INI file and update the "Locale" setting to use this configuration file (see Section 12.6.1, “Configuring the site locale”). Refer to http://ez.no/doc/ez_publish/technical_manual or ask your site administrator for more information on how to create an INI file.

12.3. Translation-related interfaces and windows

Several interfaces and windows are involved in managing multilingual site features. You should already be familiar with the **Translations** window, the **Sub items** window, the left menu and the context-sensitive pop-up menu. The **Translation** interface, a special-purpose mode of the **Object Edit Interface**, was described in *eZ Publish Content Management Basics*.

Next, we will cover the **Language selection** interface and the **Language management** interface.

12.3.1. Language selection interface

The **Language selection** interface of the Administration Interface is presented every time you create a new content object (except from the **Sub items** window, which enables you to pre-select a language), add a translation, or initiate editing on a multilingual site. The display is slightly different depending on the context. For example, when you create a new object, you only have to select the language in which to create it (the main language):

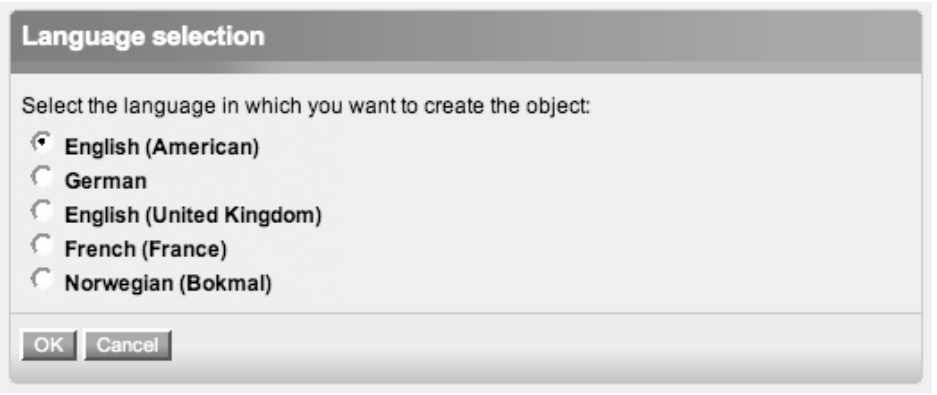


Figure 12.5. Language selection interface (reduced)

In contrast, when adding a translation, you need to select a language for the new translation and which, if any, of the existing translations on which to base it:

Figure 12.6. Language selection interface

12.3.2. Language management interface

The **Language management** interface is located within the **Setup** tab and accessed by clicking the **Languages** link in the left menu. It is shown in the screenshot below:

Figure 12.7. Language management interface

This is a special-purpose interface provided by the "translations" view of the "content" module to manage the translation languages (see Section 12.2.4, “Translation languages”). The screenshot above shows that five translation languages exist: English (both British

and American English), French, German and Norwegian. For each language, the country / region and locale (see Section 12.2.1, “Locale”) are listed, as well as the number of objects and classes that have translations in that language.

In this interface, you can view (but not edit) the specific settings of a locale by clicking on the corresponding language name. You can also add and remove languages as described in Section 12.7, “Managing translation languages”.

12.4. Working with object translations

Multilingual objects, just like any content objects, follow the life cycle depicted in Section 1.2.3, “Version management and content life cycle”. What is special is that when a draft is set to be published, its translation is *merged* with the existing translations into the published version of the object. It is this behavior that lets multiple editors work on different translations of the same object at the same time. For content objects with only one translation, no merging takes place.

The following table gives a quick overview of the operations you can perform on a translation. Refer to *eZ Publish Content Management Basics* for full descriptions of the editing procedures. Note that these operations apply to content, while the operations described in Section 12.7, “Managing translation languages” apply to the set of languages in which objects can exist. Also, adding and deleting a translation only affects the target translation; it does not create or delete content objects.

Tip

In the **Object Edit Interface**, you can toggle translator mode on and off and change the existing translation on which to base the current translation in the **Translate from** window on the left.

Table 12.1. Available translation operations

Operation	Procedure
Add translation (translate existing content)	First, locate the object you want to translate. Then, select "Another language" from the "Edit in" sub-menu from the context-sensitive pop-up menu; or choose "Another language" in the dropdown list in the **Preview** window and click the **Edit** button; or click the **Edit** button in the **Sub items** window. Then, select a language for the new translation in the **Language selection** interface and enter edit mode. When you publish the new translation, it will merge with the other translation(s).

Operation	Procedure
Edit translation	As described for adding a translation, but select the language corresponding to the translation you want to edit rather than "Another language". You can alternatively click the edit icon in the **Translations** window for the desired translation, or pick a translation in the **Version history** interface.
Delete translation	Deleting a translation affects all versions of the object that have this translation. You cannot delete the translation currently set as the main language. To delete a translation, access the **Translations** window for the target object. Mark the checkbox(es) corresponding to the desired translation(s), then click the **Remove selected** button.

Tip

Adding and editing a translation can also be done via the Website Interface.

12.5. Multilingual node URL aliases

eZ Publish URL aliases were explained in Section 3.3.2, “URL aliases”. Recall that if the system URL (in the alias destination) is of the form "content/view/full/node_id", then the alias is a *node URL alias*. All other user-defined virtual URLs are considered *global URL aliases*.

Site languages play an important role for multilingual node URL aliases. Each alias is associated with one of the site languages, and there can be multiple aliases for each language.

Global URL aliases are managed through the **URL translator** interface in the **Setup** tab. Node URL aliases can also be added through that interface, but they are not shown in the global list. It is more common that these are managed through the **Node aliases** interface as described below.

Tip

The interfaces described here are the same whether you have a single-language site or a multi-language site.

12.5.1. Node aliases interface

The **Node aliases** interface is accessed from the "Advanced" sub-menu of the context-sensitive pop-up menu for a given node.

Figure 12.8. Pop-up menu - Manage URL aliases

Tip

If you cannot access the pop-up menu, you can access the URL for the **Node aliases** interface directly, provided that you know the target Node ID. You can check the ID in the **Details** window when viewing the node or hover over a link for the node's name.

Append "content/urlalias/node_id" (replacing "node_id" with the actual ID number) to the base URL of the Administration Interface. For example, you would access `http://www.example.com/cm_admin/content/urlalias/138`. This tells the system to access the "urlalias" view (representing the **Node aliases** interface) of the "content" module for the specific node (in this case, the node with a Node ID of 138).

The following screenshot shows the **Node aliases** interface:

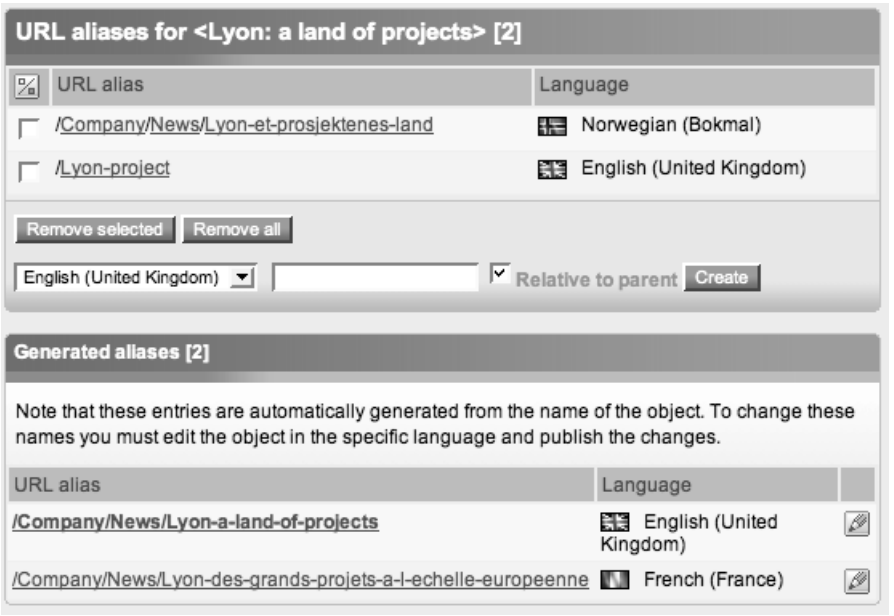


Figure 12.9. Node aliases interface

The top window is the actual management interface. Here you can add and remove existing aliases for the selected content node. It contains custom virtual URLs for the node, while the **Generated aliases** window at the bottom lists automatically-generated aliases. Together, these two windows show all of the URL aliases belonging to the selected node.

To remove a node URL alias, mark the corresponding checkbox(es), then click the **Remove selected** button. Or simply click the **Remove all** button. Note that this will only affect the custom aliases, not the ones in the **Generated aliases** window.

To create a node URL alias, use the **Language selection** dropdown list, the alias text input field, the **Relative to parent** checkbox and the **Create** button, all at the bottom of the top window.

Editing cannot be done directly on a custom alias. You have to create a new alias with the desired changes, and remove the old alias that you wanted to update. In other words, editing in this case involves replacing, rather than modifying.

By default, the **Language selection** dropdown list contains the site languages set for the current siteaccess, listed in the same order as specified in the "SiteLanguageList" setting. But, if the "ShowUntranslatedContent" setting is enabled, all of the translation languages are listed. The language associated with a particular alias affects the siteaccesses for which that particular alias will be available. For example, a French alias will be available in all siteaccesses that include French as a site language.

If the **Relative to parent** checkbox is marked, the system will add the current path to the alias text entered when creating new node aliases. In other words, it saves you the job of having to type in the full path (which is prone to errors).

12.5.1.1. Generated aliases window

Recall that the system automatically creates and maintains virtual URLs when content objects are published. For each content object there is one entry for each of the object's translations. In the case above, the object has an English and a French translation. For the current siteaccess, English has a higher priority in the site languages list than French and is displayed using bold characters to indicate this.

A child node does not inherit all URL aliases from its superior nodes. Rather, a single location and a single language is picked based on the highest prioritized site language of the siteaccess. For example, the path "/Company/News" is used in our example instead of something like "/Compagnie/Nouvelles" because English is the highest prioritized language for that siteaccess.

Note that the automatic virtual URLs cannot be edited directly, nor can they be removed. This is so because the path and name are pulled from the object itself. To alter the text in the automatically-generated URL, you have to edit the actual object by clicking the corresponding **Edit** button, which takes you to edit mode.

12.6. Configuring the site locale and languages

This section is the most technical in this chapter, and requires some knowledge of the configuration model (see Section 3.5, “The configuration model”) and the directory structure (see Figure 1.2, “Directories below the eZ Publish root directory”). We focus on the main principles rather than the exact technical details, and recommend that you supplement this manual with the technical manual at http://ez.no/doc/ez_publish/technical_manual.

12.6.1. Configuring the site locale

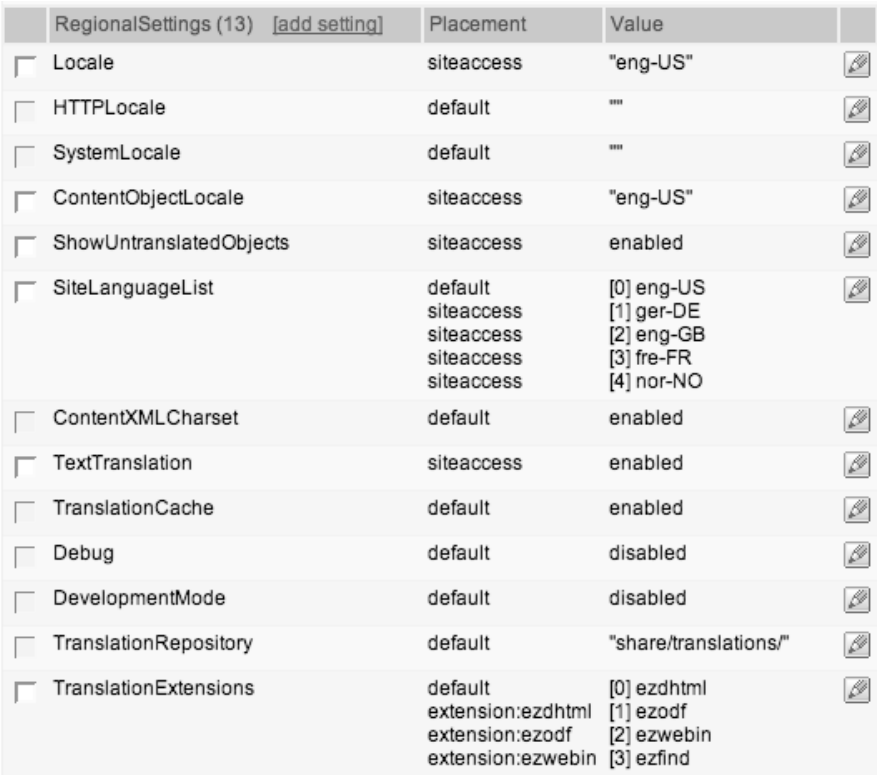
Recall that the "Locale" setting pulls the information from a locale regarding location-specific timestamps, currency formats, and so on. This way, your site can be tailored for Norwegian, British, Chinese, and other visitor locations, regardless of the languages in which content exists. The following list shows the information needed to find the "Locale" setting.

- file: *`site.ini`* (global or siteaccess override)
- block: "[RegionalSettings]"

- setting: "Locale"

Suppose you set up your site for an English audience, but discover after some focus group studies that the majority of your visitors are Norwegian. Or, suppose you want to have the Administration Interface use a different locale. You might then want to switch the site locale setting accordingly. You can edit the file manually, or through the **Settings view** interface, the procedure for which we describe below:

1. Access the **Settings view** interface as described in Section 3.5.4, “Settings view interface”. Select "site.ini" from the top dropdown list and one of the siteaccesses, then click the **Select** button. Scroll down until you find the "[RegionalSettings]" block:



RegionalSettings (13) [add setting]	Placement	Value
Locale	siteaccess	"eng-US"
HTTPLocale	default	""
SystemLocale	default	""
ContentObjectLocale	siteaccess	"eng-US"
ShowUntranslatedObjects	siteaccess	enabled
SiteLanguageList	default siteaccess siteaccess siteaccess siteaccess	[0] eng-US [1] ger-DE [2] eng-GB [3] fre-FR [4] nor-NO
ContentXMLCharset	default	enabled
TextTranslation	siteaccess	enabled
TranslationCache	default	enabled
Debug	default	disabled
DevelopmentMode	default	disabled
TranslationRepository	default	"share/translations/"
TranslationExtensions	default extension:ezdhtml extension:ezodf extension:ezwebin	[0] ezdhtml [1] ezodf [2] ezwebin [3] ezfind

Figure 12.10. Viewing the "[RegionalSettings]" block

2. The "Locale" setting is listed as the first entry for this block. The screenshot shows that the current value is "eng-US" and that it is set for the siteaccess. To change it, click on the **Edit** button to the right.

3. This will open the **Settings edit** interface:

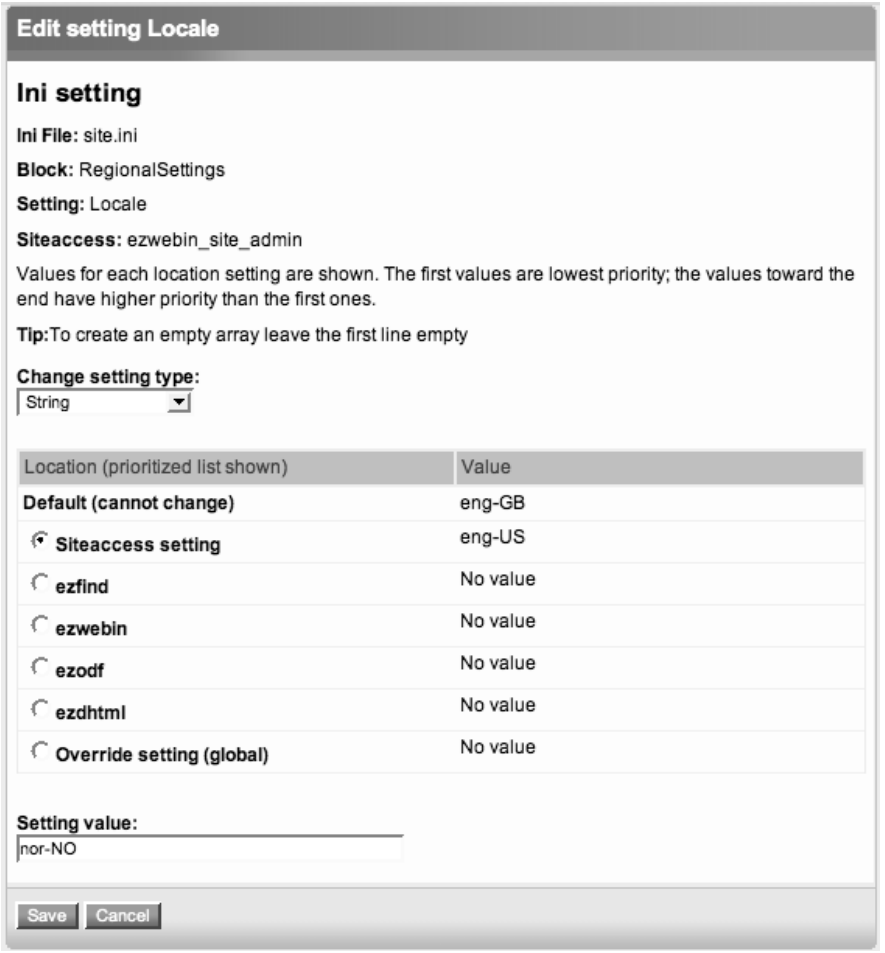


Figure 12.11. Settings edit interface

4. Ignore the setting information at the top, and leave the **Change setting type** drop-down list as is. At the bottom is a list of locations with associated radio buttons and a **Setting value** input field.

   To change the value for only one siteaccess, select the **Siteaccess setting** radio button. To change it for all your siteaccesses, select the **Override setting (global)** radio button.

5. Enter a valid locale identifier in the **Setting value** field. For illustrative purposes, we use "nor-NO".

6. Store your changes by clicking the **Save** button, or abort by clicking the **Cancel** button. The system will now start to use the locale settings specified in the *share/locale/nor-NO.ini* file for the selected siteaccess.

12.6.1.1. Updating interface translation according to your site locale

Changing the site locale as described above may not affect the translation of all interface parts accordingly. Although the site locale is "nor-NO", links and buttons might still be displayed with untranslated labels. There are two main possible reasons for this. First, text translation may be turned off for your site or the specific siteaccess. View the "[RegionalSettings]" block as previously described, and check the value set for "TextTranslation". If it is disabled, edit the setting and choose "enabled" in the **Settings value** list. If it is enabled, the system should translate all strings marked for this purpose in the template according to the current locale. The strings are translated based on eZ Publish specific translation files. Therefore, the other possible reason is that the string is not marked to be translated in the template, or that there is no translation available. Contact your site administrator for assistance.

12.6.2. Configuring site languages

The site languages concept was described in Section 12.2.5, “Site languages”. Recall that this setting specifies the priorities that determine which translation is displayed for a given siteaccess (or the entire site). In contrast to the site locale, which is a single value, the site languages setting is a list with multiple entries, one entry per language. The following list shows the information needed to find this setting.

- file: *site.ini* (global or siteaccess override)
- block: "[RegionalSettings]"
- setting: "SiteLanguageList"

You can edit the file manually or through the **Settings view** interface. The procedure is very similar to the previous one, but the settings value in this case is an array (list with multiple values) rather than a string (single value):

Figure 12.12. "SiteLanguageList" - edit settings value

To add additional site languages, add a new line for each new locale identifier in the same format as the existing entries.

If you simply want to change one of the languages, you could replace one string with another valid locale identifier.

To rearrange the priorities, simply rearrange the lines. The top line has the highest priority.

Remember to select the siteaccess radio button (not the global override) if you have multiple language-based siteaccesses.

12.7. Managing translation languages

The concept of translation languages was described in Section 12.2.4, “Translation languages”. Recall that this is the set of languages in which you can create and translate objects.

Translation languages are managed through the **Language management** interface. Remember that, in contrast to the site locale and site languages described above, translation languages are stored in the database and thus not accessible in an INI file.

As Figure 12.7, “Language management interface” shows, the global content translation list in our example consists of English, French, German and Norwegian. Follow the procedure below to add a language to the set.

1. Access the **Language management** interface by clicking the **Languages** link in the left menu of the **Setup** tab.

2. Click the **Add language** button. This will open the following window:

Figure 12.13. Add language window

3. Select the desired language from the **Translation** dropdown list. The contents of the list reflect the locales available on your installation.

   The two input fields at the bottom are only used for custom languages (see Section 12.2.1.2, “Locale-specific INI files”).

4. Click the **OK** button. The new language will be added to the list, as shown below.

Figure 12.14. Language management interface - Portuguese added

Removing a translation language is done by selecting one or more languages by marking the corresponding checkboxes, then clicking the **Remove selected** button. There will be no warning or confirmation dialog; the removal is done silently. However, you are only permitted to remove languages in which there are no object or class translations (and must therefore delete all translations in a language before doing so). Otherwise, the corresponding checkboxes are disabled. Examine the **Translations** column to see if the

language is in use. In the above screenshot, all except the French and English (American and British) languages can be removed.

12.8. Language-based permissions

Access control and user management were described extensively in Chapter 5. Recall that a policy grants access to one or all functions of a module, possibly with some limitations. The "content" module has three functions that support limitations at the language level. These are the "create", "edit" and "translate" functions. By adding language limitations, you can control in which of the translation languages these operations can be used.

Note that the "read" function does not support the language limitation, but it is required in combination with those functions to ensure that you can carry out the tasks. After all, if you cannot view some content, you should not be able to edit it.

Tip

Combining function limitations was described in Section 5.7.2.3, “Working with policies”.

The following table summarizes which content functions are needed for which tasks.

Table 12.2. Content functions

Task	**Functions needed**
Create new object	Read, Create and Edit
Edit	Read and Edit
Add translation to existing object	Read, Translate (create translation) and Edit

Suppose you want to force all Editor users to create objects in English, but you also want to allow them to translate objects into other languages. In such a case, you would use the language limitation for the "create" function, but not on the "edit" or "translate" functions.

12.9. Summary

The following list summarizes the most important concepts and interfaces introduced in this chapter.

- The *locale* is a set of country specific settings that will affect how content is displayed on the site. The "Locale" setting (the *site locale*) specifies the locale to use site-wide or for a specific siteaccess.

- *Translation languages* are the languages you can use on your site for creating and / or translating content. This set is also referred to as the *global content translation list*.

- The *site languages* list affects which translation of multilingual content is displayed on the site. The list is used by the language fallback system to do filtering and prioritizing.

- The *main language* usually refers to the language selected when the object was created. This can be changed later to a different translation.

- The *always available* property can be set on objects to ensure that critical content like folders and user accounts are not filtered out by the language fallback system if they do not exist in any of the prioritized site languages, but that they are displayed in the main language.

- The "*ShowUntranslatedContent*" setting, when enabled for a siteaccess, ensures that all content that does not exist in any of the prioritized site languages is shown in the main language.

- *Node URL aliases* are virtual URLs for content nodes, managed through the **Node aliases** interface, accessed from the pop-up menu.

- The **Language management** interface is accessed by clicking the **Languages** link in the left menu of the **Setup** tab and is used for managing the list of translation languages.

Chapter 13. WebDAV

This chapter explains how to use *Web-based Distributed Authoring and Versioning* (WebDAV) with eZ Publish. WebDAV makes it possible to browse, create, remove, upload, download and rename content on your site using a WebDAV-capable file browser on your computer. This is similar to a network share, where multiple users can access a central file repository. Using WebDAV on your site is an alternative to carrying out these tasks in the Website Interface or Administration Interface.

This chapter particularly applies to readers who are interested in learning about how WebDAV can help to make content management more efficient, for example when uploading large amounts of content to your site. In one example, media sites could let photographers submit images using a familiar file browser such as Windows Explorer, removing the requirement of understanding the eZ Publish interfaces.

Recall that:

- A site interface refers to the visual presentation, and is recognized through the address in the browser. There is a one-to-one relationship between a site interface and a *siteaccess*. A siteaccess is a collection of (override) configuration settings. The Administration Interface corresponds to the *admin* siteaccess, and the public front-end interface (possibly containing the Website Interface) corresponds to the *public* siteaccess. Multilingual sites commonly have language-specific public siteaccesses.

- An eZ Publish *INI file* is a configuration file that controls behavior of a specific part of the system. See Section 3.5, “The configuration model”. The main configuration file is `site.ini`. All setting modifications should be done in a global or siteaccess-specific override of the configuration file in question. You can check the value of a particular setting within the **Settings view** interface as described in Section 3.5.4.1, “Viewing the settings of a configuration file”.

In this chapter you will find information about the following:

- An introduction to the WebDAV protocol, including the requirements for using it and how it can benefit your site.

- How to connect to eZ Publish using a WebDAV client.

- How to browse, import, export, create, move, copy, rename and delete content via WebDAV.

The examples use Windows Explorer as the WebDAV client. If you work with one of the other clients (explained in Section 13.1.1.1, “WebDAV clients”), expect to see some minor variations in the interfaces. However, the general principles are the same. Also,

note that WebDAV is used together with files and folders on your own computer, and it is assumed that you have knowledge about general file management.

This chapter should be read in the order given. Readers who do not feel that they have a need to use the WebDAV features can either skip or skim this chapter, although it serves as a good introduction to what WebDAV can do. In Section 13.2, “Setting up a WebDAV connection”, you can focus your attention on the WebDAV client available for your operating system. Section 13.5, “Using WebDAV with your eZ Publish site” includes practical examples that can be skipped if you feel that basic usage knowledge is sufficient.

13.1. Introduction to WebDAV

WebDAV is a set of protocol extensions to HTTP that enables you to access content on a website through an interface similar to the file browser on your computer. eZ Publish supports the use of the WebDAV protocol in order to access, navigate and manage the objects of the content node tree as represented by files and directories:

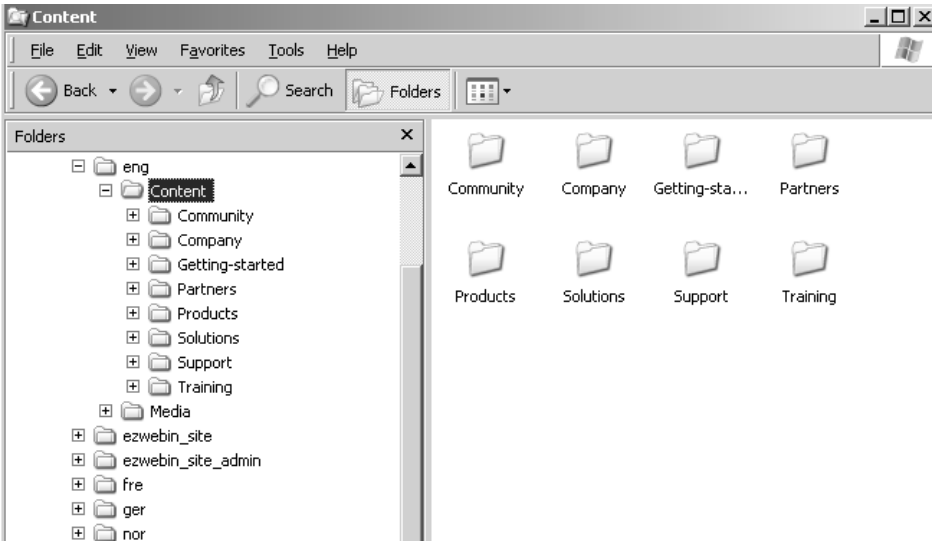


Figure 13.1. eZ Publish content in WebDAV, displayed using Windows Explorer

This requires a WebDAV-compatible client, which is a program installed on your local computer, capable of connecting to the server via the WebDAV protocol. In contrast, the Administration Interface and Website Interface are accessed through a web browser.

After you have successfully made a WebDAV connection to your site (as described later in Section 13.2, “Setting up a WebDAV connection”), you should see the list of your site's siteaccesses represented as folders.

For multilingual sites with language-specific siteaccesses, browsing the site through WebDAV works the same as when browsing through eZ Publish: available objects are shown and modified in the language specified by the fallback system, as described in Section 12.2.7, "Availability and fallback system".

The following list outlines some common WebDAV usage scenarios:

- Batch uploading and downloading of pictures and files: instead of uploading and downloading files one-by-one, you can process batches of files. You can even work offline and upload all updated content through the WebDAV client when your network access is next available. Uploading corresponds to clicking the **Send for publishing** button in the Administration Interface or Website Interface, triggering the normal actions that are part of your publication process.

- Perform content management tasks in a familiar file browser environment.

- Provide a contribution repository for non-regular users. While content managers and editors can work on the published website content, infrequent contributors can place files in a folder via WebDAV for subsequent review.

- Create an Intranet document storage system with versioned and permission-controlled content objects.

Because WebDAV connections use the HTTP protocol, the only network requirement is that you can access your site, either through the Internet or a local connection. In other words, if you are able to manage your site through the Administration Interface or Website Interface, you do not need any additional firewall configurations. However, some technical prerequisites exist, as outlined in the next section.

13.1.1. Requirements for using WebDAV with eZ Publish

There are two main requirements for using WebDAV on your site:

- WebDAV must be enabled in the installation and configured on the web server. Consult your site administrator if you are unsure about the details.

- You must have a WebDAV-compatible client application.

In addition, you need a valid user account with access permissions for the content and tasks in question. This is the same as those required when you are working through the traditional eZ Publish interfaces. Refer to Chapter 5 for a description of the built-in user groups and roles.

13.1.1.1. WebDAV clients

Depending on which operating system you are using, there are different WebDAV clients available. Windows Explorer for Windows, Transmit for Mac OS X, and Konqueror for KDE versions of Linux are examples of WebDAV clients that can be used to access an eZ Publish site.

The following screenshots show how siteaccesses are listed in the various clients.

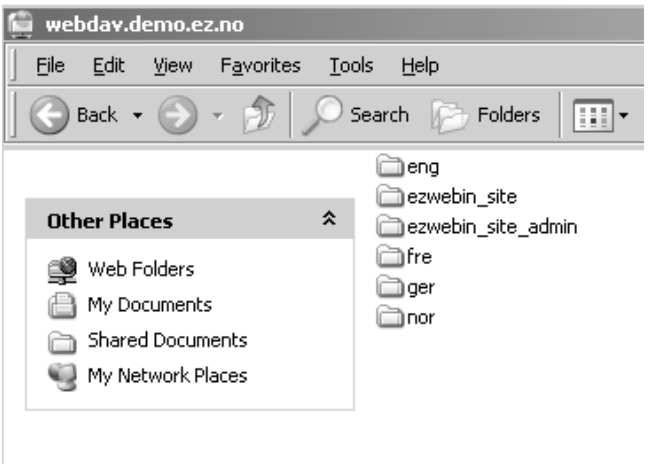


Figure 13.2. List of siteaccesses in Windows Explorer

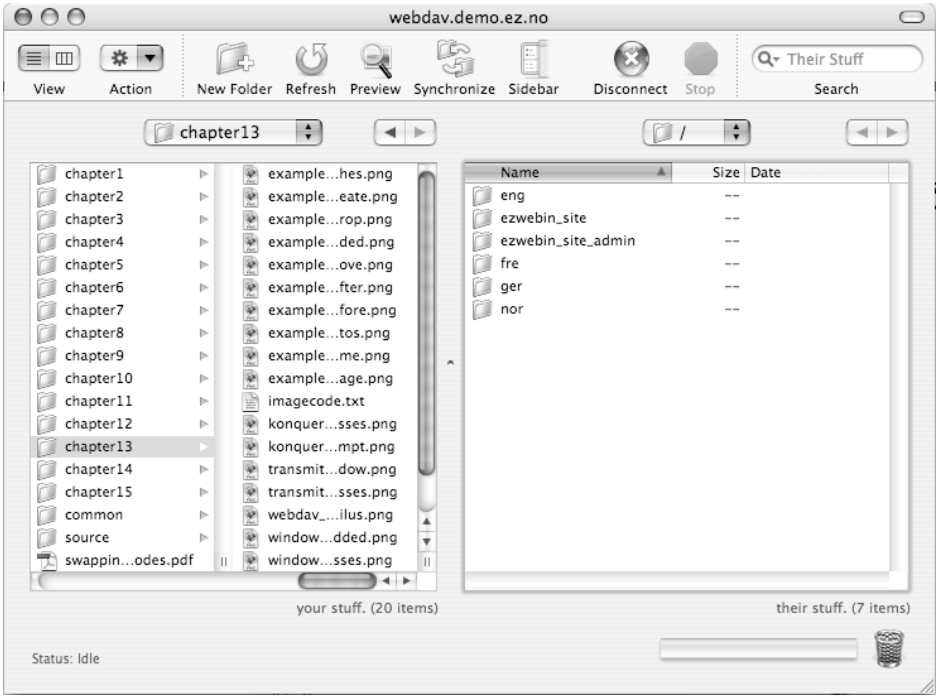


Figure 13.3. List of siteaccesses in Transmit for Mac OS X

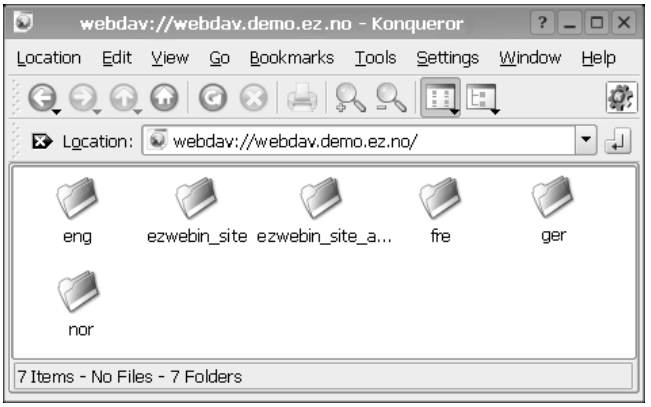


Figure 13.4. List of siteaccesses in Konqueror for Linux

Procedures for performing the operations in a WebDAV client vary from client to client. However, the basic principles should be the same. If you have any trouble using a particular client, consult its documentation.

13.1.2. Mapping eZ Publish classes to WebDAV folders and files

WebDAV clients have a limited set of content types compared to eZ Publish. They operate on folders and files, and do not recognize eZ Publish content classes. Mapping is therefore required to determine how content classes are represented on the local file system (either as folders or as specific file types.) This is done through the *webdav.ini* configuration file by your site administrator. Similarly, mapping is required to determine which uploaded file types create objects of which classes. This is done through the *upload.ini* configuration file. For specific information about settings, consult your site administrator or the technical manual at http://ez.no/doc/ez_publish/technical_manual.

Custom upload handling is configuration that extends the basic mapping. For example, with the eZ Open Document Format extension (described in the next chapter), you can upload OpenDocument Text files and they will be automatically created as objects of the `Article` class. See the technical manual for more information about custom upload handling.

13.1.2.1. Class-dependent display rules

Content is shown in the WebDAV client using the same names as within the eZ Publish interfaces, with one exception: spaces are replaced by dashes. For example, a container node with the name "Getting started" would be represented in WebDAV by a folder named "Getting-started". Recall that the name of a content node is usually pulled from the "Name" or "Title" attribute of the object it encapsulates.

By default, three display styles are available: folder, file, and image. To be considered a WebDAV folder, the object's content class has to be listed in the "FolderClass" setting in *webdav.ini* or have the container flag set. To be considered either a WebDAV image or file, the object's content class has to include an attribute of the "Image" or "File" datatype. For example, objects of an `Article` class that have an "Image" attribute will be represented in WebDAV as images. Objects of the multimedia classes (such as the `Flash` and `Quicktime` classes) and objects of the `File` class are the most common objects represented as files, with the corresponding file extension appended to their titles. For example, an object of the `Flash` class with the title "Action" would be represented as a file named "Action.swf".

Tip

Some container objects might not be recognized as folders by WebDAV. For example, objects of the `Article` class are treated as images because of their "Image" attribute, although they are containers and might hold comments. As a consequence, you cannot view and manage sub-items as you can in the Administration Interface and Website Interface.

If you do not need to access the image attribute of an article, you can add this class to the "FolderClass" setting in *`webdav.ini`* so that it is listed as a folder.

eZ Publish determines the type of each object based on an ordered list of criteria. If a specific criterion is not met, the next criterion is used, and so on. When a match is found, the process terminates. This is summarized in the list below:

1. The *`webdav.ini`* configuration file is checked. If a class is included in the "FolderClass" settings, objects of that class are represented as folders in the WebDAV client.

2. The attributes of the content class are checked in the order they appear in the class definition. (You can view class definitions by following the procedures given in Section 3.2, “Viewing and editing content classes”.) If an object has an attribute of the "Image" or "File" datatype, it is represented as the image or file in that attribute. If multiple attributes of the same class use one or both of these datatypes, it is always the first match that is used. If an object has an "Image" attribute with no image, it is represented as a zero-byte file (see below).

3. The container flag of the content class is checked. If the flag is set, objects of that class are represented as folders.

4. All other objects are represented as zero-byte files.

13.1.2.1.1. Zero-byte files

A *zero-byte file* is a named file without contents. In other words, it has a filename but the file size is zero bytes.

eZ Publish content is represented in WebDAV as zero-byte files in two cases. The first case is when the object is to be represented (in the determination process described above) as an image or file, but the attribute in question is empty. This might be the case if the attribute is optional (not marked as required), such as the "Image" attribute of the `Article` class. The second case is when objects of classes are not configured to show as folders; do not have the container flag set; and do not contain an attribute of the "Image" or "File" datatype.

Content represented as zero-byte files in WebDAV should be managed through the Website Interface or Administration Interface. If you open them on your computer, you will find nothing.

13.2. Setting up a WebDAV connection

The specific steps for establishing a WebDAV connection with your site vary across clients. The main information required is as follows:

- Your site's WebDAV address, such as "webdav.demo.ez.no"
- Your eZ Publish username and password
- The port number (usually 80)

Consult your site administrator if you are unsure about these parameters. The WebDAV address is different from your site address. For example, if your site address is "demo.ez.no", your WebDAV address might be "webdav.demo.ez.no".

Below, we explain the connection procedure for a WebDAV client on each of the Windows, Mac and Linux operation systems.

13.2.1. Windows Explorer with built-in WebDAV support

The Windows 2000, Windows XP, Windows Server 2003, and Windows Vista operating systems have built-in WebDAV support. This is accessed through the My Network Places interface. Once connected, you can perform file and folder operations on eZ Publish content nodes just as you would for local files and folders.

Warning

Although there is a Web Folders feature integrated into Internet Explorer (which can connect to your site via WebDAV), we recommend that you connect via My Network Places. We have found this to be a more stable connection method.

The procedure is slightly different for Windows Explorer on Windows Vista than for the other Windows versions. You will find Vista-specific instructions after the general procedure. To connect via WebDAV using the Add Network Place Wizard of Windows 2000, Windows XP or Windows Server 2003, follow these steps:

1. Access My Network Places, usually by clicking on the corresponding icon on the desktop.

2. Click on the **Add a network place** item under the **Network Tasks** menu on the left:

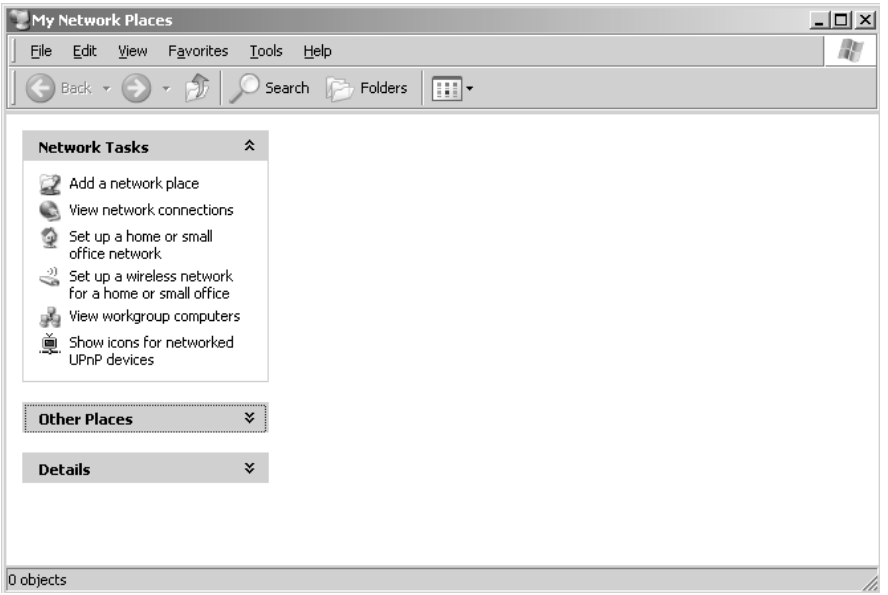


Figure 13.5. Add a network place under My Network Places

You will then be presented with the Add Network Place wizard welcome window. Click the **Next** button.

3. Select the "Choose another network location" service provider, then click the **Next** button.

4. Enter your site's WebDAV address. If the access port is a number other than 80, append the port number to the address following a colon (for example, "http://webdav.demo.ez.no:81" for port 81).

   When prompted, enter your eZ Publish username and password.

5. Choose a name for the connection. When set up, it will be listed by this name in My Network Places on your local computer. When done, click the **Next** button.

6. The wizard is now finished, and you may now open the WebDAV connection.

   For subsequent connection sessions, you can return to the My Network Places interface and select your eZ Publish WebDAV connection.

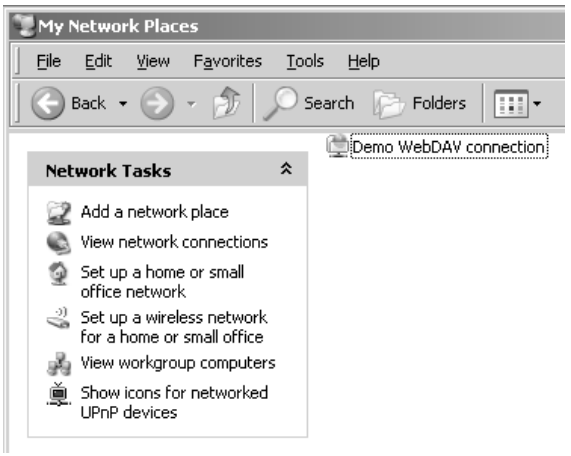


Figure 13.6. WebDAV connection in My Network Places

13.2.1.1. Windows Vista

In Windows Vista, the My Network Places interface has been renamed to Network. Starting the Add Network Place wizard differs slightly from the previous procedure for the other Windows distributions, and is outlined below:

1. Access Network, usually by clicking on the corresponding icon on the desktop.

2. Select the **Tools** menu, then the **Map Network Drive** option to bring up a connection wizard.

3. Click on the link called "Connect to a Web site that you can use to store your documents and pictures".

   The rest of the wizard steps are similar to the previous procedure for Windows 2000, Windows XP and Windows Server 2003. Proceed by filling in the connection information as previously described.

Tip

If prompted with the error "The folder you entered does not appear to be valid. Please choose another." when specifying the site's WebDAV address, try downloading the Windows Vista WebDAV patch. To do so, search for "web folders" at `http://www.microsoft.com/downloads/`.

13.2.2. Transmit for Mac OS X

For Mac OS X users, we recommend Transmit by Panic Software to connect via WebDAV to your site. The following procedure describes how to do this.

1. The initial window when you load Transmit is the connection window:

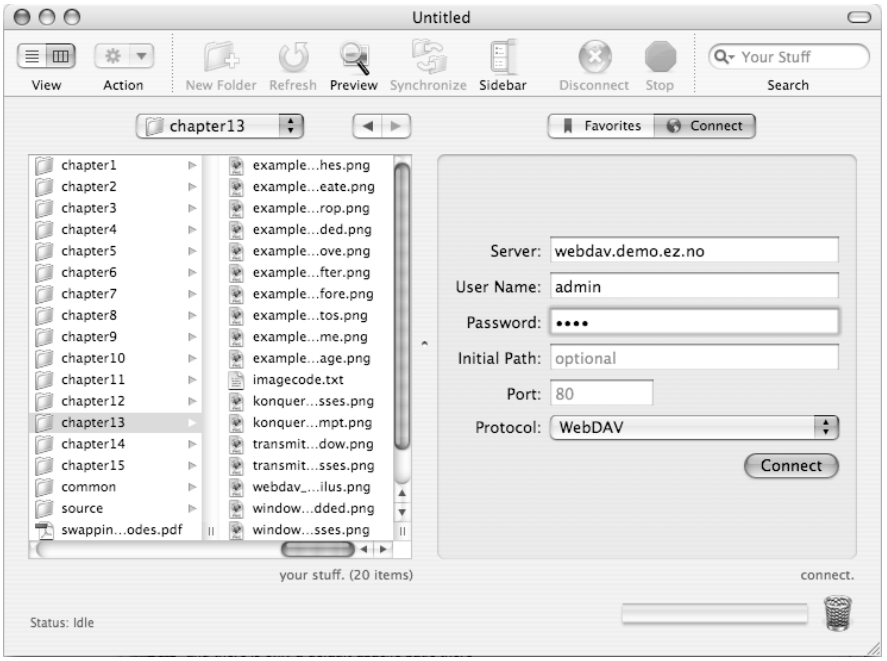


Figure 13.7. Initial connection window in Transmit for Mac OS X

Enter the connection information required. Be sure to select "WebDAV" from the **Protocol** dropdown list.

2. Click the **Connect** button.

13.2.3. Konqueror for Linux

The web browser Konqueror for KDE versions of Linux is also capable of serving as a WebDAV client. (Different WebDAV clients are available for other Linux distributions, such as Nautilus in the GNOME desktop environment.)

To connect to your site using Konqueror, follow the procedure below:

1. Type the address to the WebDAV connection in the **Location** bar - the same field in which you would type in a website address. However, instead of the "http://" prefix, use "webdav://", for example "webdav://webdav.demo.ez.no".

Append the port number following a colon, if it is a number other than 80.

2. You will be prompted for your username and password when you navigate into one of the siteaccesses. This differs from the other clients, in which you enter this information in the connection setup.

13.3. Default language for WebDAV

Content added to eZ Publish through WebDAV is created in the default language (see Section 12.2.3, “Default language”), which is specified in the "ContentObjectLocale" setting of *`site.ini`*. You can check the value of this setting in the **Settings view** interface of the **Setup** tab in the Administration Interface, as described in Section 3.5.4.1, “Viewing the settings of a configuration file”.

Tip

The "ContentObjectLocale" setting might be the same as the site locale ("Locale" setting), but it is sometimes different. In other words, your Norwegian siteaccess might have English as the default language, but Norway as the site locale (for country-specific display). In that case, content added will be created in English. These concepts were described in the previous chapter.

For multilingual sites with language-specific siteaccesses, you should navigate into the appropriate siteaccess before adding content, in order to ensure that the new object is created in the desired language. This is very similar to how you add translations in the Website Interface.

13.4. Content versioning with WebDAV

Documents in WebDAV are versioned. This means that if you overwrite a file in the WebDAV client, eZ Publish will create a new version of the document; you can roll back to the previous version later if the need arises. Versioning is only possible for image objects, multimedia objects, and file objects. In other words, versioning in WebDAV only works for objects represented as images or files, not for those represented as folders.

For versioning to work, uploaded files must have the exact same name as the destination file displayed in the WebDAV client. If you upload a file with the same name as an existing file, a new version of that existing object is created. Otherwise, a new object is created. (See Section 14.2.3, “Importing via WebDAV” for more information on versioning when uploading *`.odt`* and *`.doc`* files with the eZODF extension.)

WebDAV does not interfere with unpublished drafts of existing objects. Editing conflicts must still be addressed when the object in question is accessed in the Website Interface or Administration Interface.

13.5. Using WebDAV with your eZ Publish site

Recall that WebDAV is an alternative to the Website Interface or Administration Interface for some content management operations. This section outlines what can be done using WebDAV and how it relates to familiar content management operations. We also provide examples on how to perform these operations. Note that content type mapping between eZ Publish and WebDAV can differ from what is described here, depending on how your content classes are defined and the settings within `webdav.ini` and `upload.ini`.

For illustrative purposes, we include an example where we will upload images to an image gallery, rename the gallery, and then delete one of the images. We will also move the location of the gallery to a separate folder. The initial state of the gallery is shown in Figure 13.8, “Photo gallery before any modifications”, below. The example does not introduce any new concepts or interfaces and can be skipped if you feel that you have a good understanding of WebDAV and eZ Publish.

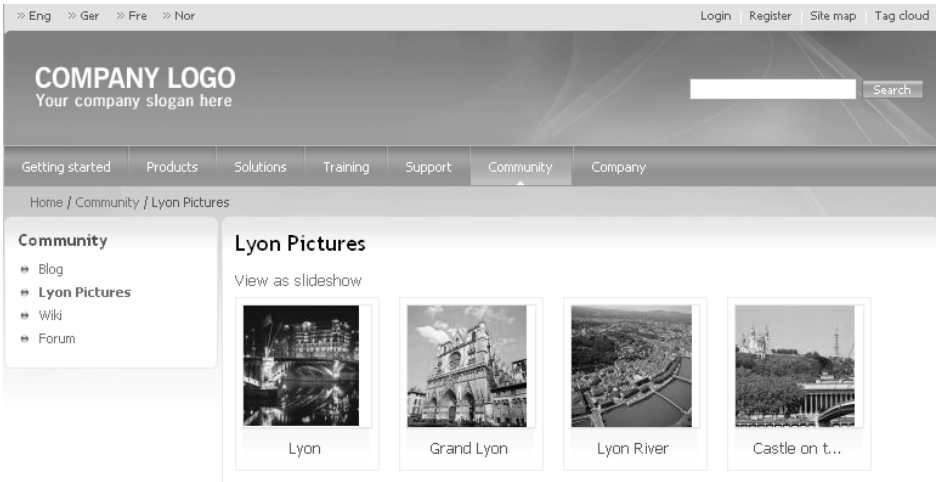


Figure 13.8. Photo gallery before any modifications

Throughout this section, Windows Explorer is used as the WebDAV client. It is assumed that you are connected and logged in with an account with Editor or Administrator user permissions. In addition, make sure that the Folders panel is showing on the left. Click the **View** menu at the top, expand the "Explorer Bar" item, then click the "Folders" item if it is not already marked.

13.5.1. Browsing via WebDAV

Once you have logged in through the WebDAV client, you will see the different siteaccesses, as shown in Figure 13.2, “List of siteaccesses in Windows Explorer”. Recall that the normal access control and language fallback mechanisms apply, and that eZ Publish content objects are represented as either folders, images, files or zero-byte files in WebDAV according to the list of criteria explained previously.

Navigation follows standard file browsing principles: double-click folders to view their contents (as if you were viewing their sub-items), and use standard **Back**, **Forward** and **Up one level** buttons in your WebDAV client as navigation aids.

You have to select a siteaccess and one of the main branches of the content node tree (for example, the Content or Media branch) before getting to the real content. Selecting a specific siteaccess by clicking its name corresponds to accessing the URL of the public or admin siteaccess, or clicking one of the language links at the top of the front-end of a multilingual site.

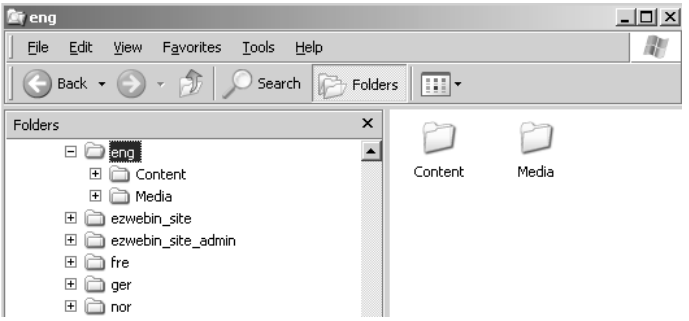


Figure 13.9. Content and Media branches via WebDAV

13.5.2. Creating content via WebDAV

In addition to importing from your local file system (described below), you can also create objects in eZ Publish through the WebDAV client. This is done similar to how you would create a file or folder in your file browser: by selecting "New > Folder" or a similar item from the File menu or from the pop-up menu accessed by right-clicking somewhere in the window. Generally, the creation feature is used simply to create folders.

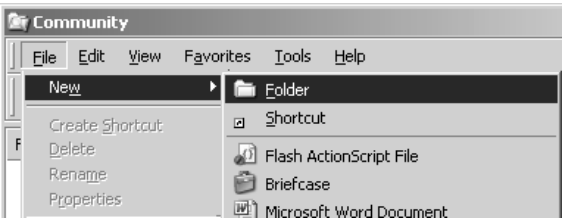


Figure 13.10. Create a new folder via WebDAV

Not all WebDAV clients have support for creating content objects. From the point of view of eZ Publish, dragging and dropping an object from another location into the WebDAV client is the same as creating an object via the WebDAV client.

The result of creating a new folder via WebDAV is similar to creating a new Folder object via the Administration Interface or Website Interface. A new object of the `Folder` class is created with the specified name. Your client might support setting other properties for the folder, but that will not affect eZ Publish or the new content object.

Tip

Your site might be configured to use another class as the main folder class, such as the `Image gallery` class.

Setting up folders like this is useful to create a structure into which you can drag and drop other content. It is typically faster if you are already working in your WebDAV client, and the mapping goes to the desired container class in eZ Publish. However, if you need to create a container of a different class, such as a `Documentation page`, `Frontpage`, `Blog` or `Forum`, you have to log in to the Website Interface or Administration Interface and perform the operations there.

Creating a new file via WebDAV is similar to creating a new File object via the Administration Interface or Website Interface. This triggers the same behavior as uploading a blank (zero-byte) file of that particular file type. For example, if you create a new text file called "Skien.txt", the default behavior is to create an object of the `File` class with "Skien.txt" as the Name attribute and the empty *`Skien.txt`* file as the File attribute.

13.5.3. Importing content via WebDAV

WebDAV import means to upload content to an eZ Publish site through a WebDAV client. This operation is equivalent to clicking either the **Create here** or **Edit** button in one of the eZ Publish interfaces, filling in the appropriate fields in edit mode, and clicking the **Send for publishing** button.

Importing via WebDAV triggers either the creation of a content object or the creation of a new version of an existing object. The class is automatically assigned based on configuration settings as previously described.

To import a file, first ensure that the target location is displayed in the WebDAV client. (Expand folders by double-clicking.) Then drag and drop a file from a location on your computer to the desired location in the WebDAV client window. Remember that choosing the right siteaccess is very important, especially for multilingual sites.

Tip

When working on Windows, it is practical to have two Windows Explorer instances open at the same time, one for the source and one for the target location.

You can select multiple items in several ways. First, you can use the mouse to drag a frame around the items. Second, you can select a range of items in a listing by clicking on the first item, then holding the Shift key while clicking on the last item. Finally, you can select specific items by clicking on the first item, then holding the Ctrl key while clicking on each of the additional items.

13.5.3.1. Example: uploading images

Let us start by navigating to an image gallery where we are going to upload some images. In this case, navigate to the "eng" siteaccess, then to Content -> Community -> Lyon-Pictures. The partially hidden window in Figure 13.11, “Four photos to add to the existing gallery” below shows that there are already four images from our demo installation. The selected Explorer window shows the folder containing the images we are going to add to eZ Publish:

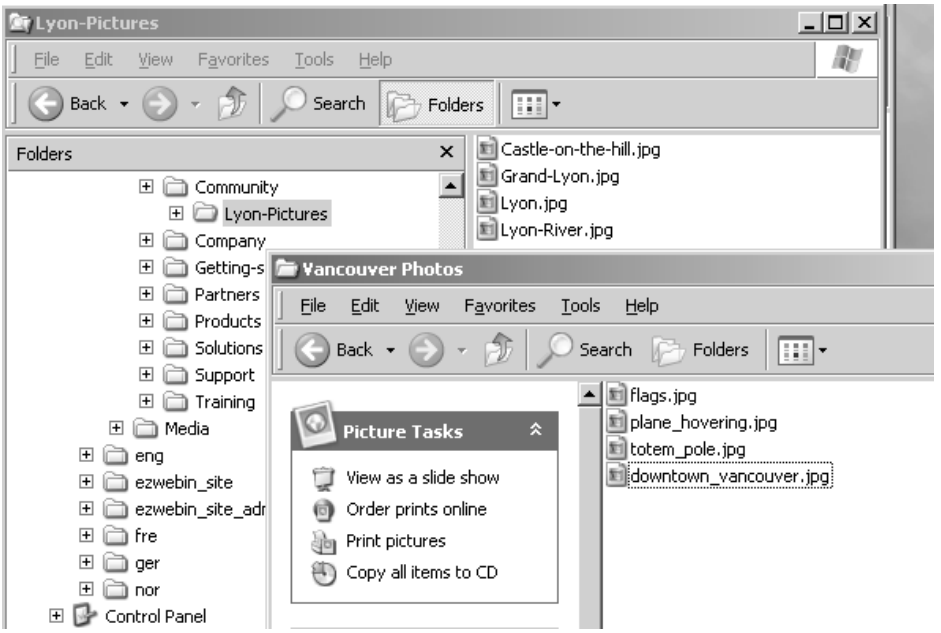


Figure 13.11. Four photos to add to the existing gallery

We are now ready to drag and drop the images. Simply select all the images you wish to upload and then drag them to the WebDAV window.

eZ Publish will automatically create Image objects for these image files. The images are now visible in the "Photos" gallery on the public siteaccess:

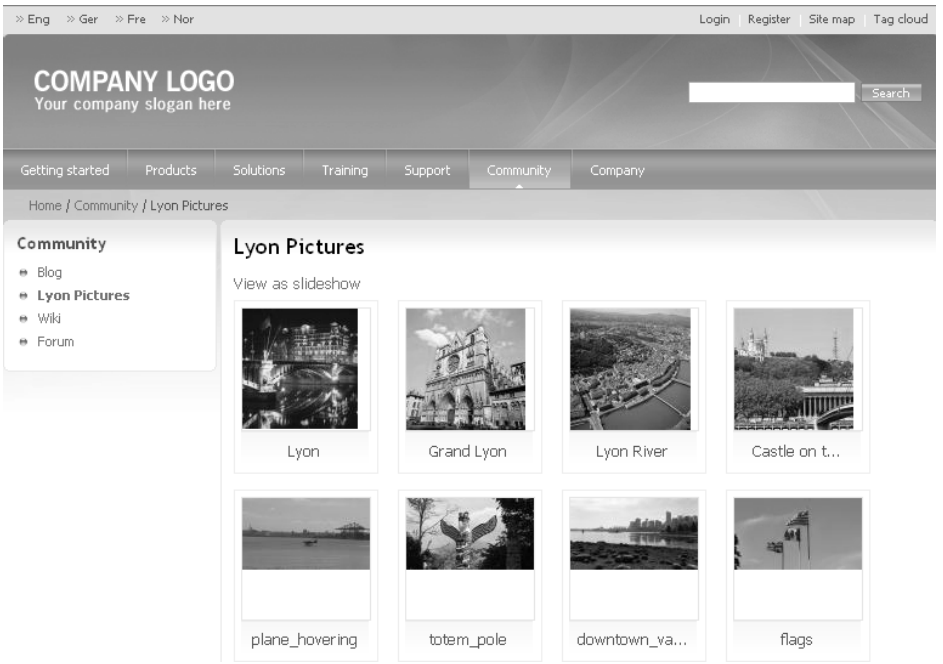


Figure 13.12. Images added via WebDAV to the image gallery

13.5.4. Exporting content via WebDAV

WebDAV export means to download content from an eZ Publish site using a WebDAV client. This operation is equivalent to navigating via a link to a file, right-clicking the link, and clicking the "Save as..." option in your browser. Or, for an image, you would locate it, right-click it, and click the "Save image as..." item in your browser.

Exporting is only available for content represented as images or files by WebDAV. In other words, you cannot export content represented as folders or zero-byte files.

Tip

Images are exported according to their original upload size. Keep in mind that this may be larger than expected, as images can be down-sized for display on the website.

To export a file or image, simply drag and drop it from the WebDAV client window onto your desktop or a location on your computer in your file browser. This is the reverse operation of importing.

13.5.5. Other content management tasks with WebDAV

This section describes how to work with existing content, explaining the tasks of renaming, moving, copying and deleting content.

Warning

Often, a more appropriate operation than moving or copying content would be to assign it to a secondary location (cross-publishing). However, you cannot assign a secondary location to an object using WebDAV. For that purpose, log in to the Website Interface or Administration Interface.

13.5.5.1. Renaming content via WebDAV

Renaming existing content via WebDAV is the equivalent to editing the object using the **Object Edit Interface** or the **Content Editing Interface** and modifying its "Name" or "Title" attribute. Renaming can be particularly useful for images, since the image captions displayed in galleries are pulled from the image's Name attribute, and digital cameras usually create filenames (for example, "DC409.jpg") that are not very descriptive.

The effect of renaming is slightly different depending on whether the content is represented as a folder or a file in WebDAV. This is because files are listed with filename extensions like ".txt" or ".jpg". When renaming a file, the file extension is appended to the object's name. As an example, consider a Flash object whose Name attribute is "Skien tour". It is represented in WebDAV as "Skien tour.swf". If you rename this object in WebDAV to "Skien hiking tour.swf", the ".swf" file extension will also be included in the Name attribute. Therefore, you should simply rename it to "Skien hiking tour".

Renaming usually follows these steps:

1. Locate the item to be renamed by navigating to it in the WebDAV client.
2. Right-click on the item, and click "Rename".
3. Enter the new name.
4. Press Enter on your keyboard to apply the new name.

13.5.5.1.1. Example: renaming an object

Let us rename the gallery from our example, giving it a more descriptive name, "Office city photos" (since eZ has offices in Vancouver and Lyon). Assuming we are still looking at the gallery, browse one node up by clicking the "Community" folder in the left panel

of Windows Explorer. You should then see a list of folders representing sub-items of the "Community" node.

Right-click on the Photos folder and select the "Rename" option. Then, type in the new name, "Office city photos". The image gallery now has the new name:

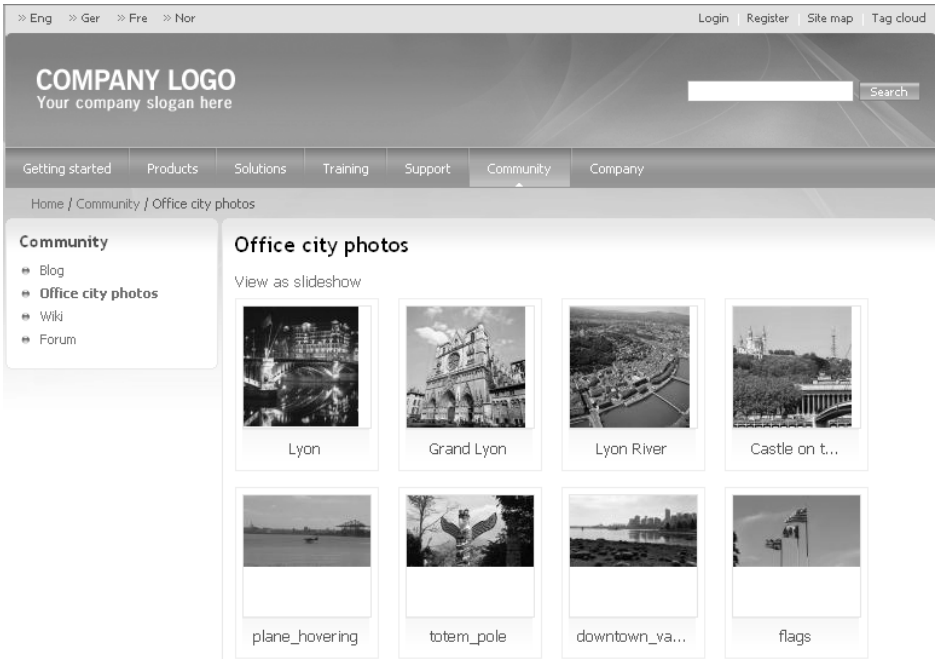


Figure 13.13. Rename an image gallery via WebDAV

13.5.5.2. Copying content via WebDAV

Copying an object through WebDAV creates a duplicate object, just like when you click the **Copy** button in the Website Interface or Administration Interface. In other words, there will be a new content object with identical content to the original one, and it will be encapsulated within a new node at the given location. This is not the same as cross-publishing (which creates two or more references to a single content object).

In WebDAV, however, the copy action is always a "copy subtree" action. If the object's node has sub-items, all of these sub-items are always copied. This is in contrast to the regular eZ Publish interfaces, where you can only copy a subtree in the Administration Interface, and only if you select the corresponding option via the pop-up menu. Copying a subtree can be a resource-intensive operation, and can negatively impact your server performance if there are many sub-items. As a result, check whether the object to be copied has any sub-nodes and be aware of the possible consequences.

If your WebDAV client supports copying, this is usually done using the same method as copying files on your operating system:

1. Locate the item to be copied by navigating to it in the WebDAV client.
2. Right-click on the item and click "Copy".
3. Navigate to the new location.
4. Right-click on an empty space in the right panel and select the "Paste" option.

13.5.5.2.1. Example: copying an object

Returning to our gallery example, suppose that we would like to copy one of the recently uploaded images to the media library, in order to easily re-use it in future articles and other content objects. In this case, we will create a copy of the image "totem_pole".

First, locate the image located at Content -> Community -> Outdoor-office-photos. Right-click the image, then select the "Copy" option. Next, navigate back up past the "Content" folder and into the "Media" folder. Then double-click the Images folder. Finally, right-click on an empty space in the right panel and select the "Paste" option.

Access the Administration Interface **Media library** tab in your web browser, navigate to the "Images" node, then check the result in the **Sub items** window:

Figure 13.14. Sub items window - image copied to the media library via WebDAV

13.5.5.3. Deleting content via WebDAV

Content removal is trivial using WebDAV. It corresponds to clicking the **Remove** button in the Website Interface or Administration Interface for a target node. This is typically useful if you have uploaded an image that turned out to be of poor quality. Note that this operation applies to entire content objects, including all their versions and translations

and sub-items (if any). It is not possible to delete specific versions of objects through the WebDAV interface. Version management is done through the Administration Interface.

Deleting an object in a WebDAV client moves it to the eZ Publish trash. However, as for ordinary removal through eZ Publish, you should be sure that this is what you want to do.

The effect differs depending on whether the object has one or multiple locations, and, if multiple locations, the location where you attempt to remove it. As is the case in the Website Interface, you cannot tell whether an object in question has multiple locations, so be aware of the possible consequences.

When the object has only one location, the object is removed. If you are deleting a main location of an object with multiple locations, eZ Publish will arbitrarily assign a secondary node as the new main node before removing the targeted node. The new main location might belong to a section different from the old main node (see Chapter 4 for more information). If you are deleting a secondary location, only that location and its sub items are deleted. The latter is the same behavior as if you were to mark the corresponding checkbox(es) and click the **Remove selected** button in the **Locations** window of Administration Interface.

Deleting usually follows these steps:

1. Locate the item to be removed by navigating to it in the WebDAV client.

2. Right-click on the item, then click "Delete".

   Alternatively, left-click it and press the Delete button on your keyboard.

13.5.5.4. Moving content via WebDAV

Some WebDAV clients support dragging and dropping files within the WebDAV interface. This will move a node, assigning it (and its sub-items) to a different location, similar to clicking the **Move** button in the Website Interface or Administration Interface, and selecting a new location for the node.

Moving usually follows these steps:

1. Locate the item to be moved by navigating to it in the WebDAV client.

2. Grab the item with your pointing device and drop it into the new location.

13.5.5.4.1. Example: moving a node

Suppose that the company responsible for the site is based in Lyon and Vancouver, so we would like to show the image gallery, which highlights scenery from both locations, under the Company node. Therefore, we will move the gallery to the more appropriate

location. To do so, drag and drop the "Office-city-photos" folder from the right panel, to the "Company" folder in the left panel:

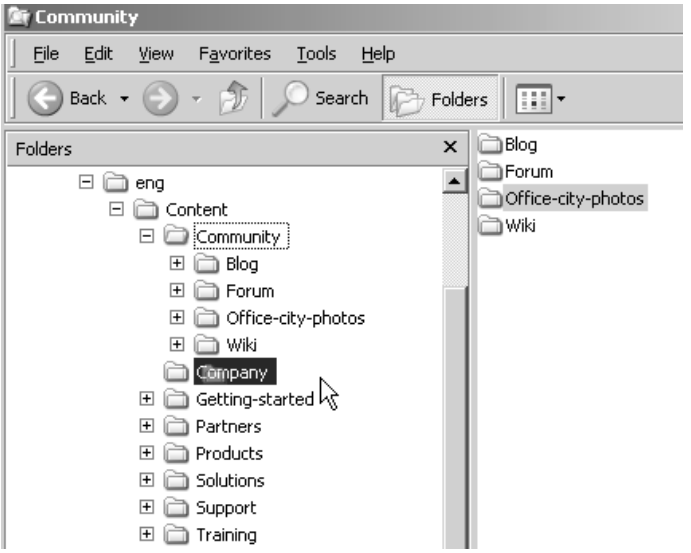


Figure 13.15. Moving a node via WebDAV

The "Office city photos" image gallery is now available under the Company node:

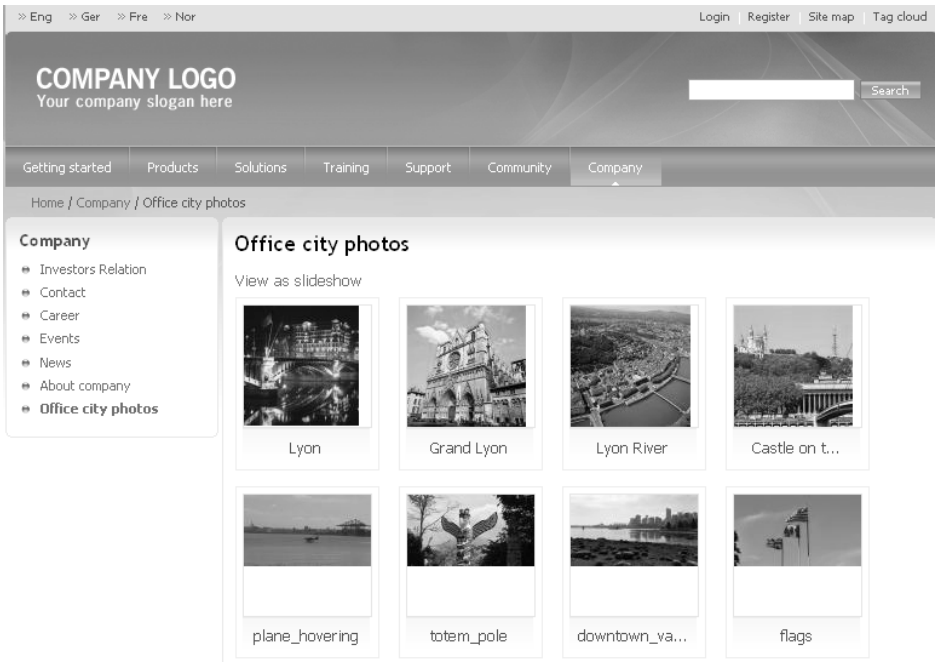


Figure 13.16. Photo gallery after modifications

13.6. Summary

Using a WebDAV client, you can access an eZ Publish site via the WebDAV protocol in order to view objects in the content node tree as represented by files and folders. You can thus browse through your site in a similar manner to browsing through the file and folders on your computer.

In your WebDAV client, you need to supply your site's WebDAV address, the port number (usually 80) and a valid username and password for a user account on your site.

Objects are displayed in the WebDAV client as files, folders or images, depending on the mapping settings. Similarly, other mapping settings determine how different uploaded file types create objects of various classes.

Many of the same content management operations that can be performed in the Website Interface and Administration Interface can be performed via WebDAV. Creating, renaming, copying, deleting, and moving objects and nodes can all be done via WebDAV. WebDAV is particularly suited for batch file and image uploading and exporting, where you drag and drop between your local file system and the web server.

Chapter 14. OpenDocument Text and Microsoft Word documents

The *eZ Open Document Format* (eZODF) extension is an eZ Publish plugin, included and enabled in default installations. The import and export features provided by eZODF enable you to share and work on site content during times when you do not have access to the site. This chapter explains how to convert between objects in eZ Publish and OpenDocument Text and / or Microsoft Word document format using the Website Interface, the Administration Interface or a WebDAV client.

This chapter is relevant to all content managers who would benefit from working with site content in a word processor. Each eZODF operation is described along with its equivalent action in the Website Interface and Administration Interface.

Recall that:

- *WebDAV* enables you to drag and drop files representing content objects between an eZ Publish site and your local file system.

- An eZ Publish *extension* is a plugin that provides additional custom functionality, without requiring major changes to your existing system. Extensions are activated and deactivated (see Section 3.6, “Extension management”) through **Available extensions** window (also known as the **Extension configuration** interface). To access it, click the **Extensions** link in the left menu of the Administration Interface **Setup** tab.

- An eZ Publish *INI file* is a configuration file that controls the behavior of a specific part of the system. See Section 3.5, “The configuration model”. The main configuration file is `site.ini`. All setting modifications should be done in a global or siteaccess-specific override of the configuration file in question. You can check the value of a particular setting within the **Settings view** interface as described in Section 3.5.4.1, “Viewing the settings of a configuration file”.

In this chapter, you will find information about the following:

- Step-by-step procedures for working with the eZODF extension via the Website Interface, the Administration Interface and WebDAV clients.

- Which formatting styles can be imported and exported by the eZODF extension.

- How importing content via the eZODF extension affects editing conflicts, multilingual content and objects with multiple locations.

Although this chapter provides basic instructions on how to install the eZODF extension, fine-grained configuration details and server issues (especially when importing and exporting Microsoft Word documents) are beyond the scope of this book. Interested readers should consult the documentation on the eZODF extension at http://ez.no/doc/extensions/odf_import_export. Also, this chapter assumes that you have general knowledge about word processors and can work with word processor documents.

All of the topics are relevant to content managers, unless you are not going to be working with Microsoft Word documents at all. In that case, you can skip that section. The bulk of this chapter assumes that you will be importing and exporting OpenDocument Text (`.odt`) documents.

14.1. The eZ Open Document Format extension and OpenDocument Format documents

OpenDocument Format is an Open Source file format that defines the storage of text, spreadsheets and other document types. *OpenDocument Text* files implement this format for word processor documents. These files can be created and edited in OpenOffice.org Writer for Windows and Linux, and NeoOffice Writer for Mac.

The terms related to the eZODF extension are quite similar, and are summarized in the table below.

Table 14.1. Terms related to the eZODF extension

OpenDocument Format (abbreviated as "ODF")	A file format used by, among other things, word processor files.
OpenDocument Text	Refers to the word processor files that store content in OpenDocument Format (ODF). These files have the file extension `.odt`.
OpenOffice.org Writer, NeoOffice Writer	Word processor applications that read files that use OpenDocument Format, including OpenDocument Text files.
eZ Open Document Format (eZODF) extension	An eZ Publish extension that enables you to import and export OpenDocument Text files to and from an eZ Publish site.

OpenDocument Text (`.odt`) documents are capable of storing all of the typical formatting characteristics applied in word processors. The eZODF extension preserves most of that

formatting when `.odt` content is imported to and exported from eZ Publish (as is explained in Section 14.1.3, “Datatypes and formatting supported by eZODF”).

At the time of publication, the latest version of the eZODF extension is version 2.1.

14.1.1. eZODF usage scenarios

The eZODF extension is useful in many scenarios. The following list contains some of the more common scenarios:

- When a content manager needs to create or edit site content, but cannot access the site. For example, he or she can type up a news release in the familiar OpenOffice.org Writer application on a laptop while on a plane ride. Upon landing, he or she can quickly import the file to create an eZ Publish Article object once internet access is found.

- When site content needs to be worked on by people who do not have a user account on the site (out of choice or necessity). This is often the case for translation work or with content that is not yet published (see Chapter 6, *Node visibility*).

- To facilitate multi-channel publishing. For example, a documentation page could be created on the site and then later re-purposed as a brochure. Exporting the contents of the page to a word processor document speeds up the brochure publishing process.

14.1.2. eZODF requirements

The eZ Open Document Format extension works with eZ Publish versions 3.8 and higher. It is included in installations 3.9 and higher (and also available as a separate extension).

In order to import and export OpenDocument Text documents, the following is required:

- The eZODF extension must be uploaded and activated on the web server. You can verify this in the **Active extensions** window in the Administration Interface. (Check that "ezodf" is listed and enabled). Refer to Section 3.6, “Extension management” or consult your site administrator.

- Your user account must have the appropriate access permissions, including a policy granting access to the "ezodf" module. This is included by default in the Editor role. Refer to Chapter 5 or ask the person in charge of user management.

- You must have installed the desktop software capable of creating and editing `.odt` documents. The OpenOffice.org office suite is freely downloadable at `http://www.openoffice.org`, while the NeoOffice office suite is freely downloadable at `http://www.neooffice.org`. Full requirements for these two programs are available on their respective websites.

Tip

In order to import and export Microsoft Word documents, OpenOffice.org must be installed on the web server and configured by your site administrator according to the instructions in the eZODF extension documentation.

14.1.3. Datatypes and formatting supported by eZODF

Contents of the following datatypes can be imported and exported via eZODF:

- Text line
- Text block
- XML block
- Image
- Date
- Date and time
- Matrix

When working with "Date and time" attributes (see Section 7.2.1, “Date and time datatype”), you must ensure that they strictly follow the "dd/mm/yyyy hh:mm" format. For example, to import the date March 23, 2008, 1:00pm, you would enter "23/03/2008 13:00" into the appropriate `.odt` section.

Note that the "Keywords" and "Checkbox" datatypes are not supported by eZODF. As a consequence you have to manually mark the **Show children** and **Enable comments** checkboxes for imported container objects like articles and documentation pages. You must also manually enter tags after the import operation for attributes that use the "Keywords" datatype (for more information on tags, see Section 9.1.1, “Keywords datatype” and Section 9.1, “Content relevance with tag clouds”).

The following formatting styles are preserved for rich text content (in "XML blocks") when imported or exported:

- Normal text
- Bold and italic text spans (import only)
- Text links

- Custom tags (such as a factbox)
- Numbered and bulleted lists (import only, one level)
- Headers (multiple levels for export only)
- Tables (without collapsed rows or columns)
- Embedded images (import supports alignment, size, and image captions)

14.1.4. Mapping OpenDocument Text content to content attributes and classes

In eZ Publish, a content object is defined by an ordered sequence of content class attributes. In contrast, OpenDocument Text documents are structured by *.odt sections*. The latter is not to be confused with the eZ Publish section concept, described in Chapter 4. Importing and exporting via the eZODF extension moves content between these content structures, and to ensure that the content ends up in the desired place, mapping is required.

Mapping is handled by eZODF based on configuration settings in the extension-specific `odf.ini` file. Among other things, the settings in this file specify the default class for imported content; the content classes supported; and how the attributes of these classes map to `.odt` sections.

Tip

Because the `odf.ini` file is not part of the standard eZ Publish settings, it is not available in the **Settings view** interface of the Administration Interface (see Section 3.5.4, “Settings view interface”). In other words, in order to view and modify this INI file, you must directly access the web server's file system. For more information, contact your system administrator.

When importing content, the content class of which the new object will be created is determined by the named sections of the imported file. In OpenOffice.org Text documents, sections are configured via the **Section** interface accessed from the **Insert** menu. For example, the default configuration is that if the imported file has sections called "name", "short_description" and "description", it is imported as an object of the `Folder` class. The contents in the `.odt` sections will be placed in the corresponding attributes. The order of the sections in the document does not need to match the order of the attributes in the class. What matters is that the section names correspond to the attribute identifiers of the class.

By default, the `Article`, `Folder`, `Image` and `Documentation page` classes are supported. We include the default mapping for these classes below, as you would need this information to configure document sections correctly (as described in Section 14.1.5,

"Configuring document sections in OpenDocument Text"). Each block name corresponds to a class name. Each entry has the name of the attribute on the left in square brackets and the name of the `.odt` section to the right.

```
[article]
Attribute[title]=title
Attribute[intro]=intro
Attribute[body]=body
Attribute[image]=image

[folder]
Attribute[name]=name
Attribute[short_description]=short_description
Attribute[description]=description

[image]
Attribute[name]=name
Attribute[caption]=caption
Attribute[image]=image

[documentation_page]
Attribute[title]=title
Attribute[body]=body
```

If the sections of an imported document do not fit any of the mapping definitions above, the file will import to an object of the default class, which is usually the `Article` class. In this case, the title of the object is derived from the file name and the "Body" attribute (if it exists) is populated with the contents of the document.

Tip

In eZ Publish 4.1, the eZODF extension will be enhanced to improve the mapping of content in OpenDocument Text and especially Microsoft Word documents to eZ Publish content attributes and classes. In addition to using `.odt` and `.doc` sections as described in this book, the enhanced extension is also compatible with `.odt` and `.doc` styles to segment and subsequently map content for both file formats. See the extension documentation at http://ez.no/doc/extensions/odf_import_export for full information.

14.1.5. Configuring document sections in OpenDocument Text

A section in an OpenDocument Text document is visually displayed surrounded by thin borders. When the cursor is positioned inside a section, its name is visible at the bottom right of the window:

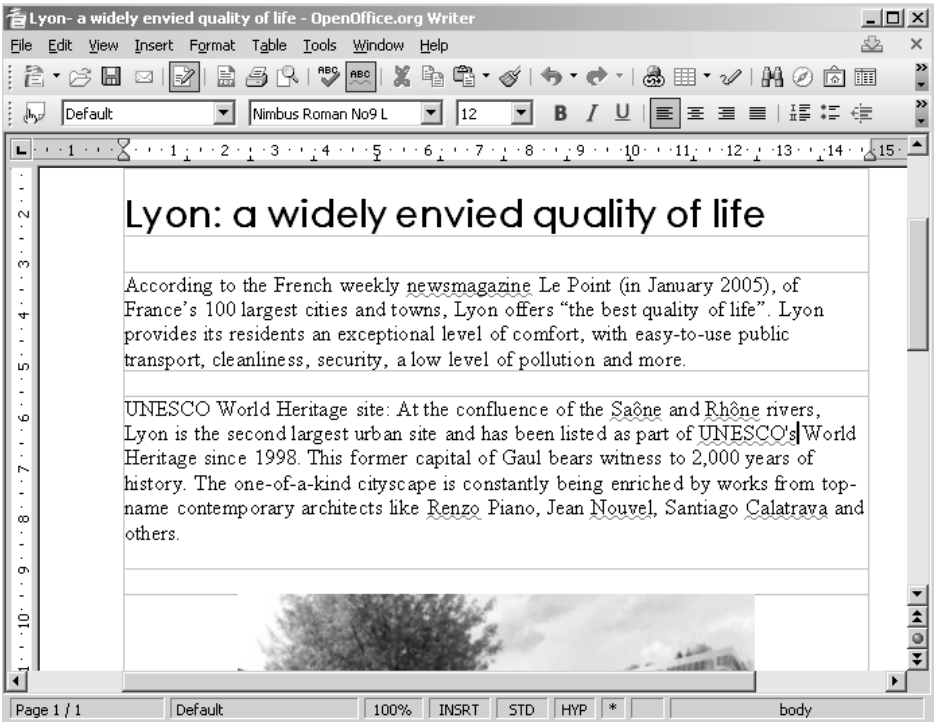


Figure 14.1. Sections in an .odt file - current section name in bottom right

The following procedure illustrates how to create an `.odt` section in OpenOffice.org Writer. The details are similar for NeoOffice Writer. We assume that you start with a newly created, blank `.odt` file.

1. Open the "Insert" menu and select the "Section..." item. This will open the window shown below:

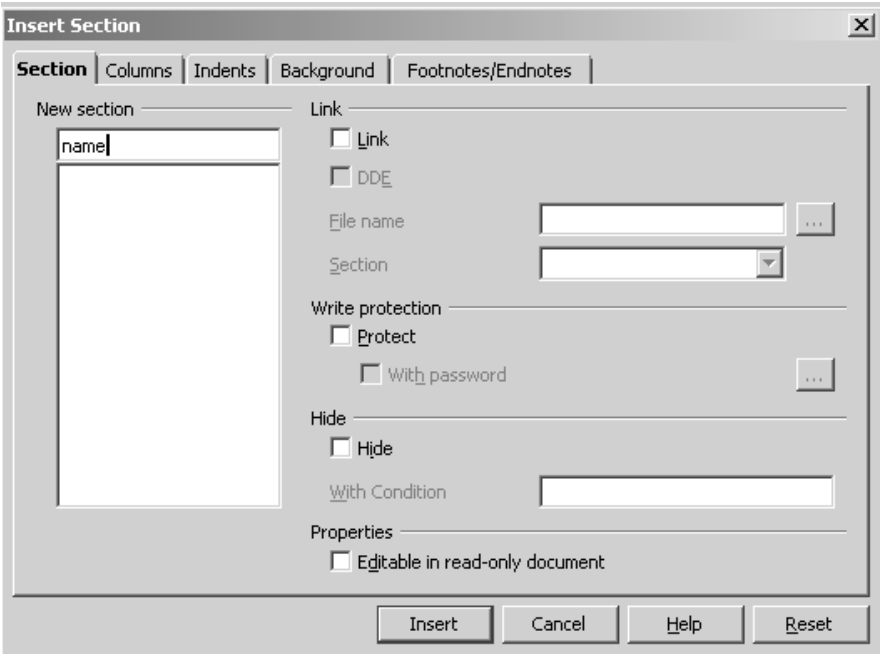


Figure 14.2. Insert Section window

2. Depending on the content class for which you are setting up sections, enter a section name corresponding to a valid attribute identifier, then click the **Insert** button. For example, enter "name" for an attribute of the `Folder` class. Consult the previous listing to find the correct identifiers.

3. Repeat the above steps to create sections corresponding to the other attributes of the class, such as "short_description" and "description" for a folder.

After configuring the document sections, you can fill in values to be imported to the corresponding attributes.

Tip

If you do a lot of importing, you can save some time by creating class-specific OpenDocument Templates (with the *.ott* extension), and simply create a new document based on one of the templates rather than setting up the sections manually each time.

14.1.5.1. Naming sections in .doc files

In Microsoft Word, you can create sections, but it is not possible to name them. When *.doc* files are imported to eZ Publish, the OpenOffice.org installation on your server

converts the file to an `.odt` file before it is processed by eZ Publish. Sections in `.doc` files are automatically given the names "Section 1", "Section 2" and so on by OpenOffice.org in the converted `.odt` file. As a result, the order of sections in your `.doc` file is important. These default section names are mapped to content class attributes based on the configuration in the `odf.ini` file. Additional configuration and custom mapping is also required. Consult your site administrator for information on the section order and naming.

14.2. Importing OpenDocument Text files

OpenDocument Text import can be used to create objects or new versions of existing objects. This corresponds to clicking the **Create here** or **Edit** button in eZ Publish, entering (or modifying) the desired content in edit mode, then clicking the **Send for publishing** button. The following sections explain how to perform both operations in the Website Interface and Administration Interface.

Tip

Imported content is published immediately, and site visitors can see the object right away. This corresponds to the illustration shown in Figure 1.8, “Life cycle of a content object”.

In order to make edits (to correct formatting issues, for example) in uploaded content before it is publicly available, you can import it to a restricted section (see Chapter 4, *Sections* and Section 5.8, “Example: Creating a protected area”), make your changes, and move it to the desired location. This is only possible when creating objects, not for updating existing ones. Imported content also follows visibility status propagation (see Chapter 6, *Node visibility*). Therefore, if an article is created beneath a hidden folder, the article will also be hidden (by superior).

14.2.1. Importing to a new object

When you upload a new `.odt` file, eZODF creates a new draft object of the appropriate class, pulling the contents from the `.odt` document into the matching attributes, and then publishes the draft. Upload is done through an import form accessed via either the Website Interface or the Administration Interface.

14.2.1.1. Creating an object via the Website Interface

The following procedure illustrates how to import an `.odt` file via the Website Interface, creating and publishing a new content object. We assume you are logged in with an account that has the appropriate access permissions.

1. Navigate to the location for your new content. This should be a container node.

2. Click the **Import** button on the right side of the **Website Toolbar**:
 .

3. This will open the import form:

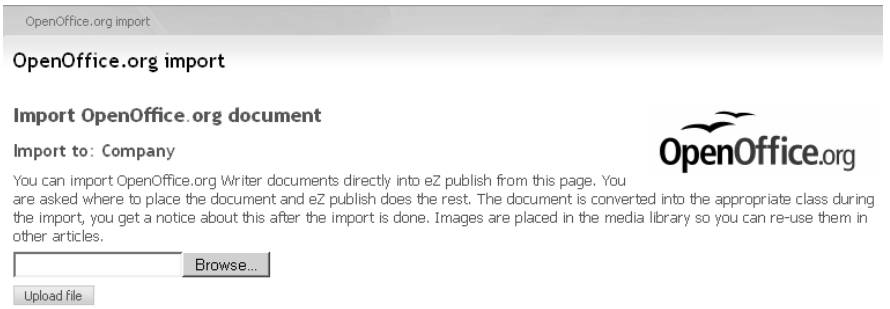


Figure 14.3. Website Interface eZODF view before importing

Click the **Browse** button and select the `.odt` file to import. Then, click the **Upload file** button.

4. eZ Publish will create the object under the node that you were viewing, as seen in Figure 14.4, “Website Interface eZODF import complete” below. You can then view the object that was just created (by clicking its title), create another object (by clicking the **Import another document** link or the **Upload file** button), or browse to somewhere else on the site.

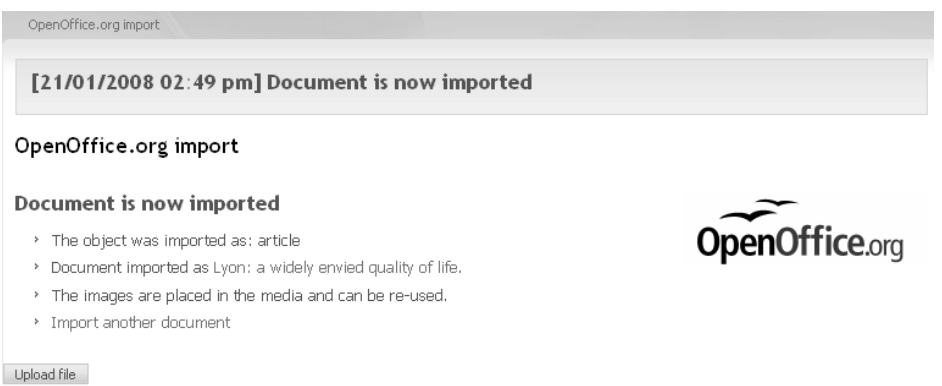


Figure 14.4. Website Interface eZODF import complete

14.2.1.2. Creating a new object via the Administration Interface

The Administration Interface uses the same import form as described above. It is opened from the context-sensitive pop-up menu for the target parent node. Click the "Import OpenOffice" item under the "OpenOffice.org" sub-menu:

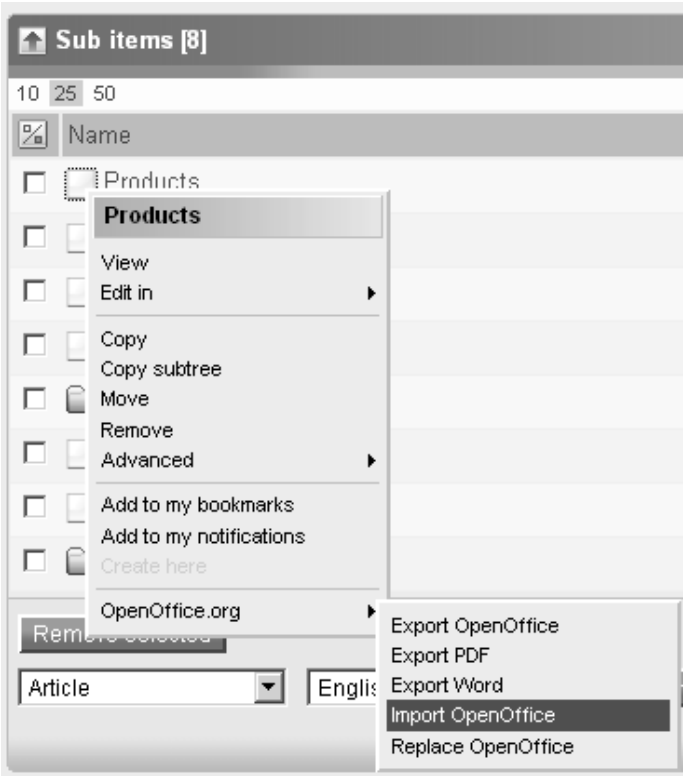


Figure 14.5. Pop-up menu: Import OpenOffice

Tip

You can also directly access the import form in both the Website Interface and the Administration Interface by accessing the system URL and appending "ezodf/import" as index parameters. For example, the complete URL for the form would be `http://www.example.com/ezodf/import` or `http://www.example.com/ezwebin_site_admin/ezodf/import`.

14.2.2. Importing a new version of an object

New versions of existing content objects can also be created with the contents of an `.odt` file. In this case, the eZODF extension takes care of creating a new draft version of the target object, pulling the contents from the `.odt` document and into the matching attributes (as before), then publishing the new draft. It also sets the status of the previously published draft to "Archive".

Tip

When creating new versions of multilingual objects, importing a file will create a version in the last edited language. Only via WebDAV is it possible to import a new translation of an existing object in a language other than the last edited language. This is described in the following section.

Warning

If you are creating a new version of an object that has one or more unpublished drafts, the system does not warn you about an editing conflict. It creates a new draft with the next version number, leaving the existing unpublished draft in the system.

Keep in mind that for the attribute values to be properly updated, the `.odt` sections in the document must match those in the content object according to the mapping settings. If there is a mismatch, some of the attributes will not be updated. The new version would thus have some attributes as imported from the file and others as copied from the previous version of the object. This occurs in both the Website Interface and Administration Interface.

Tip

To fix a small typographical error, it is more convenient to edit the object as is normally done through the Website Interface or Administration Interface. Creating a new version of an object from an `.odt` document is more suitable when making more significant changes.

14.2.2.1. Importing a new version of an object in the Website Interface

Importing a new version of an existing object using the Website Interface is identical to the procedure described above, except that in this case you click the **Replace** button on

the right of the **Website Toolbar** for the object in question (instead the **Create** button for the target parent node):  .

14.2.2.2. Importing a new version of an object in the Administration Interface

Uploading a new version of an existing object in the Administration Interface is identical to the description in Section 14.2.1.2, “Creating a new object via the Administration Interface” except that in this case you select the "OpenOffice.org" menu item, then click the "Replace OpenOffice" item.

14.2.3. Importing via WebDAV

Importing files via WebDAV is a simple drag-and-drop operation in a WebDAV client, as described in Chapter 13. If the eZODF extension is active, `.odt` files can be uploaded to eZ Publish via WebDAV and be recognized as content of the supported content classes by the named sections.

Keep in mind that filename extensions are included when updating content through WebDAV. If the file name does not exactly match a file on the eZ Publish site, a new object is created. To compensate, you should remove the extension on the file on your local system for it to create a new version of the existing object.

14.3. Exporting to OpenDocument Text files

OpenDocument Text export can be used to download objects published on eZ Publish websites. While this is similar to copying and pasting text from a website to a local file, the export function preserves section names (enabling subsequent re-import), and preserves formatting characteristics (such as headers, lists, etc). The following sections explain how to export from objects in the Website Interface and Administration Interface.

All objects in eZ Publish can be exported to OpenDocument Text files. When exporting, eZODF will create a new `.odt` file, pull the attribute values from the published version of the object into the matching `.odt` section, then prompt you to download the `.odt` file. But, keep in mind that not all datatypes are supported. As a result, objects of certain content classes are not well suited to be exported to an `.odt` file. For example, if you were to export an object of the `File` class, you would get a document with the title of the file, but the file itself would not be attached.

OpenDocument Text files can be exported using either the default Openoffice.org styles or using a custom OpenOffice.org Writer template file. This is set in the `odf.ini` configuration file. By default, an eZ Publish custom template is used. You can define custom styles for formatting such as tables, different header levels, lists and images. You

can also set a document header and footer, which will apply to all exported `.odt` documents. In order to do so, provide a template file to your site administrator for upload to the web server.

14.3.1. Exporting via the Website Interface

In the Website Interface, you can export content objects of those classes for which there is defined import mapping by clicking the **Export** button on the right side of the **Website Toolbar**:  . You will then be prompted to save the `.odt` file representing the object to your local file system.

For objects that belong to classes that have no defined import mapping, you have to access the "export" view of the "ezodf" module by directly loading the URL (as described in the "Tip" in the next section).

Tip

To export multilingual objects in a specific language from the Website Interface, browse the language-specific site interface to export the latest translation in that language. In other words, browse to the object that you wish to export in the desired language, then click the **Export** button on the right side of the **Website Toolbar**.

14.3.2. Exporting via the Administration Interface

The Administration Interface enables you to export any content object from the context-sensitive pop-up menu for the target node. Click the "Export OpenOffice" item under the "OpenOffice.org" sub-menu. You should then be prompted to download the `.odt` file representing the object that you are exporting.

Tip

When exporting objects through the Administration Interface, the system exports the displayed translation in that siteaccess, according to the availability and fallback system (see Chapter 12, *Working with multilingual sites*). To export specific translations of objects, you must browse the corresponding site interface in the Website Interface.

Tip

You can also access the export operation directly in both the Website Interface and the Administration Interface by accessing the system URL and appending "ezodf/export" as index parameters. For example, the complete URL might be:
`http://www.example.com/ezodf/export` or
`http://www.example.com/ezwebin_site_admin/ezodf/export`.

On multilingual sites, the exported translation will depend on which siteaccess the URL contains. In this "export" view, you must use the **Browse** interface to select the object to export.

14.3.3. Exporting via WebDAV

Exporting OpenDocument Text files is possible in WebDAV under two circumstances:

- You are exporting an `.odt` file that was just imported in the same session. If you need to export an OpenDocument Text file that was just imported, simply drag and drop the file from your WebDAV client to a folder on your computer as explained in Chapter 13, *WebDAV*. This is only useful if you have already deleted the file that you were importing in the same session from your computer; otherwise, you should already have the file on your computer.

- You are exporting an `.odt` file that is used as the "File" attribute for a content object.

In other words, you cannot convert an article or similar object to an OpenDocument Text file via WebDAV. Recall the WebDAV display rules from Section 13.1.2.1, “Class-dependent display rules”. In WebDAV, most objects are either represented by zero-byte files (see Section 13.1.2.1.1, “Zero-byte files”), folders, images, or, for `File` objects, the files themselves.

14.4. Microsoft Word documents

Importing from and exporting to Microsoft Word documents in eZ Publish follows similar procedures to performing the same operations with OpenOffice.org documents. The difference is that you are now working on `.doc` files. Also, while import can be done both through the Website Interface and the Administration Interface, export is only supported in the Administration Interface. To do so, follow the steps described in Section 14.3.2, “Exporting via the Administration Interface”, but select "OpenOffice.org - Export Word" in the pop-up menu, as illustrated below:

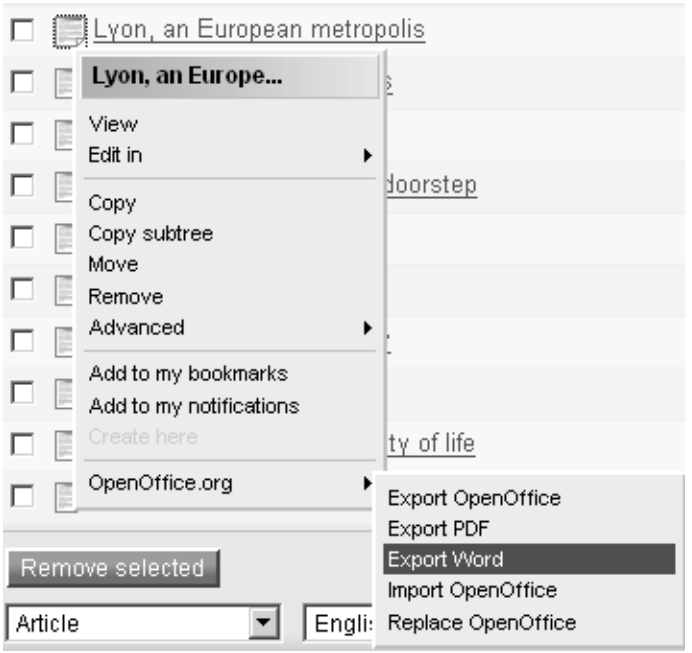


Figure 14.6. Use the Administration Interface pop-up menu to export a file

Recall from Section 14.1.2, "eZODF requirements" that there are extra requirements for Microsoft Word functionality to work on your site.

14.5. Summary

In this chapter, we introduced OpenDocument Text files and the eZ OpenDocument Format (eZODF) extension.

OpenDocument Text files use the OpenDocument Format for word processor documents and have the `.odt` extension. They can be opened in the OpenOffice.org Writer and NeoOffice Writer applications.

When the eZODF extension is installed and enabled in eZ Publish, you can import `.odt` files to create objects or new versions of existing objects. By mapping sections in `.odt` files to attributes in content classes, different documents, when imported, can create objects of different classes. You can also export any object on the system to an `.odt` file. eZ Publish can also be configured to import from and export to Microsoft Word documents.

Chapter 15. Webshop

eZ Publish comes an integrated shop mechanism called the *Webshop*, which plugs directly into the content model. A Webshop implementation can be as simple as offering a book or company t-shirt, or it can be a full-blown online retail business. Accordingly, the complexity in managing this mechanism varies greatly, depending on how much of the available functionality you use. This chapter explains the components, concepts and usage of this sub-system, as well as the **Webshop** tab and shop-related datatypes.

This chapter is relevant to webshop managers and deals mostly with the Administration Interface. It can also be read as a conceptual introduction to what the eZ Publish Webshop offers.

Recall that:

- A *datatype* describes the type of value that can be stored in a content object attribute and is the smallest possible entity of storage. It determines how a specific type of information should be validated, stored and retrieved.

- A *valid user* is an object of the `User` content class (or any other object with an attribute of the "User account" datatype; see Section 5.3.1.1, “User account datatype”). The *anonymous user* is a special-purpose virtual user account. Each time someone visits your site, they are silently logged in by the system as "Anonymous". Then, the permission system can correctly apply the rules specified for this user (or the user group). If the visitor decides to log in with a personal account, the current user will be changed accordingly.

- The *section ID* of an object denotes the section to which the object belongs. Each object belongs to exactly one section. See Section 4.2.2, “Section properties”.

- An eZ Publish *INI file* is a configuration file that controls the behavior of a specific part of the system. See Section 3.5, “The configuration model”. You can check a specific settings value through the **Settings view** interface, as described in Section 3.5.4.1, “Viewing the settings of a configuration file”.

- A *module* offers an interface for web-based interaction, providing access to eZ Publish core functionality. Each module has a set of *views* that enable you to reach the functions provided by the module. See Section 1.3, “Modules and views”.

In this chapter, you will find information about the following:

- Webshop-related requirements

- The components of the e-commerce sub-system, how they relate, and how to manage them

- Webshop-related datatypes, and how attributes of these are used
- The layout and interfaces of the **Webshop** tab
- How to use multiple currencies on a site
- How the Value Added Taxes (VAT) charging system works
- How to apply discount rules for certain users

This chapter assumes that you are familiar with content creation and editing in general. You may use Chapter 2 and the Appendix for a quick overview, or refer back to *eZ Publish Content Management Basics* for detailed instructions. This chapter is not targeted at developers or site administrators, and technical issues such as creating custom classes, creating VAT handlers, billing and payment systems, shipping handling, and exchange rate update handlers have been deliberately omitted. Interested readers should refer to http://ez.no/doc/ez_publish/technical_manual.

In Depth: Payment, shipping and handlers

A *handler* is a means of controlling something. For example, a VAT handler controls the charging of Value Added Taxes, while a shipping handler controls shipping procedures. In eZ Publish, such handlers are usually represented as PHP classes implementing the functionality. For a domain of control, for example VAT, you can apply different logistics by substituting which handler is used, as specified in the configuration files.

A *shipping handler* enables you to add shipping cost and options to your webshop products. For example, costs can vary depending on where the order is to be mailed, or whether regular airmail or express delivery is chosen. Shipping costs are added to the order total, viewable by customers in their basket and during checkout. eZ Publish does not include any built-in shipping handlers, but supports custom-made handlers. For more information, refer to the *Features - Improved shipping handling* section of http://ez.no/doc/ez_publish/technical_manual.

Some countries have regulations preventing you from storing credit card information. *Payment gateways* enable you to satisfy these regulations by letting some third-party entity take care of processing payments. For example, eZ Publish has built-in support for integration with Paynet through a payment gateway extension. Other extensions exist and can be created to integrate with other payment gateways. For more information, refer to http://ez.no/doc/extensions.

You will find all or just some parts of this chapter relevant, depending on the level of sophistication of your shop. The sections should be read in the order listed. If your site only sells products in one currency, you can skip Section 15.6, “Prices and multiple

currencies". If all your prices are entered including taxes, you can skip Section 15.7, "VAT charging system". If you do not offer discounts, you can skip Section 15.8, "Discount rules".

15.1. Webshop components

The eZ Publish Webshop components include: products; Value Added Taxes and discounts affecting prices; wish lists and shopping baskets for customers; and orders. The relationship between these six components is shown in the illustration below.

Figure 15.1. Webshop components

15.1.1. Products

A *product* is a content object with a price. The price is represented by an attribute of the "Price" or "Multi-price" datatype (see Section 15.5.1, "Price and Multi-price datatypes").

It is this attribute that makes an object a defined product, in the same way that an attribute of the "User account" datatype defines an object as a user (see Section 5.3.1.1, “User account datatype”). Depending on which of these datatypes is used, the product is referred to as a *simple-price product* or *multi-price product*. Section 15.6, “Prices and multiple currencies” explains the currency-related concepts in detail and how to work with a multi-currency webshop.

In addition, there are several other Webshop-specific datatypes, which make it possible for visitors to customize product orders (for example, to select different shoe sizes or t-shirt colors) and to charge country-dependent taxes (see Section 15.7.3, “Country-dependent VAT”). These datatypes are all described in Section 15.5, “Shop-related datatypes”.

Adding products to your shop works in the same way as you add other content objects, using the familiar editing interfaces. The only difference is the input fields for the Webshop-specific datatypes, which are explained in this chapter. In other words, you can create and edit products both through the Website Interface and the Administration Interface.

In the Administration Interface, products are identified with the product icon:

15.1.1.1. Product class

Sites that use the Website Interface come with a content class that defines a structure for storing information about products. The `Product` class has the following attributes, with their datatypes in parentheses.

- *Name* ("Text line"); for example "CD/DVD box 1"
- *Product number* ("Text line"); for example "1234"
- *Short description* ("XML block")
- *Description* ("XML block")
- *Image* (and caption) ("Image")
- *Price* ("Price"); see Section 15.5.1, “Price and Multi-price datatypes”
- *Additional options* ("Multi-option"); see Section 15.5.2, “Range option, Option and Multi-option2 datatypes”
- *Tags* ("Keywords"), see Section 9.1.1, “Keywords datatype”

The screenshot below shows the "Description", "Price" and "Image" attributes of the `Product` class in edit mode.

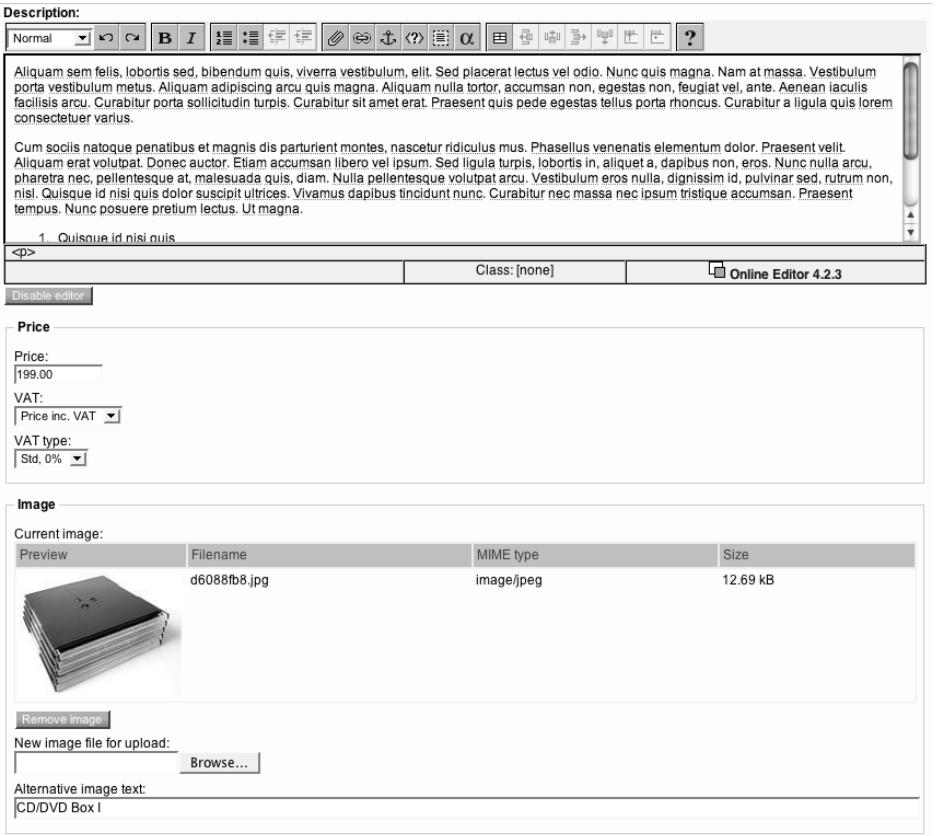


Figure 15.2. Product class in edit mode

15.1.2. Value Added Taxes (VAT)

Value Added Taxes (VAT) affect the price of a product by specified percentages. For example, if the product price is $100 and the VAT is 12.5%, the final price would be $112.50. A price can be entered including or excluding VAT when editing the Product object. If it is entered excluding VAT, VAT is calculated and added to the set price (as in the previous example), whereas if entered including VAT, the price stored in the object is also the final price.

VAT commonly vary from country to country and also depending on the product types. Because of this, eZ Publish supports setting up different VAT types to be used by the VAT charging system, as described in Section 15.7, “VAT charging system”.

15.1.3. Discount rules

A *discount rule* reduces the final price of a product (after VAT and currency exchanges) by a given percentage. For example, partners could be given 20% off all products. Managing discounts is described in Section 15.8, “Discount rules”.

15.1.4. Wish list

The *wish list* is user specific (not accessible to anyone else) and contains products of interest added to the list (by clicking the **Add to wish list** button) by that user. In a way, it is a list of product bookmarks that the user can refer to later. To purchase an item in a wish list, click on the link to the product to view it, then add it to the basket. (You cannot directly transfer items from the wish list to the shopping basket.)

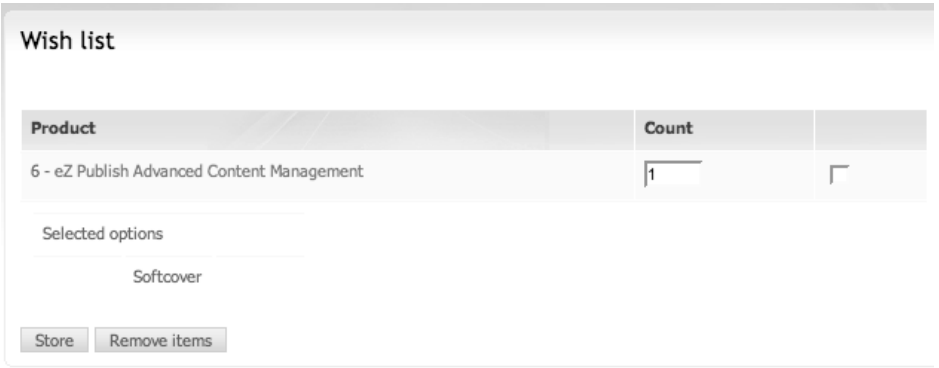


Figure 15.3. Wish list

Wish lists are stored between browsing sessions, preserving the contents when users log out. Users who are logged in can access their wish list from their personal space. Section 15.3.1, “Browsing and buying” gives a walk-through of the wish list and the basket (described next) from a customer perspective, illustrating the entire shopping scenario.

15.1.5. Basket

The *shopping basket* ("basket" for short) is also user specific, and can be updated at any time when logged in. In contrast to the wish list, it is not preserved across browsing sessions, meaning that when you log out, the basket is cleared. The items in the basket can be purchased by initiating the checkout process, as described in Section 15.3.1, “Browsing and buying”. Note that if you do not have any items in the basket, it is hidden (in other words, there is no link to it).

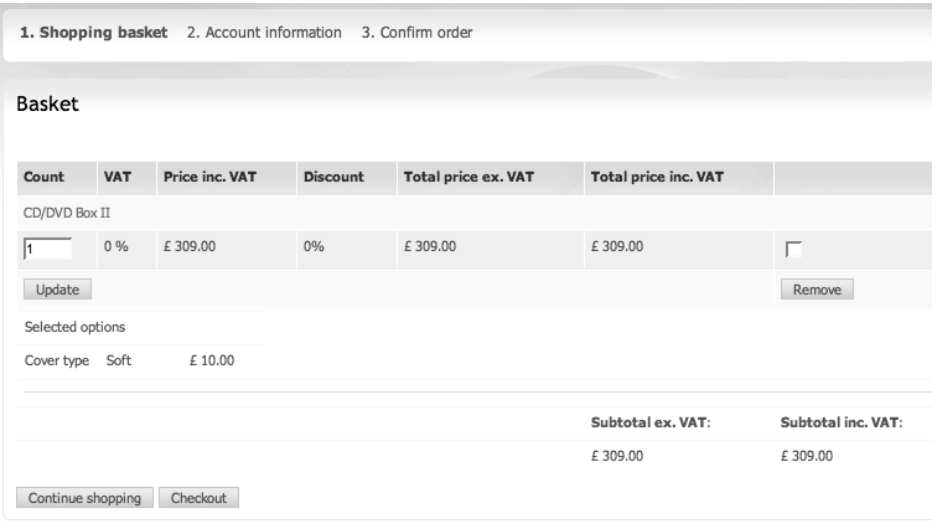


Figure 15.4. Shopping basket

15.1.6. Orders

An *order* is created when the checkout process from the shopping basket is completed. Orders are stored in the database as special-purpose objects (not content objects). When an order is created, both the customer and site administrator (specified in the "AdminEmail" setting in the "[MailSettings]" block of `site.ini`) are notified by email with details of the order. Section 15.10, “Managing orders” describes how to manage orders from the Administration Interface, while the checkout process is described in Section 15.3.1, “Browsing and buying”.

15.2. Shop-related requirements

In addition to the general requirements for eZ Publish specified in Section 1.5, “eZ Publish access requirements”, there are a few more requirements in order to use the Webshop. As previously mentioned, you need at least one content class with an attribute of the "Price" or "Multi-price" datatype. Also, webshop managers need to be assigned a role including a policy granting access to all functions of the "shop" module. Customers need access to only the "buy" function of the "shop" module. Content editors do not usually need shop-specific permissions to add or edit products. Refer back to Chapter 5 for more information on user management.

If you have a webshop with simple-price products and want to use multi-price products, you should upgrade the products. This is done by first creating the desired currencies (described later), and then running a special-purpose upgrade script for the Webshop. The script simply changes the datatype for the "Price" attribute and updates the values

accordingly. Ask your site administrator or refer to http://ez.no/doc/ez_publish/technical_manual for detailed instructions.

If you are going to use the following Webshop enhancements, additional configuration is needed. Refer to http://ez.no/doc/ez_publish/technical_manual for full details.

- Billing and payment systems (for example, integration with Paynet; see http://ez.no/doc/extensions)
- Shipping handling (not a built-in feature but can be added via a custom handler)
- Country-dependent VAT (see Section 15.7, “VAT charging system”; you must enable the built-in VAT handler for country-dependent VAT, or have a developer create a custom handler)

15.3. Webshop from a customer perspective

Before delving into Webshop management, let us first look at the shop as it is experienced by customers. The following example illustrates a typical shopping scenario. We then describe how users register for a shop account.

15.3.1. Browsing and buying

All prices are shown including VAT, although VAT is specified in the basket per product. If your site uses a VAT handler, VAT is recalculated at checkout.

The following screenshot shows the "Products" frontpage, with some featured products.

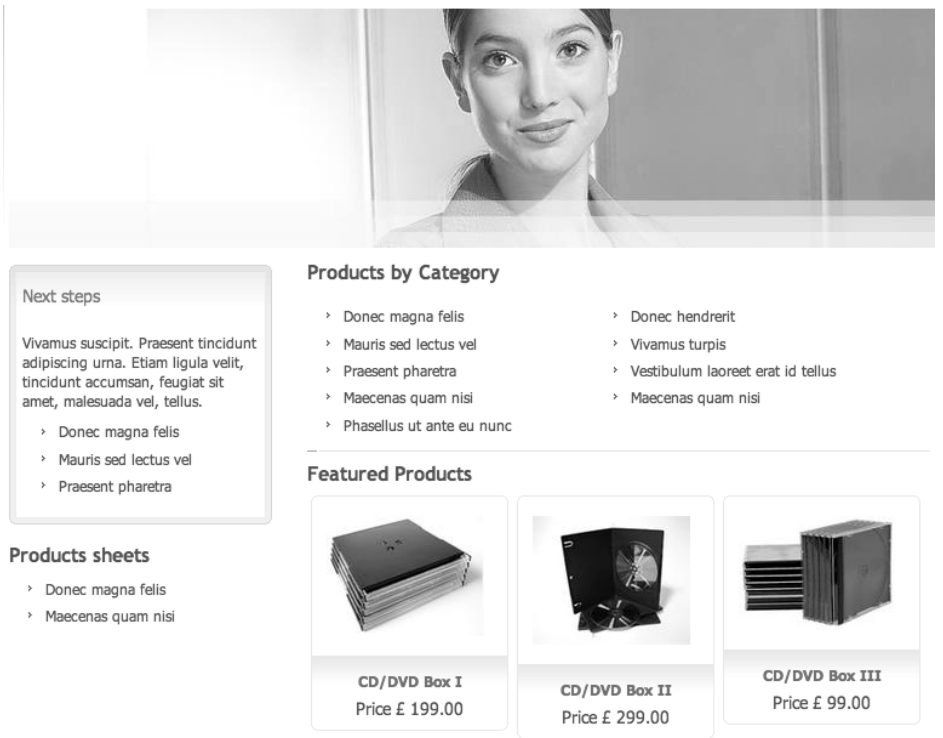


Figure 15.5. "Products" frontpage

The next screenshot shows one of the product folders, containing one product named "OS Type 1".

Figure 15.6. Product folder

Clicking the product name will take you to a page with full product information (shown below). You can also navigate to the other product folders by using the left menu.

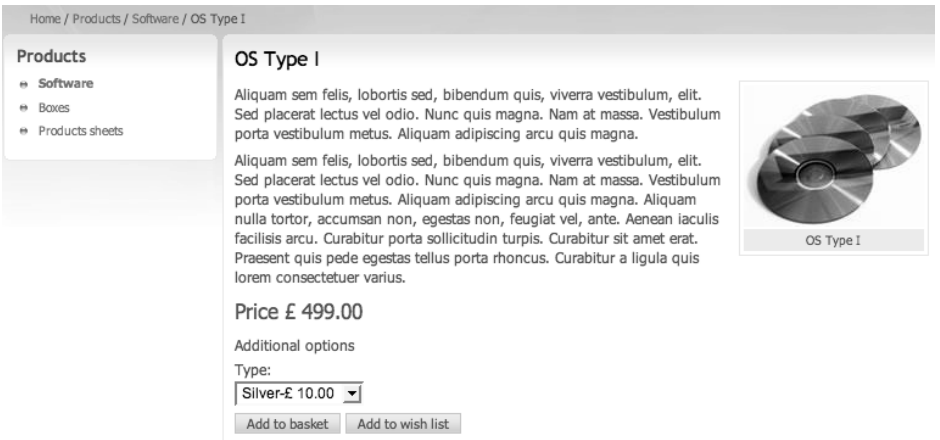


Figure 15.7. Product information page

As seen in the above screenshots, it is only when viewing a specific product that the **Add to basket** and **Add to wish list** buttons become available. Let us browse and put a few interesting items in the wish list. The following screenshot shows a wish list containing 3 items.

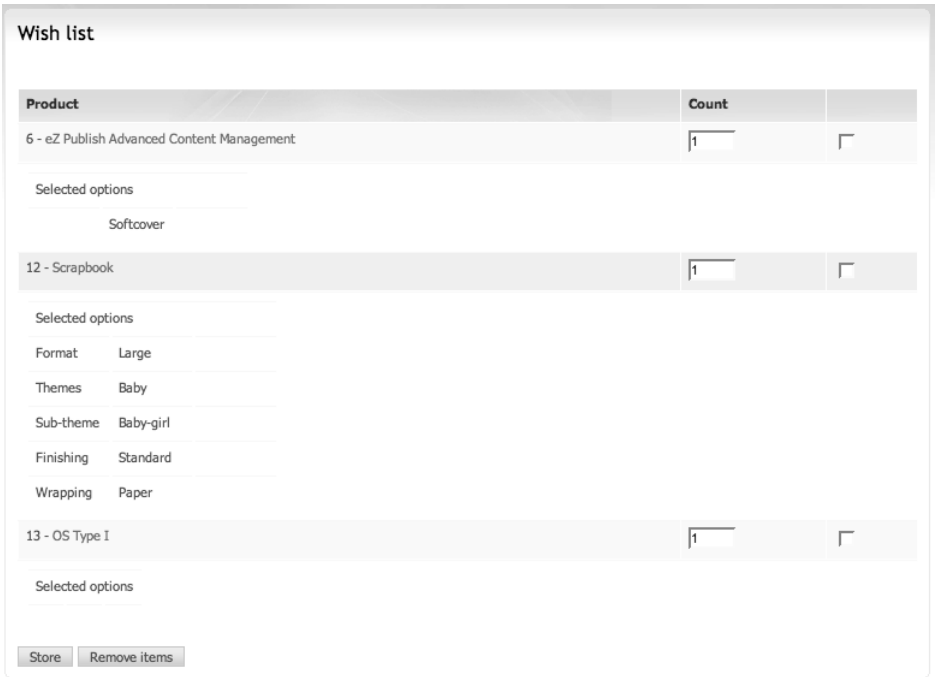


Figure 15.8. Wish list with 3 items

Next, let us add the book and scrapbook to the basket. Either browse the product pages or access an item from the wish list. Recall that you can open the wish list from your personal space. When you add the first item to the basket, the **Shopping basket** link will appear in the top right corner.

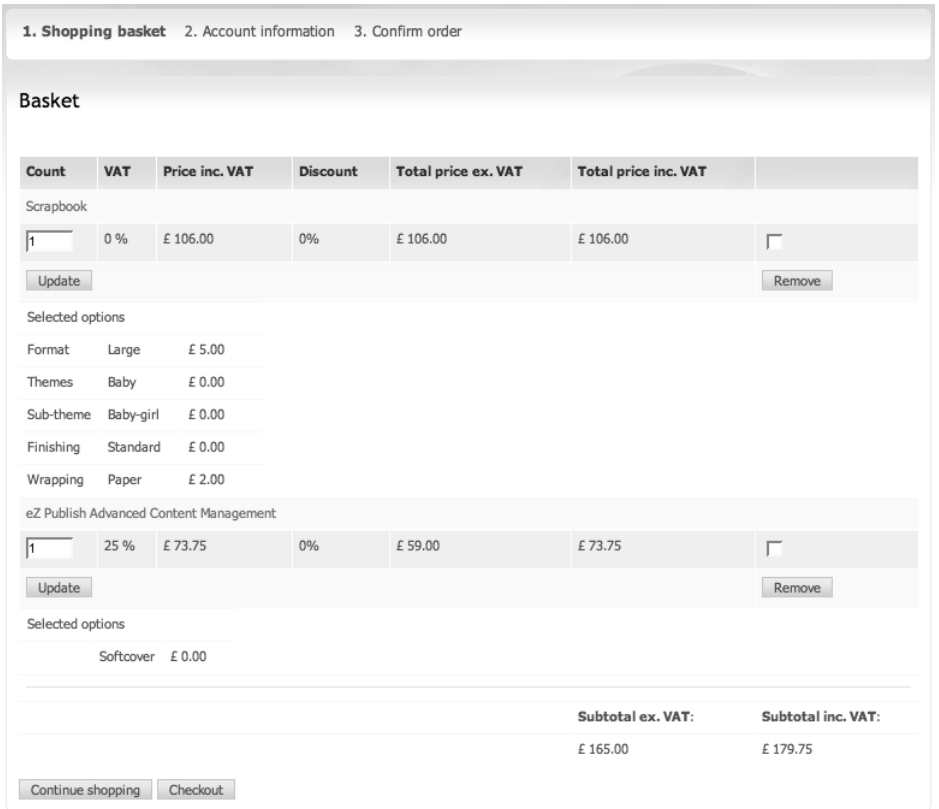


Figure 15.9. Basket with 2 items

The above screenshot shows the shopping basket with 2 items: "Scrapbook" and "eZ Publish Advanced Content Management".

When viewing the shopping basket, you have the following options: change the amount by editing the **Count** input field and clicking the **Update** button; mark the corresponding checkboxes, then click the **Remove** button to take something out of the basket; resume browsing the shop by clicking the **Continue shopping** button; or proceed to the checkout by clicking the **Checkout** button.

Up to this point, the scenario is the same regardless of the site configuration. Depending on the shop account handler (described in Section 15.3.2, “Shop account”) used, the **Order confirmation** interface is either loaded immediately, or one of two forms must first be filled in. These forms are described next.

15.3.1.1. Checkout form

The *checkout form* is part of the default checkout process, and it does not require the customer to log in. It allows anonymous users to be customers, provided that the Anonymous role includes a policy for the "buy" function of the "shop" module. The checkout form is shown below:

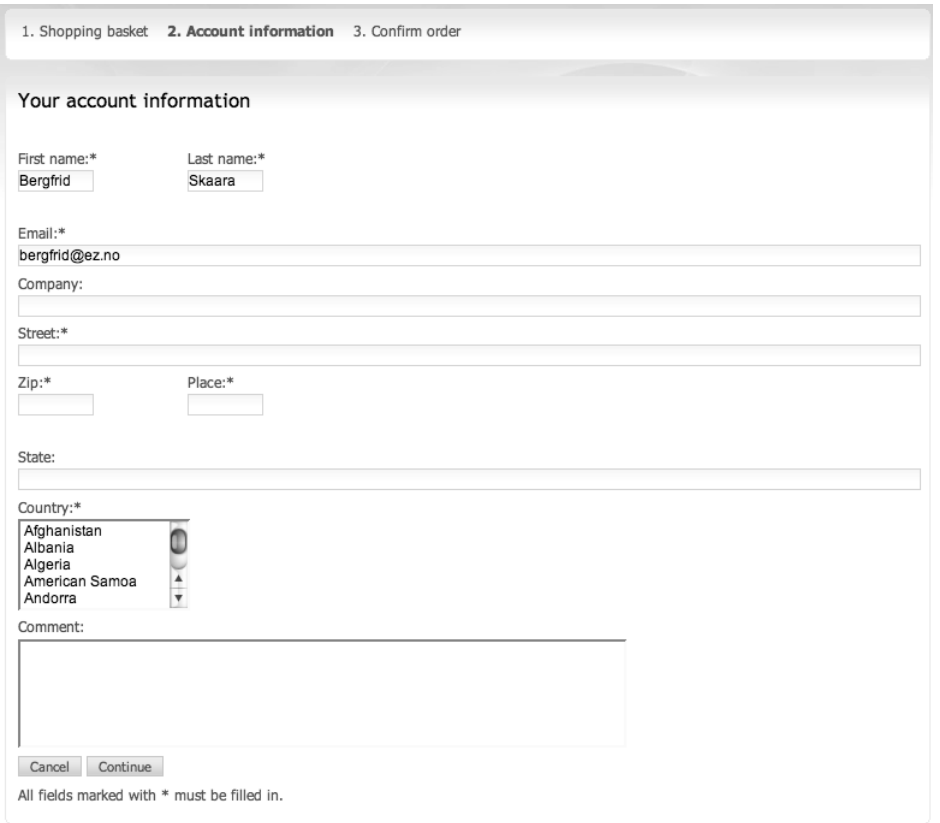


Figure 15.10. Checkout form

If the customer has a user account and is currently logged in, the system will pull information from the content object, pre-filling some of the input fields of the checkout form (as shown above). The contents of the pre-filled fields can be changed, so having a user account is purely a convenience and not a requirement.

After the customer fills out at least the required fields, clicking the **Continue** button will load the **Order confirmation** interface.

15.3.1.2. Shop account registration form

The *shop account registration form* is used to create new shop accounts, which are independent of user accounts. When shop accounts are used, it does not matter whether the customer is a registered user on the site, or whether or not he or she is logged in. As with the above scenario with the checkout form, anonymous users are allowed as customers. Also, the decoupling of shopping account and user account allows both personal orders and orders on behalf of others.

The screenshot below shows the shop account registration form. Note that it collects less information than the checkout form. In particular, it does not have a specific country field, making it unsuitable for country-dependent VAT (which is explained in Section 15.7.3, "Country-dependent VAT").

Figure 15.11. Shop account registration form

In this case, the customer is responsible for entering a well-formed mailing address.

15.3.1.3. Order confirmation interface

The **Order confirmation** interface is shown for all customers after they click the **Checkout** button, although the checkout form or shop account registration form might first need to be filled in (depending on the site configuration).

The following screenshot shows the **Order confirmation** interface.

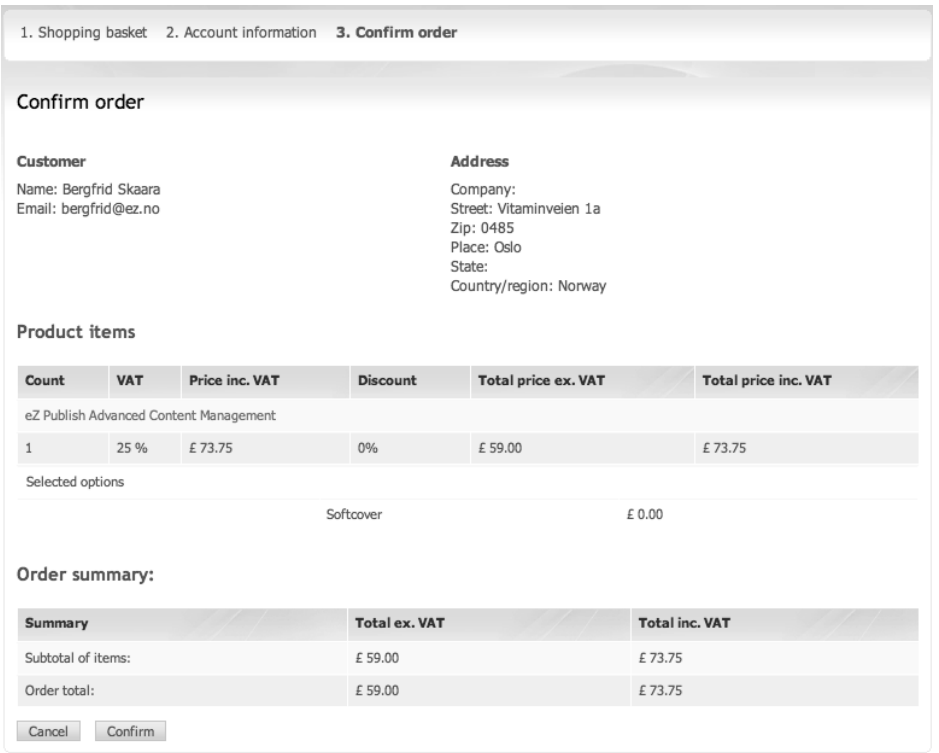


Figure 15.12. Order confirmation interface

After the **Confirm** button is clicked, the following occurs:

- The details of the now submitted order are shown.
- The customer receives a confirmation email to the email address specified.
- The order status is set to "Pending". Further action depends on your site's billing and shipping routines.
- The site administrator receives an email notification. This is an exact copy of the confirmation sent to the customer. The administrator can then access the order in the **Webshop** tab, or view the customer information that is added to the *customer list*. This is described in Section 15.10, “Managing orders”.

15.3.2. Shop account

The Webshop stores customer information in different ways, depending on a configuration setting as described in the in-depth block below. The following alternatives are available:

- Users do not need to be logged in when making orders. If they are logged in, some fields in the checkout form are pre-filled according to their User object. This requires that Anonymous users are granted access to the "buy" function of the "shop" module.
- Users must be logged in when making orders. Otherwise, they will be prompted to log in before the checkout process completes.
- Users need to fill in the shop account registration form when making orders. This requires that Anonymous users are granted access to the "buy" function of the "shop" module. If logged in, the current user's information (stored in their User content object) is not applied in this scenario, meaning that he or she can create both personal orders and orders on behalf of others.

In Depth: Shop account handler

A *shop account handler* handles and stores customer information for Webshop orders. There are three built-in shop account handlers (corresponding to the list entries above) called "ezuser", "ezdefault" and "ezsimple". The default value of this setting is "ezuser", which means that users do not need to be logged in when making their orders.

To check which handler is used on your site, access the **Settings view** interface in the **Setup** tab, then select *`shopaccount.ini`* (and a siteaccess). Look for the "Handler" setting in the "[AccountSettings]" block. The handler in the "[AccountSettings]" block should not be confused with the handler in the "[ConfirmOrderSettings]" block, which is used for additional and custom-made checkout functionality such as sending confirmation emails.

15.4. Webshop tab

Webshop management includes tasks such as managing orders, working with VAT and currencies, and specifying discounts, in addition to managing the actual products. The **Webshop** tab offers special-purpose interfaces to view and manage your shop. The following screenshot shows this tab. Table 15.1, “Left menu items in the Webshop tab” below outlines the options found in the left menu.

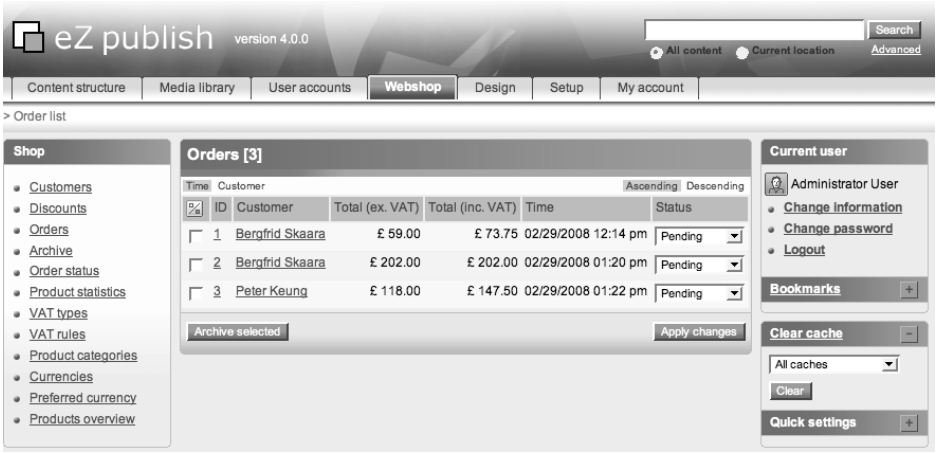


Figure 15.13. The Webshop tab

As previously explained, products are content objects with a price. Being content objects means that they are stored in the content node tree, usually within the Content branch, and are thus accessible from the **Content structure** tab (or through the Website Interface). As a consequence, setting prices and options, updating product information and so on works like ordinary content editing, outside the **Webshop** tab.

Technical Webshop administration, such as working with the configuration files or installing add-ons (for example, for billing and shipping), is done through the **Settings view** interface of the **Setup** tab, or directly on the file system. These tasks are usually handled by the site administrator and not webshop managers.

In Depth: The "shop" module

The e-commerce capabilities are actually a sub-system of eZ Publish that interfaces with - but is not part of - the content model. The "shop" module provides interfaces to both the content and the e-commerce engines, as well as views for the Webshop (for example, for the basket, wish list and order list). Because of this, using and managing the Webshop involves many special-purpose interfaces.

The following table describes the available left menu items of the **Webshop** tab.

Table 15.1. Left menu items in the Webshop tab

Menu item	Description
Customers	Brings up the **Customer list** interface where you can view the name, number of orders and the total value of purchases for each customer. Clicking the customer name opens the **Customer information** interface.
Discounts	Brings up the **Discount group** interface for managing discounts. See Section 15.8, “Discount rules”.
Orders	Brings up the **Order list** interface used for order management. See Section 15.10, “Managing orders”.
Archive	Brings up an interface similar to the **Order list** interface, but containing orders that have been archived. See Section 15.10, “Managing orders”.
Order status	Brings up the **Order status** interface for managing order statuses. See Section 15.10.1, “Order statuses”.
Product statistics	Brings up the **Product statistics** interface. See Figure 15.65, “Product statistics interface” and Section 15.10.3, “Sales statistics”.
VAT types	Brings up the **VAT type management** interface. See Section 15.7.1, “VAT types”.
VAT rules	Brings up the **VAT rule management** interface. See Section 15.7.3.2, “VAT charging rules”.
Product categories	Brings up the **Product categories** interface. See Section 15.7.3.1, “Managing product categories”.
Currencies	Opens the a list of available currencies. See Section 15.6, “Prices and multiple currencies”.
Preferred currency	Displays the preferred currency. See Section 15.6, “Prices and multiple currencies”.
Products overview	Opens the **Products overview** interface. See Section 15.4.1, “Products overview interface”.

15.4.1. Products overview interface

The following screenshot shows the **Products overview** interface. It lists all products published on your site, regardless of their location within the content node tree. In other words, you can access all of your products on a single page. This is particularly useful if products are scattered throughout the content structure, or when you want to review prices or compare products.

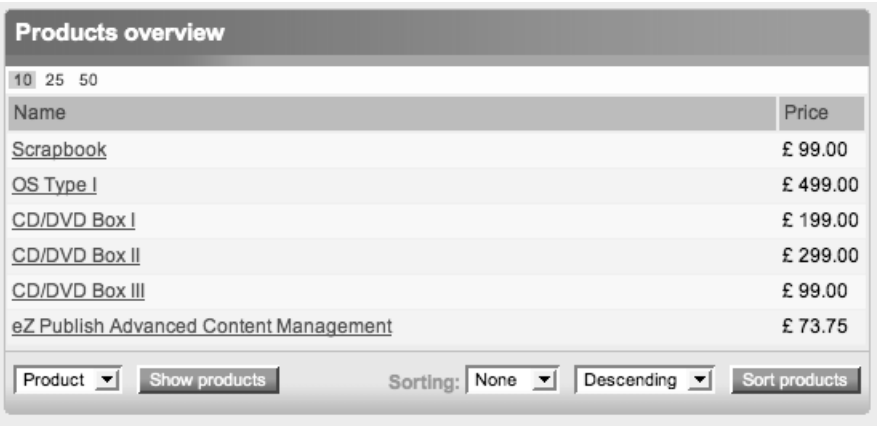


Figure 15.14. Products overview interface

You can filter the list by class and sort it by name or price. Clicking a product name enables you to view the product in its main location, commonly in the **Content structure** tab.

15.5. Shop-related datatypes

It is the Webshop-specific datatypes that differentiate Product objects from other content objects. Recall that at a minimum, you need one attribute of either the "Price" or "Multi-price" datatype in your product class(es). The other datatypes are used for additional functionality.

15.5.1. Price and Multi-price datatypes

A *price attribute* is an attribute of either the "Price" or "Multi-price" datatype. The attribute label does not differentiate between which datatype is actually used, but you can easily identify the difference when editing it, as illustrated in the following screenshots:

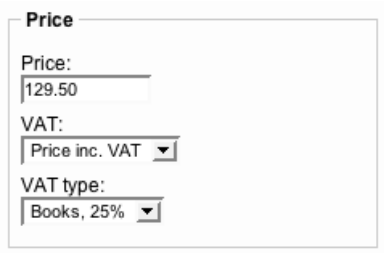


Figure 15.15. Attribute of the Price datatype

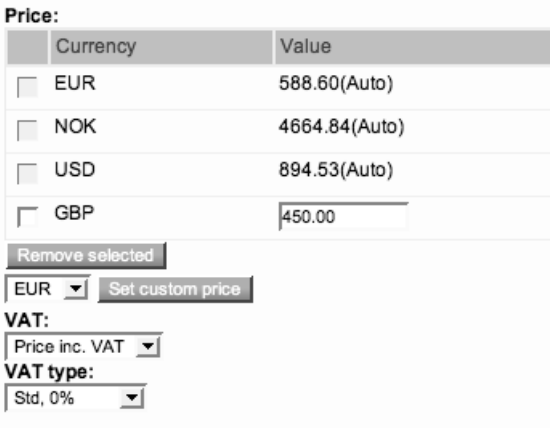


Figure 15.16. Attribute of the Multi-price datatype

Price attributes store a price including or excluding VAT. All prices are stored with a precision of two decimals. The difference between attributes of these datatypes is that "Price" attributes store a single price per product, whereas "Multi-price" attributes store multiple prices in different currencies per product. Simple-price products are easily edited, since you only have to worry about a single value. Multi-price products require some more management, and this is explained in detail in Section 15.6, “Prices and multiple currencies”.

Warning

A content class can only use one of the "Price" and "Multi-price" datatypes at a time, since they both store a product's price. Also, you cannot have an object with a "Price" attribute and an object with a "Multi-price" attribute in the shopping basket simultaneously. It is up to your developers to ensure consistency, and preferably the same datatype should be used for the "Price" attribute in all product classes.

15.5.2. Range option, Option and Multi-option2 datatypes

Attributes of the "Option", "Range option", and "Multi-option2" datatypes are used to enable customers to choose from customization options for the product being purchased. For example, they can select a shoe size, t-shirt color, brand, or whether or not to add gift wrapping.

Consider the following scenario. Your webshop is selling shoes, and you have just gotten a batch of sneakers in stock. They come in 10 different sizes, each in the colors black, red, brown and white. This gives 40 unique pairs of sneakers. Having an attribute for additional options in a product gives you the advantage of storing a single Product object for the sneaker, rather than having 40 objects that are very similar. For a customer, the shop is simpler, since only one product is shown, not 40.

A product can use none, one or more of the three different option datatypes.

15.5.2.1. Range option datatype

A *range option* specifies a single group of numerical options without price differentiation (the additional price will be 0). This is typically used for sizes. You can specify start, stop and step values. For example, a start size of "4", a stop size of "12" and a step value of "2" would result in the options "4", "6", "8", "10" and "12". The following screenshot shows an attribute of this datatype in edit mode.

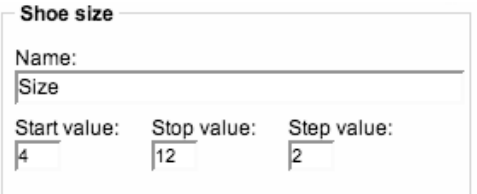


Figure 15.17. Range option attribute

15.5.2.2. Option datatype

An *option* also specifies a single group of options, but with a short text and additional price. For example, this can be used for colors or sizes that are represented by text. The following screenshot shows what editing an attribute of this datatype looks like.

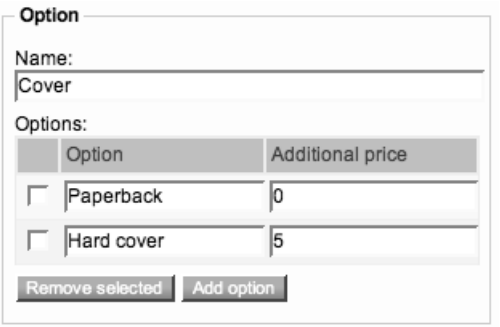


Figure 15.18. Option attribute

15.5.2.3. Multi-option2 datatype

The most sophisticated alternative in the group are *multi-option* attributes, which use the "Multi-option2" datatype. Basically, multi-options work similarly to normal options (with a name and additional price), but are more complex. Multi-option attributes provide the following additional features:

- They are fully configured at the object level. As a consequence, your site can have a single generic product class and yet have the flexibility for a wide range of diverse products with different traits and custom configuration. If your shop previously sold only books, you can add t-shirts or movies with tailored options, while using the same product class. This enables content managers to expand the shop without having to create new content classes and consult designers about custom templates.

- You can create multiple levels of options where the available options at the nested level(s) are dependent on the selected option from the previous level. These are managed by *dependency rules*. For example, you might be selling a t-shirt that comes in small, medium, and large sizes. This t-shirt might come in two colors, dependent on the size: red for all sizes but gray only for the medium and large sizes. Therefore, the size option would be at one level, and the color option would be at a level below it, with rules determining which colors are available for which sizes.

- Options are organized in groups. This lets you define distinctive groups of traits for the same product.

Here, we will outline the basic structure of a multi-option attribute, while Section 15.9, “Example: Working with multi-options” will go through a detailed example demonstrating how to cater a multi-option attribute in edit mode to suit your product's traits.

An attribute of the "Multi-option2" datatype has a minimum of 1 multi-option group, containing at least 1 multi-option which again contains 1 or more options:

Figure 15.19. Multi-option attribute

Each multi-option represents a trait (or sub-trait), while each group represents a collection of traits. For example, you could use groups to differentiate between interior and exterior traits of a car. Adding multi-options within a group corresponds to adding more traits to the product. Adding a multi-option beneath an existing multi-option enables you to have sub-traits, which can be dependent on their parent traits. The following screenshot shows a multi-option attribute with 2 groups, where the top group has 3 levels of multi-options:

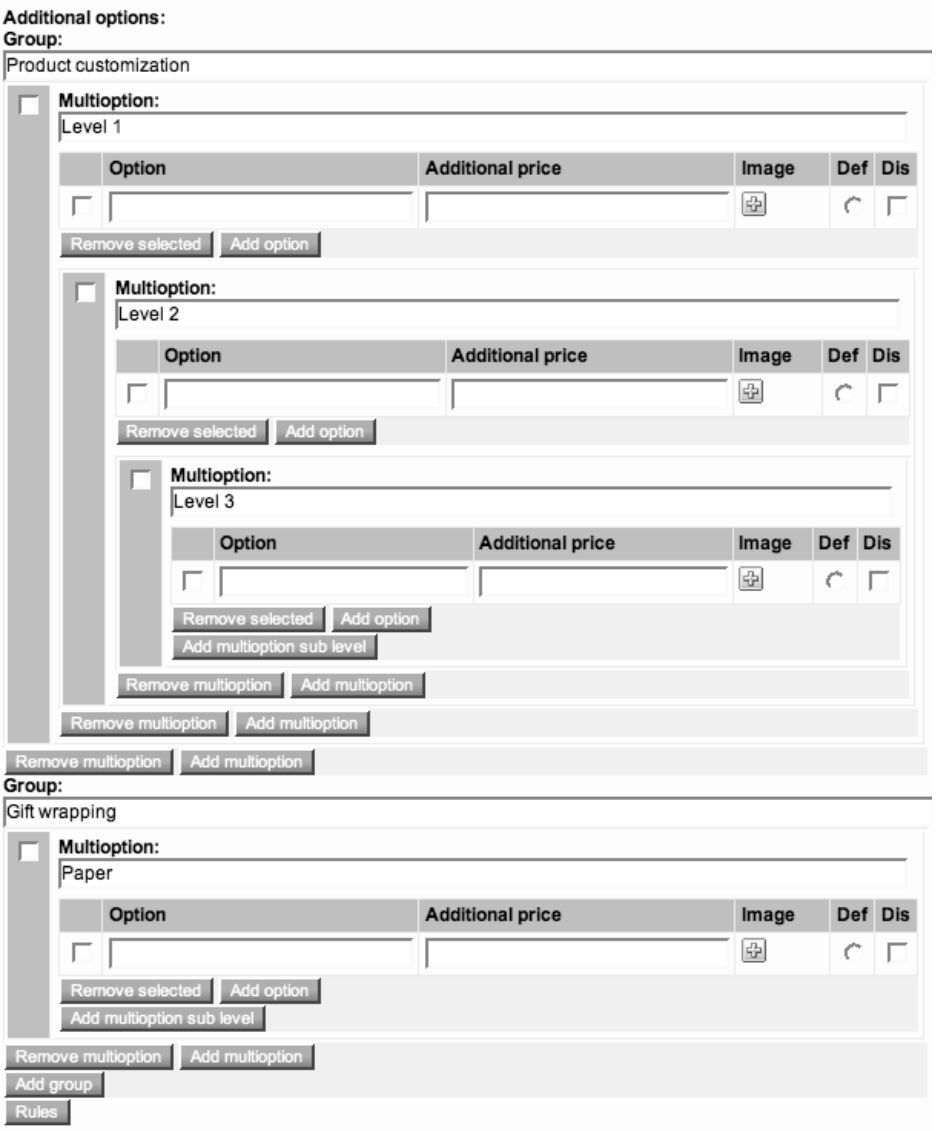


Figure 15.20. Multi-option attribute with multiple levels

Each option has a name and an additional price (which can be 0), the possibility of associating an image, marking it as the *default selection* (within one multi-option) and marking it as *disabled* (not selectable). When you click the **Rules** button at the bottom, you can manage the dependency rules for all of the options in the attribute. This is illustrated in detail in Section 15.9, “Example: Working with multi-options”.

Tip

You have probably noticed that we sometimes write "Multi-option" and sometimes "Multi-option2". This is so because there is actually two datatypes, one called "Multi-option", and one called "Multi-option2". The former was deprecated in eZ Publish version 3.10, although it is still used in the default `Product` class in sites that use the Website Interface 1.3. It has been replaced by the enhanced "Multi-option2" datatype, which we describe in this chapter. In this book, the term "multi-option attribute" refers to attributes of this enhanced datatype.

15.5.3. Product category datatype

Attributes of this datatype simply store a product category to be used by the VAT charging system. By default, the `Product` class does not include an attribute of the "Product category" datatype, so it must be explicitly added to the class definition if you need to use it.

The following screenshot shows the dropdown list displayed when editing an attribute of the "Product category" datatype. The list will contain the categories you have added to your system. See Section 15.7.3.1, “Managing product categories” for more information.

Figure 15.21. Product category attribute

15.5.4. Country datatype

The "Country" datatype is the only one of the Webshop-specific datatypes that is not usually used for attributes in the `Product` class. Instead, it is normally used for an attribute in the `User` class in order to store a country. This is used in conjunction with VAT rules for country-dependent VAT, which is described further in Section 15.7, “VAT charging system”. The available countries are defined in the *`country.ini`* configuration file, with a standards-compliant name (such as "Norway"), code (such as "NO") and International Dialing Code (such as "47"). The following screenshot shows the dropdown list displayed when editing an attribute of the "Country" datatype.

Figure 15.22. Country attribute

15.6. Prices and multiple currencies

Currency-related management only applies to webshops with products using price attributes of the "Multi-price" datatype. A multi-price product will always have as many prices as there are defined currencies in a webshop. We start by explaining some concepts that are specific to multi-currency shops, then show you how to manage currencies and prices.

15.6.1. Currency-related concepts

This section establishes a common language in order for you to understand the subsequent section on multi-currency management.

15.6.1.1. Currency

Let us start by looking at the **Currency management** interface (opened by clicking the **Currencies** link in the left menu of the **Webshop** tab) to see what a currency includes.

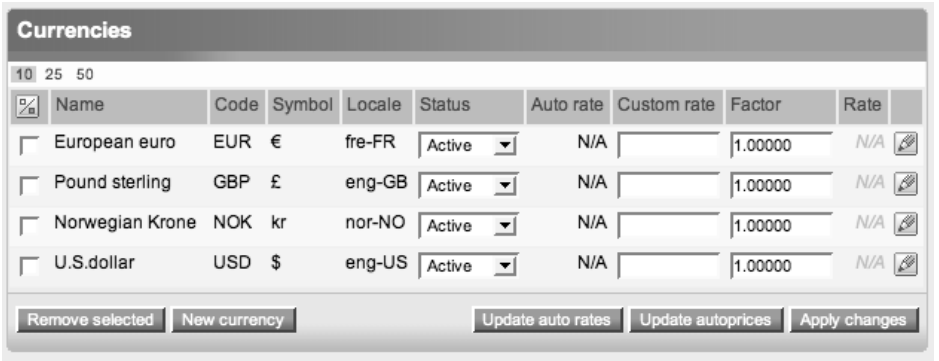


Figure 15.23. Currency management interface

A currency is represented by a three-character *currency code*, such as "USD", "EUR" and "NOK". This code can be thought of as an unique identifier for the currency. eZ Publish has built-in support to map well-known currency codes to *currency names*, such as "European euro". For an overview, see http://en.wikipedia.org/wiki/ISO_4217#Active_codes, or other web resources that list the *ISO 4217 standard*. If the code is unknown, it is listed as "Unknown currency name".

The *currency symbol*, such as "$" or "€", denotes that a number is a monetary value.

A *formatting locale* is a locale that is used for price formatting, including variations such as which character to use as the separator between dollars and cents (for example, "1,00" or "1.00" to represent one unit of currency). The concept of a locale was described in Section 12.2.1, “Locale”. Prices will automatically be formatted according to the currency settings of the chosen locale.

The *base currency* is the currency the site administrator has chosen on which to base the currency rates (controlled by the "BaseCurrency" setting in `shop.ini`). In other words, the rate for this currency is always 1. Rates for other currencies are specified relative to the base currency.

The *default currency* specifies which currency the base price should be in when new products of a specific class are created. When a price is entered, auto prices are calculated in the other currencies based on this. This setting is set in the **Class edit** interface, for the "Multi-price" attribute. The base and default currencies are often the same, but you can set them to be different.

The *preferred currency* specifies the currency that is shown to the user. It is set by individual users; if a user does not specify a preferred currency, the currency displayed is as specified within a `shop.ini` setting. The preferred currency for site administrators is set by clicking the **Preferred currency** link in the left menu of the **Webshop** tab, selecting from the dropdown list, then clicking the **Set** button. This does not affect price and currency management - only which currency is displayed. Refer to the technical manual at http://ez.no/doc/ez_publish/technical_manual for more information.

A *currency status* indicates whether a given currency is active or inactive. Only active currencies are available to site visitors. An inactive currency is hidden in the sense that it cannot be used as the preferred currency, and product prices are not displayed in this currency. Yet, its definition is preserved and can be used internally by the system when calculating auto prices for products using this currency as the base currency. Due to this reason, deactivating a currency is the recommended way of removing a currency. This is illustrated in Section 15.6.3.3, “Deactivating a currency”.

15.6.1.2. Custom and auto prices

Recall that a multi-price product will always have as many prices as there are currencies. You do not have to enter all these prices manually, although you can. You must specify at least one price per product, the *base custom price* ("base price" for short). Additional custom prices (set by you) are called *non-base custom prices*. Prices in the currencies for which you do not specify a custom price are called *auto prices*. These are calculated from the base price using the currency rates. Custom prices (non-base) are fully independent of the currency rates as well as the base price.

Note that the base price does not have to be in the base currency. For example, suppose the base currency is in EUR but the base price for a product is in USD. To arrive at auto prices, the base price is converted from USD to EUR, then from EUR to the other currencies.

15.6.1.3. Currency exchange rates

A *currency exchange rate* (or "currency rate" for short) specifies the amount of some currency that must be given up in order to obtain one unit of the base currency. Base

prices are multiplied by currency rates to arrive at auto prices. This does not affect custom prices. All exchange rates must be relative to the same base currency.

There are two types of currency rates, based on how they are obtained. An *auto rate* is a value obtained from an external source, such as the European Central Bank. *Custom rates* are in contrast specified manually when editing the currency. This is used when automatic retrieval is not available, or you want to override an auto rate. You may say that auto rates are subject to automatic / dynamic updates, while custom rates must be manually updated.

Auto rates have to be configured by your administrator. Refer to the *Features - Multi-currency* section of http://ez.no/doc/ez_publish/technical_manual for more details.

A *rate factor* is a number by which the currency rate (auto or custom) is multiplied to arrive at the final rate. By default this factor is "1", meaning that the values seen for auto and custom rates are identical to the final rates. If you specify a different rate factor, the currency rate for the specific currency will be adjusted accordingly. For example, you could use a rate factor to compensate for bank expenses related to currency exchange (you cover the extra expense by charging the customer slightly more).

15.6.2. Managing prices

As shown in Figure 15.16, “Attribute of the Multi-price datatype”, auto prices are usually marked as "(Auto)" in the **Object Edit Interface** (or **Content Editing Interface**) while the custom prices are not. There is no special mark for a base custom price because this value usually comes right below the auto prices and right above the non-base custom prices (if there are any).

If you are not satisfied with the auto price value in some particular currency, you can set a non-base custom price instead. This value will be independent of the base price.

Price management means, in most cases, to edit the price attribute of a product in standard edit mode. The exception is when you set the default currency for some class, which is done in the **Class edit** interface.

The different price management scenarios are described in the following sections.

The following configuration applies to our examples:

- Base currency = EUR (allowing currency rate updates from the European Central Bank)
- Default currency (set for the `Product` class) = USD
- Preferred currency = USD

15.6.2.1. Setting the default currency for a class

The default currency of a content class, in our case the `Product` class, is set when editing the class definition by selecting one of the available currencies in the **Default currency** dropdown list:

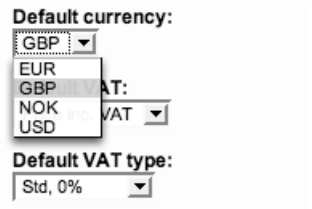


Figure 15.24. Default currency class setting

When a new product is created, it will get a custom price in this currency, and auto prices in all other currencies. Note that you can still modify and remove custom prices for individual Product objects independent of this setting. It simply gives the initial value for new objects.

15.6.2.2. Overriding an auto price by specifying a custom price

Let us start by looking at the "Price" attribute of a multi-price product. In the following screenshot, the base price is in USD ($9.99), while prices in EUR, GBP and NOK are auto prices.

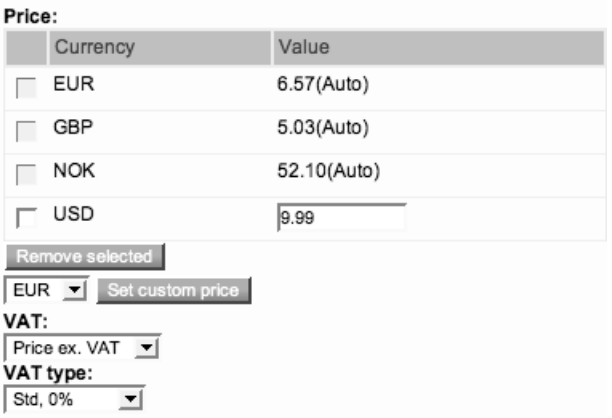


Figure 15.25. Multi-price attribute with base price in USD

Tip

If your auto prices display as 0 in the "Price" attribute, it is usually caused by one of two reasons. First of all, the rate update handler might not be configured properly. In this case, you should consult your system administrator. Alternatively, your auto rates might not be up to date. In that case you should bring up the list of available currencies by clicking the **Currencies** link in the left menu of the **Webshop** tab. If the **Auto rate** column shows "N/A", simply click the **Update auto rates** button, then click the **Update autoprices** button.

Non-base custom prices are independent of the base price. Setting a non-base custom price will fix the price in that particular currency, and it will not be subject to rate updates.

Overriding an auto price with a custom price is done when editing the "Price" attribute of the product. Simply select one of the auto price currencies from the dropdown list, then click the **Set custom price** button. This will reload the page, turning the selected price value into an input field, pre-filled with the old price in that currency. You can then edit the price directly.

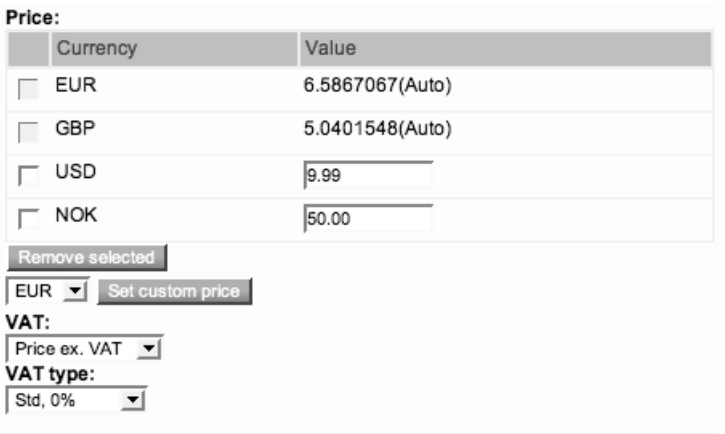


Figure 15.26. Custom price in NOK

15.6.2.3. Changing a base price

If you edit a non-base price, like the one in NOK above, it only affects the price in the specific currency. However, if you edit the base price, it will affect auto prices (but not non-base custom prices). In the screenshot below, we have altered the NOK price to 80.00 and altered the USD price to 20.00. Notice the changes to the EUR and GBP auto prices, relative to the change in the USD price.

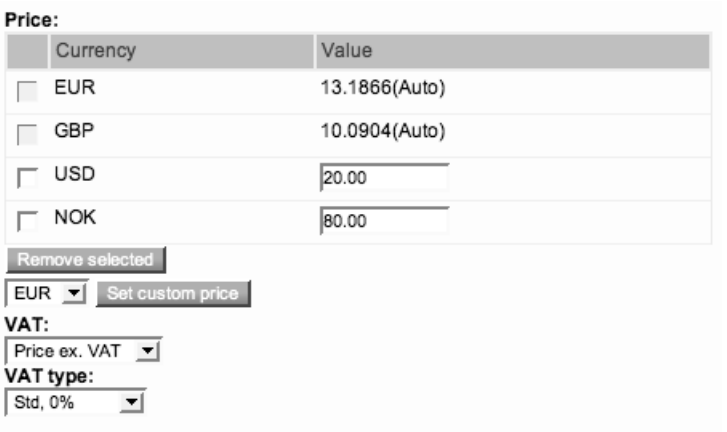


Figure 15.27. Prices updated to reflect changed base price

15.6.2.4. Removing a non-base custom price

The following example shows how to remove a non-base custom price, which reverts it back to an auto price. Simply mark the corresponding checkbox(es), then click the **Remove selected** button. The screenshot below shows the result of removing the custom price in NOK.

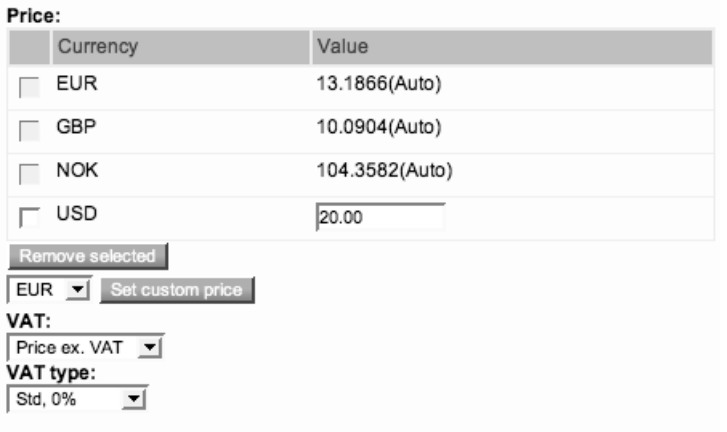


Figure 15.28. Non-base price removed

As shown, the non-base custom price is converted into an auto price, which is calculated instantly from the base price. The other prices are unaffected.

15.6.2.5. Removing the base price

The procedure for removing a base price is identical to removing a non-base price, but the effect is different, because all auto prices are calculated from the base price. When you remove a base price, the system changes the earliest non-base custom price into the new base price and updates the auto prices accordingly. You should go to the **Currency management** interface of the **Setup** tab and click the **Update autoprices** button after you remove a base price. However, if your product does not have any non-base custom prices, it is unable to select a new base price, which leads to the following scenario where all prices are auto prices and reset to 0. The system cannot calculate new auto prices in any currency because there is not base price from which to do so.

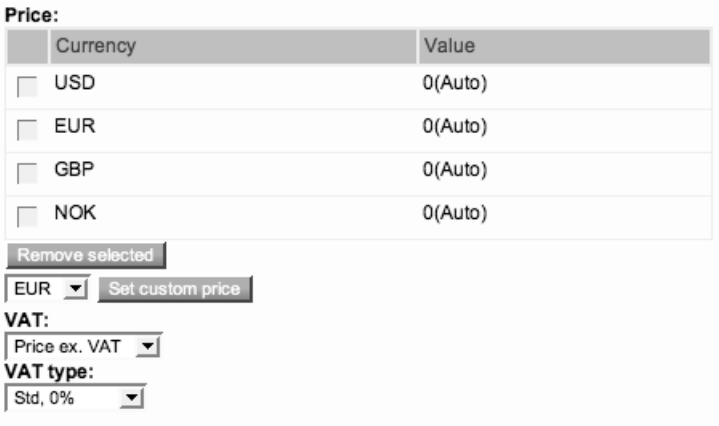


Figure 15.29. Result of removing base price with no substitute

To avoid scenarios like this, it is recommended that you ensure that there is at least one other custom price before you remove a base price. If there is not, you should convert one of the auto prices to a custom price first. Otherwise, you can either discard the draft (cancelling your changes) or simply set a new custom price (which will become the new base price).

15.6.3. Managing currencies

Currencies are managed through the **Currency management** interface, shown in Figure 15.23, “Currency management interface” above. The following sections explain how to create, edit and remove currencies, and how to work with currency rates.

15.6.3.1. Creating a currency

To create a currency, following the procedure below.

1. Open the **Currency management** interface by clicking the **Currencies** link in the left menu of the **Webshop** tab.

2. Click the **New currency** button located below the list of available currencies. This will open the **Currency edit** interface:

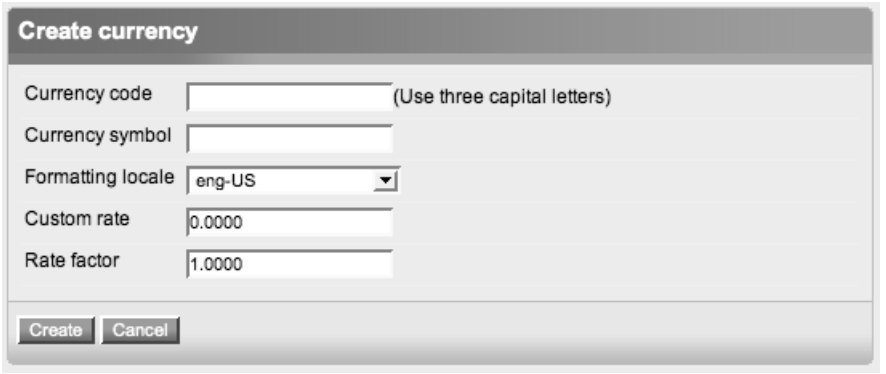


Figure 15.30. Currency edit interface

3. Specify the currency attributes (described in Section 15.6.1, “Currency-related concepts”), then click the **Create** button. The currency code, formatting locale and rate factor are required. If the custom rate is set to "0" (default), the system will use an auto rate for the currency.

The newly created currency will be added to the list of available currencies, sorted alphabetically.

Tip

Creating a new currency triggers the automatic creation of auto prices (with values of zero) in this currency for all your products. When you are finished managing currencies, click the **Update autoprices** button in order to trigger price updates. Otherwise, your shop prices will be incorrect.

15.6.3.2. Editing a currency

You can update a currency by clicking the **Edit** button for the desired currency in the **Currency management** interface. Simply change the currency attributes as desired and click the **Store changes** button.

Rate information can be updated directly in the **Currency management** interface, without entering edit mode. You should click the **Apply changes** button and relevant update button when you are finished making your changes.

15.6.3.3. Deactivating a currency

You can remove a currency by marking the corresponding checkbox in the **Currency management** interface and clicking the **Remove selected** button. However, this is not recommended, since removing a currency also removes prices in this currency for all products, and this is problematic for products having base prices in the removed currency.

Because of this, you should deactivate rather than delete unwanted currencies. This is similar to disabling a user account (see Section 5.7.1.5, “Disabling and enabling users”), where the internal references are preserved while the user is prevented from logging in. Simply select "Inactive" in the **Status** column of the **Currency management** interface for the currency you want to deactivate, then click the **Apply changes** button.

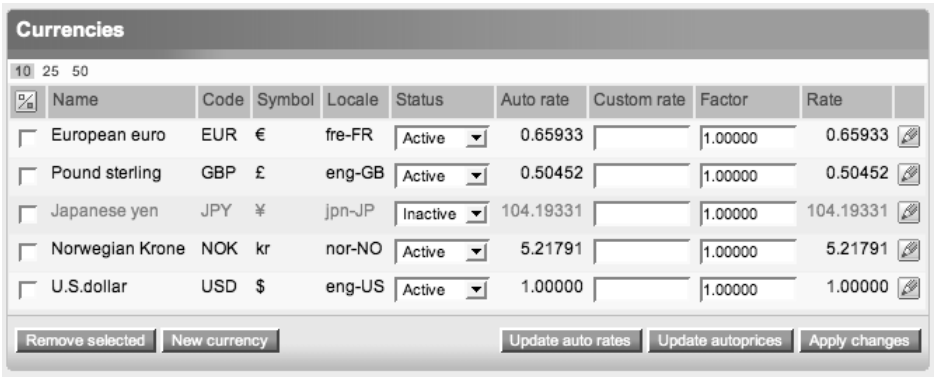


Figure 15.31. Inactive currency

15.7. VAT charging system

Recall that Value Added Taxes (VAT) represent one of the six Webshop components, illustrated in Figure 15.1, “Webshop components”, and that they affect product prices by specified percentages. VAT charging is done with *VAT types*. The e-commerce subsystem of eZ Publish has three different approaches to VAT charging:

- VAT per product
- Country-dependent VAT
- Extended VAT

The above approaches are listed in order of complexity of the VAT charging logic, with *VAT per product* being the simplest. It does not require additional management beyond editing the "Price" attribute of individual products. This section covers VAT types, VAT charging rules and the above approaches, except for *extended VAT*. The latter is used in sites requiring more sophisticated VAT charging logic than country-dependent VAT offers.

The logic of extended VAT depends on site-specific custom solutions (see "In Depth: Payment, shipping and handlers" in the introduction to this chapter) and is beyond the scope of this book.

15.7.1. VAT types

A *VAT type* is basically a name and a percentage, used by the "Price" and "Multi-Price" datatype. It is assigned to a product by editing the "Price" attribute:

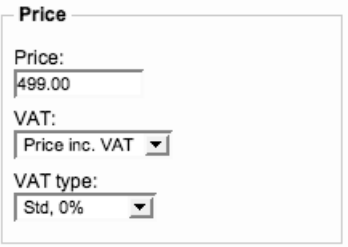


Figure 15.32. VAT type in Price attribute

The screenshot above shows that the price "499.00" includes VAT, and the VAT type is "Std, 0%". This implies that the price excluding VAT is also 499.00. In fact, this is the only built-in VAT type, ensuring that the system meets the requirement of having at least one VAT type. VAT types are displayed and managed in the **VAT type management** interface. To access the interface, click the **VAT types** link in the left menu of **Webshop** tab.

Figure 15.33. VAT type management interface

You can add, modify and delete VAT types in this interface, as described shortly.

15.7.1.1. Static and dynamic VAT types

A VAT type is either static or dynamic. A *static VAT type* is what we have described above - a fixed percentage rate. All VAT types listed and managed through the **VAT type management** interface are static.

In addition to the static VAT types, there is one *dynamic VAT type*. When a product uses the dynamic VAT type, this is replaced during the checkout process with a static VAT type, according to defined VAT rules. In other words, it is a special-purpose VAT type assigned to products that will be affected by VAT rules (see Section 15.7.3.2, “VAT charging rules”), according to the VAT charging logic.

The dynamic VAT type is not listed in the **VAT type management** interface and it does not store any fixed VAT rate, since it is to be replaced with a static VAT type. Also, it can only be assigned to products with prices that are entered excluding VAT. When you are in edit mode, the dynamic VAT type is represented by the text "Determined by VAT charging rules". You will not see this option if your VAT approach is "VAT per product", which is the default configuration in eZ Publish. You will learn more about dynamic VAT in Section 15.7.3, “Country-dependent VAT”.

15.7.1.2. Creating a VAT type

In order to make use of Value Added Taxes in your webshop, you need to create additional static VAT types, as explained by the following procedure.

1. Access the **VAT type management** interface by clicking the **VAT types** link in the left menu of the **Webshop** tab.

2. Click the **New VAT type** button. The system will add a new list item with a default percentage of zero:

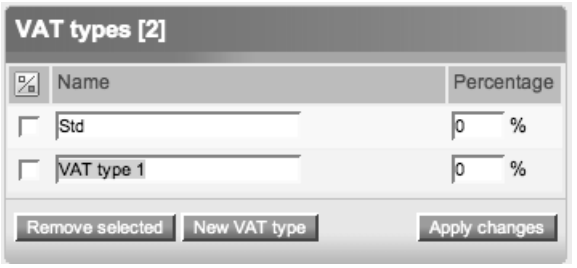


Figure 15.34. New VAT type

3. Edit the name and percentage as desired.

4. To add more VAT types, click the **New VAT type** button again for each additional type. Click the **Apply changes** button to store the new VAT type(s).

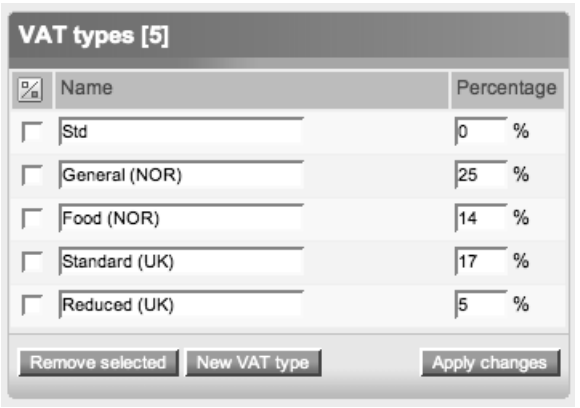


Figure 15.35. VAT type management interface - 4 VAT types added

15.7.1.3. Editing a VAT type

Editing an existing static VAT type is done by directly modifying the name and / or percentage within the **VAT type management** interface (as when creating a new type), then saving it by clicking the **Apply changes** button.

Tip

When you edit a VAT type, it is updated for all price attributes that make use of that particular type. This percentage is very important, and should only be modified to reflect public VAT regulations (or typos).

15.7.1.4. Deleting a VAT type

Deleting a static VAT type requires some extra care. Recall that you are not permitted to remove a content object from its main location, nor can you remove a specific translation from a multilingual object if it is the main language for that object. A similar restriction applies to VAT types: you cannot remove a VAT type that is used as "default VAT type" for a product class. Furthermore, it is not recommended that you remove a VAT type that is being used by products or a VAT charging rule.

To check what the default VAT type is, access the **Class view** interface (as described in Section 3.2, “Viewing and editing content classes”) for the `Product` class, and locate the "Price" attribute:

Figure 15.36. Price attribute specification

To remove a VAT type, first access the **VAT type management** interface by clicking the **VAT types** link in the left menu of the **Webshop** tab. Mark the checkboxes corresponding to the VAT types you wish to delete, then click the **Remove selected** button.

15.7.2. VAT per product

VAT per product is the simplest of the VAT charging approaches, and can be managed exclusively from the Administration Interface. In other words, webshop managers do not need to worry about extra installation or configuration.

The first step is to set up the static VAT types, as described in the previous section. The second step is to assign a VAT type to a product.

15.7.2.1. Assigning a VAT type to a product

The following procedure shows how to assign a VAT type to a particular product. You can alternatively change the default VAT type for a product class (see Figure 15.36, “Price attribute specification”) through the **Class edit** interface.

1. Locate the product for which you wish to assign a VAT type. The **Products overview** interface (Section 15.4.1, “Products overview interface”) provides quick access to all products. You can alternatively navigate the node tree.

2. Initiate editing (by bringing up the **Object Edit Interface**).

3. Select one of the options listed in the **VAT type** dropdown list in the "Price" attribute.

4. Click the **Send for publishing** button to store your changes.

Tip

You can also edit the product in the **Content Editing Interface**, but in this case you have to locate it manually since the **Products overview** interface is not available.

15.7.3. Country-dependent VAT

Country-dependent VAT is the second VAT charging approach, and is used in webshops where you need to differentiate VAT based on customers' locations. In eZ Publish terms, different VAT rules (described below) are applied to products according to product categories and customers' countries. For example, if you are a Norwegian company shipping books to both customers in Norway as well as other parts of Europe, you need to charge taxes based on the tax regulations of the country in which the buyer lives.

Recall that all products that are to be assigned country-dependent VAT must have "Determined by VAT charging rules" (representing the dynamic VAT type) set as the VAT type, and they must have prices excluding VAT.

Country-dependent VAT requires some configuration in addition to the default settings, and should be set up by your site administrator as described in the in-depth block below.

In Depth: Setup for country-dependent VAT

Below, we list the attribute and settings requirements in order to configure a site to use the country-dependent VAT approach:

- Your user class must include an attribute of the "Country" datatype (see Section 15.5.4, “Country datatype”). Otherwise, it is impossible to assign a country to a customer. The attribute identifier of the "Country" attribute must be specified in the "UserCountryAttribute" setting in the "[VATSettings]" block of `shop.ini`.

- The product class(es) must include an attribute of the "Product category" datatype (see Section 15.5.3, “Product category datatype”). Otherwise, it is impossible to assign a category to a product; different VAT rules are assigned according to product categories (see Section 15.7.3.2, “VAT charging rules”). The attribute identifier of the "Product category" attribute must be specified in the "ProductCategoryAttribute" setting in the "[VATSettings]" block of `shop.ini`.

- The system must use the built-in default VAT handler. To do so, "ezdefault" must be specified in the "Handler" setting in the "[VATSettings]" block of `shop.ini`.

The following sections cover the product category attributes used with country-dependent VAT, and describes how to work with VAT charging rules.

15.7.3.1. Managing product categories

We saw previously in Section 15.5.3, "Product category datatype" that a *product category* is a Webshop-specific datatype, and you can assign a category to a particular product when it is being edited. Product categories are basically only names, yet they enable you to charge different VAT for different groups of products. Recall that the VAT type is assigned directly when using the VAT per product approach. Country-dependent VAT decouples the static VAT types from the individual products (otherwise you could not make it country-dependent). However, product categories are needed to ensure that the correct VAT type is selected.

Tip

Carefully consider which product categories are needed in order to map to appropriate VAT types. Over-categorizing only makes your webshop more complex to manage.

The following subsections describe how to add, edit, remove and assign product categories. This is done in the **Product categories** interface of the **Webshop** tab (opened by clicking the corresponding link in the left menu), except for assigning categories, which is done in edit mode for the products' content objects.

15.7.3.1.1. Creating a product category

Follow the procedure below to add a new product category to your webshop. It is very similar to the one for creating a new VAT type. Note that the e-commerce system does not contain any pre-configured product categories. Below we have already added "DVD" and "Food".

1. Access the **Product categories** interface:

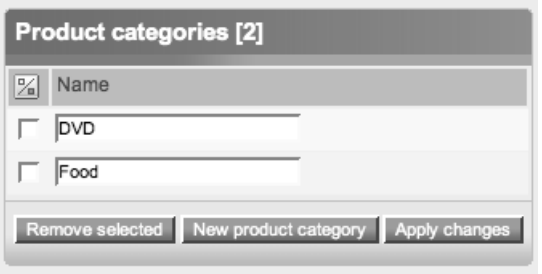


Figure 15.37. Product categories interface

2. Click the **New product category** button. The system will add a new list item.

3. Edit the name of the new category.

4. To add more categories, click the **New product category** button again. Click the **Apply changes** button to store the newly created categories.

The categories are listed in alphabetical order in the **Product categories** interface, and new categories are automatically sorted when you apply the changes:

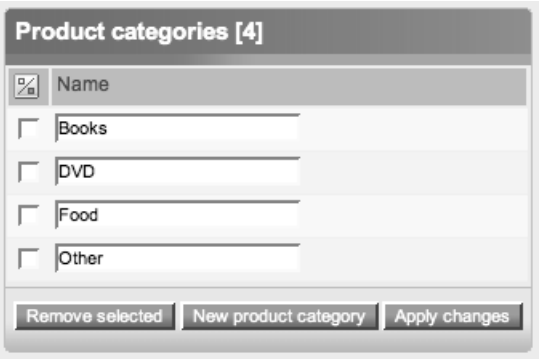


Figure 15.38. Product categories interface - 2 new categories

15.7.3.1.2. Editing a product category

Since a product category is merely a name, you can edit it by accessing the **Product categories** interface, changing the name for one or more categories, then clicking the **Apply changes** button.

15.7.3.1.3. Deleting a product category

Removing a category that is assigned to your products and / or used by your VAT rules is possible but not recommended. It is better to rename existing categories or rules. Removing a category has the following consequences:

- The category attribute of the products that make use of the removed category will be unset (no category chosen). The content objects themselves are not deleted.

- VAT rules specified for this particular category will be modified or deleted (depending on whether the rule applies to one or more categories; see Section 15.7.3.2, “VAT charging rules”).

To remove one (or more) product categories, first access the **Product categories** interface. Then, mark the checkboxes corresponding to the categories you wish to delete and click the **Remove selected** button.

15.7.3.1.4. Assigning a product category

The following procedure outlines how to assign a category to a particular product. Note that it should only be used when the dynamic VAT type is used, as indicated by the "Determined by VAT charging rules" option for the VAT type.

1. Locate the product to which you wish to assign a VAT type. The **Products overview** interface (Section 15.4.1, “Products overview interface”) provide quick access to all products. You can also navigate the node tree to locate the product.

2. Initiate editing (by bringing up the **Object Edit Interface**).

3. Select one of the options listed in the **Category** dropdown list in the "Product category" attribute:

Figure 15.39. Product category attribute - select from list

4. Store the assignment by clicking the **Send for publishing** button.

Tip

You can also edit the product in the **Content Editing Interface**, but in this case you have to locate it manually since the **Products overview** interface is not available.

15.7.3.2. VAT charging rules

A *VAT charging rule* (or just "VAT rule" for short) consists of three parts: a user country, one or more product categories, and a VAT type. Its purpose is to determine which static VAT type to apply based on the customer's country and the product's category. The following screenshot shows the **VAT rule management** interface, accessed by clicking the **VAT rules** link in the left menu of the **Webshop** tab. Initially, your e-commerce system does not have any predefined rules:

Figure 15.40. VAT rule management interface

Rule matching is based on "country-category" pairs, such as "Norway-Food", "Norway-Books" and "Any-Books". When multiple matches exist, the rule with the more exact match for a "country-category" pair is credited the higher priority. The following table shows the possible combinations, listed from the highest to lowest match priority. The default VAT rule specified for "Any" country and "Any" category determines which VAT rate to use in case if all other VAT rules do not match.

Table 15.2. VAT rule matching for country-category pairs

Country	**Category**	**Example**
Exact match	Exact match	Norway-DVD
Exact match	Weak match	Norway-Any
Weak match	Exact match	Any-DVD
Weak match	Weak match	Any-Any

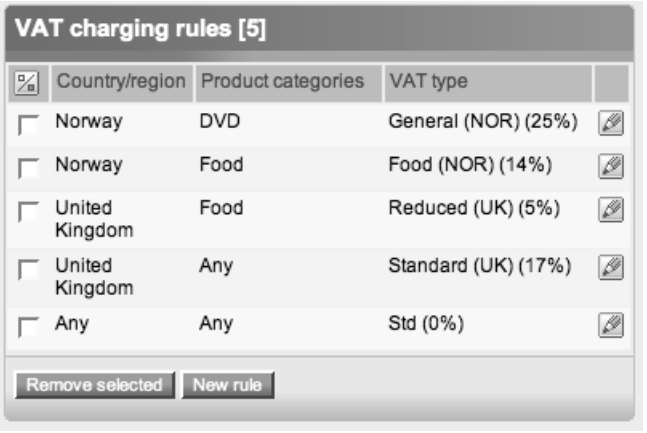


Figure 15.41. VAT rule management interface - examples

The following sections show how to manage VAT rules. Note that the above-explained matching logic takes care of the assignment of rules, so there is no explicit "assign" operation for VAT charging rules that webshop managers have to take care of.

15.7.3.2.1. Creating a VAT rule

To add a new VAT rule, follow the procedure below.

1. Open the **VAT rule management** interface by clicking the **VAT rules** link in the left menu of the **Webshop** tab.

2. Click the **New rule** button. The system will bring up the **VAT rule edit** interface:

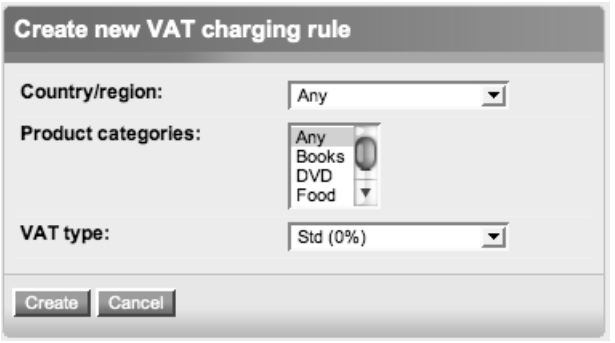


Figure 15.42. VAT rule edit interface

3. Specify a country / region and a VAT type by selecting from the dropdown lists. Select one or more product categories from the selection list.

4. Click the **Create** button to add the new rule to the system. It will then show up in the **VAT rule management** interface. If there is no default rule ("Any-Any"), you will get a warning when adding a more specific rule, as shown in the following screenshot. Simply create the default rule as the system recommends.

Figure 15.43. Default VAT rule missing

15.7.3.2.2. Editing a VAT rule

VAT rules are edited by accessing the **VAT rule edit** interface for the desired rule (click the corresponding **Edit** button in the **VAT rule management** interface). Simply make your changes to the parameters and click the **Store changes** button.

15.7.3.2.3. Deleting a VAT rule

To remove a VAT rule, first open the **VAT rule management** interface. Then, mark the checkboxes corresponding to the rules you wish to delete and click the **Remove selected** button.

Note that you should not remove the default VAT rule ("Any-Any"), unless it is the last existing rule and you are not going to use country-dependent VAT anymore.

15.8. Discount rules

The previous section described how VAT may alter the price of a product. A *discount rule* also affects a product's price. A discount rule belongs to a *discount rule group*, which is applied to valid users and user groups. This is similar to assigning roles that consists of policies to users and user groups.

The following screenshot shows the rules within a specific group, indicating that partners get a 20% discount on all products.

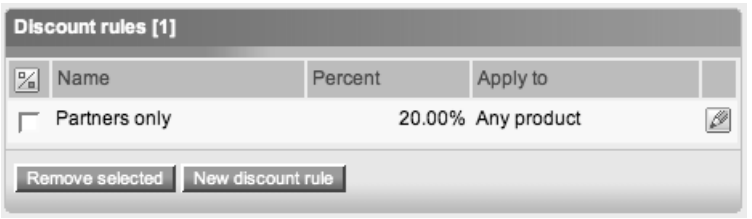


Figure 15.44. Discount rules panel

A discount rule reduces the final price of certain products by a percentage. By default, all products are affected, but you can specify the users to which the rule applies, and / or limit a discount rule to a subset of the products in your shop. Once set, the rule is always active. In other words, you cannot toggle it on and off, and must remove it if you do not want it to be used.

Discounts are accessible in the **Webshop** tab by clicking the **Discounts** link in the left menu. This will open the **Discount management** interface:

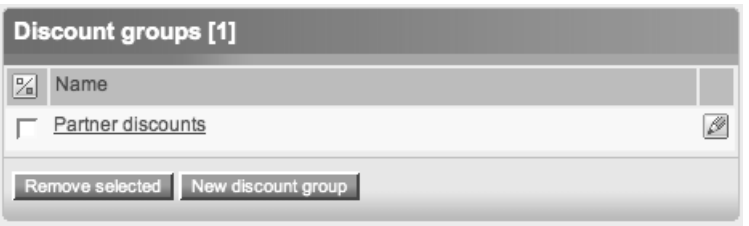


Figure 15.45. Discount management interface

15.8.1. Discount rule limitations

Readers familiar with user management might recall that it is possible to set limitations for policies and roles. Discount rule limitations work in the same way, and have a similar editing interface - limitations for a discount are set when creating or editing the rule.

A *Product type limitation* restricts the discount to products of one or more specific classes. By default, the type is set to "Any", making the rule applicable to all products.

A *Section limitation* restricts the discount to products located within one or more specific sections, such as the Standard and Restricted sections. Section management was covered in Chapter 4.

Each discount rule also has a product list, to apply the rule to specific products, as illustrated in Section 15.8.2.4, “Discount rule product list”.

It is not possible to limit a discount to a product category - product categories are used for VAT rules only. If you wish to limit discounts based on categories, either use custom

product classes, or add individual products to the product list of a particular discount. The procedure for assigning limitations is found in Section 15.8.2.3, “Assigning discount rule limitations”.

15.8.2. Managing discounts

The following subsections describe and illustrate how to work with discounts. The procedures show you how to manage discount groups and individual rules. To access the rules within a specific group, click on the discount group's name in the **Discount management** interface.

15.8.2.1. Discount rule groups

To create a new discount rule group, first access the **Discount management** interface by clicking the **Discounts** link in the left menu of the **Webshop** tab. Then, click the **New discount group** button. Enter a name for the new discount group, as shown in the following screenshot. When done, click the **OK** button.

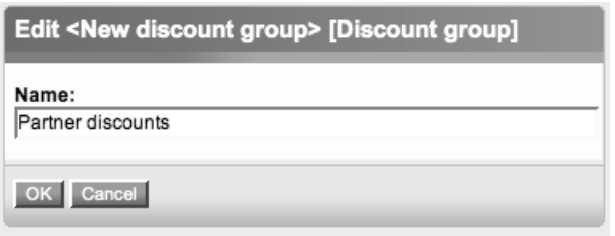


Figure 15.46. Edit discount group

The newly created discount rule group will show up in the **Discount management** interface.

You can edit existing discount groups by clicking the **Edit** button corresponding to the group within the **Discount management** interface. Selected groups can be deleted by marking the checkbox(es) to the left of the groups and clicking the **Remove selected** button. Alternatively, you can edit or remove a specific group by clicking the corresponding buttons when you are viewing the group (click the group name in the **Discount management** interface).

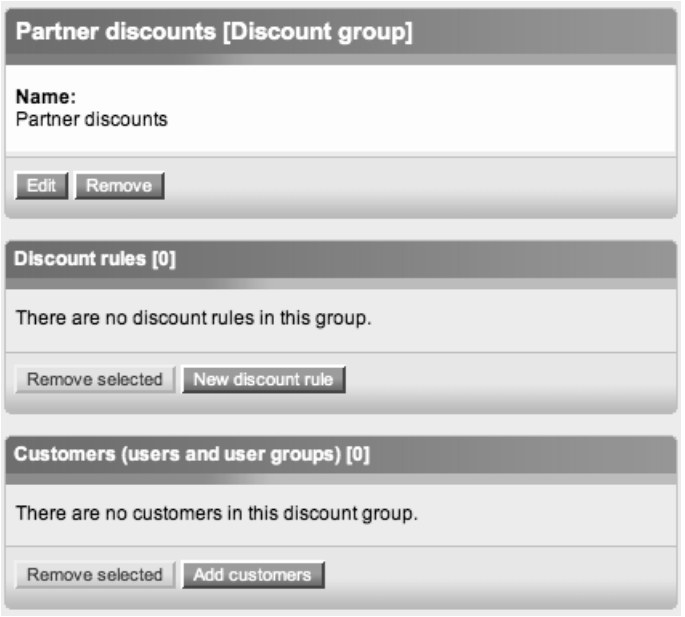


Figure 15.47. Discount group view

As shown in the above screenshot, you can manage rules and assign them to customers when viewing a group. These operations are described next.

15.8.2.2. Adding a new discount rule

In order to create a new discount rule, you first have to select the discount group in which to place it. If this is the first time you are setting up discounts, you need to create the group(s) first, as previously described. Then, follow the procedure below.

1. Access the **Discount management** interface and select a group for your new rule.

2. In the **Discount rules** panel, click the **New discount rule** button. This will open the **Discount editing** interface:

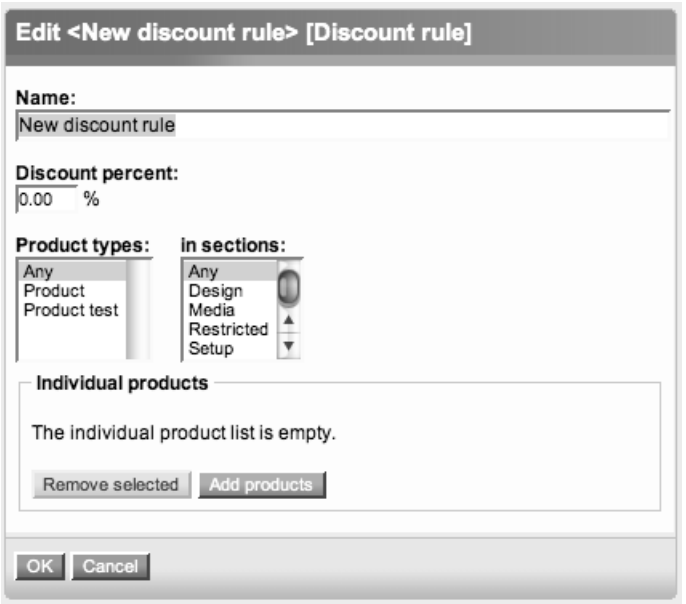


Figure 15.48. Discount editing interface

3. Enter a name and a discount percentage. Optionally assign limitations (see Section 15.8.2.3, “Assigning discount rule limitations” and Section 15.8.2.4, “Discount rule product list”). As an example, we name the rule "Partners only", supply "20" as the discount percentage and leave "Any" as the choice for the limitations, making the rule apply to all products. You can also add products under the rule, as described in Section 15.8.2.4, “Discount rule product list”.

4. Click the **OK** button to store the rule.

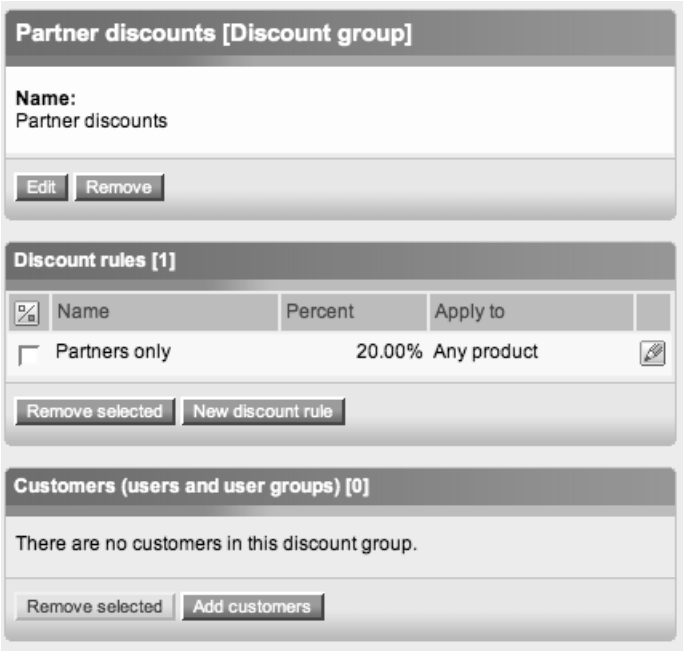


Figure 15.49. Discount management interface - new discount rule

As shown in the above screenshot, you can view the new rule in the **Discount rule** panel (there is no special interface for viewing a single discount rule). Click the **Edit** button to change the name, percentage or limitations for a rule. If you want to cancel a particular discount, mark the corresponding checkbox and click the **Remove selected** button.

15.8.2.3. Assigning discount rule limitations

You can assign limitations and add products to the list of product rules when creating or editing a specific discount. In other words, you can create a discount, and assign limitations and products to it in a single operation. The following procedure illustrates how to add a discount rule limitation to an existing discount rule.

1. Locate the discount rule for which you want to set a limitation by selecting a group in the **Discount management** interface, then click the **Edit** button to the right of the rule in the **Discount rules** panel.

2. The **Discount editing** interface will look something like Figure 15.48, “Discount editing interface”. Select a class, such as "Product", in the **Product types** selection list. You can also select a section, such as "Standard", within the corresponding selection list. Keep in mind that you can select multiple items within each selection list by using the Shift or Control buttons on your keyboard.

3. Click the **OK** button to save your changes. The limitation(s) can be seen in the **Discount rules** panel:

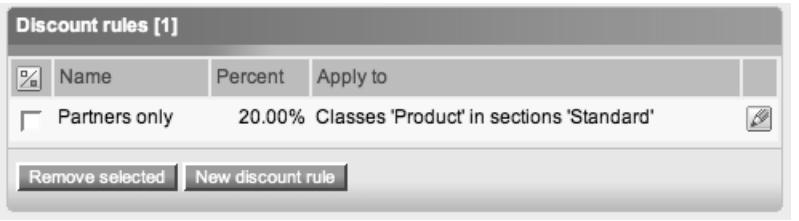


Figure 15.50. Discount rules panel - rule with limitations

You can add, remove and modify limitations of existing discount rules by re-editing the rule. To cancel the entire discount, you have to delete the rule.

15.8.2.4. Discount rule product list

We indicated previously that each rule has an individual *product list*. You can add and remove individual products (in other words, Product content objects) to this list, both when adding and editing the rule. The following procedure explains how to add products to the product list in an existing rule.

1. Access the **Discount editing** interface for the desired discount rule.

2. Click the **Add products** button near the bottom of the editing interface.

3. This will open the **Browse** interface. Use this to navigate to the desired products and mark the corresponding checkboxes. Then, click the **Select** button.

   Note that you can only select multiple products in the same operation if they are located within the same container. To select multiple products from different locations, repeat steps 2 and 3.

4. The product(s) will show up in the product list:

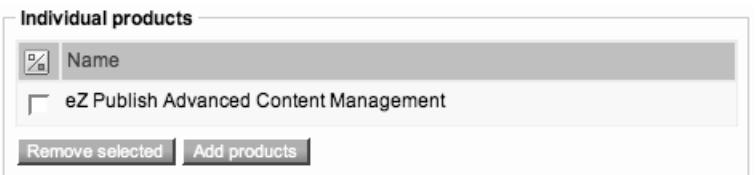


Figure 15.51. Discount rule product list with one product

When you are finished adding or removing products, click the **OK** button to store your changes.

15.8.2.5. Connecting a group of discounts to specific users

You can specify the customers to which a group of discount rules should apply. *Customer-specific discounts* are viewed and managed through the **Customers** panel when viewing a discount group:

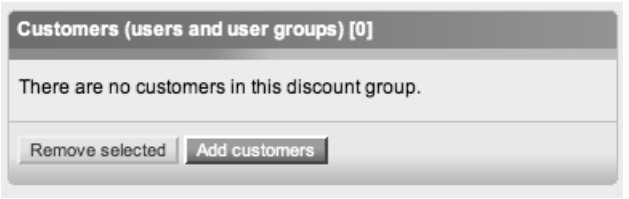


Figure 15.52. Customers panel - empty

By default (if no customers are specified), all discounts within the group apply to all customers. To add users and / or user groups, first select a discount group in the **Discount management** interface, then click the **Add customers** button in the **Customers** panel. This will open the **Browse** interface, with the **User accounts** tab pre-selected. Navigate to locate your target customer(s), mark the corresponding checkbox(es), then click the **Select** button.

The **Customers** panel will reload to show the selected user(s) and user group(s):

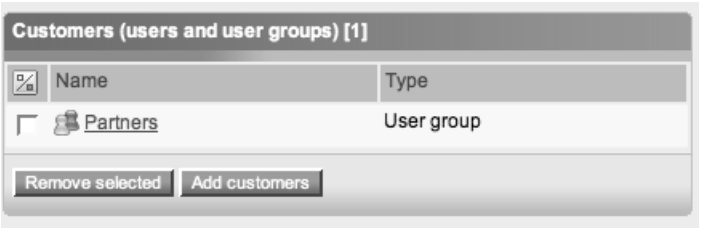


Figure 15.53. Customers panel - Partners user group

15.9. Example: Working with multi-options

The concept of *multi-option attributes* using the "Multi-option2" datatype was introduced in Section 15.5.2.3, “Multi-option2 datatype”. This section explores such attributes further and shows how to work with them in edit mode by going through an example for a scrapbook product.

Let us start from scratch with a newly created Product object, specifying the traits to our needs. Recall that the `Product` class for sites that use the Website Interface still have an attribute of the deprecated "Multi-option" datatype instead of the "Multi-option2" datatype. We have modified the default `Product` class to use an attribute of the "Multi-option2" datatype for this example.

Tip

Plan your trait structure before creating the product and editing the multi-option attribute by simply drawing up a multi-level table on paper. This will help you identify groups, multi-option sub-levels, multi-option sets and individual options.

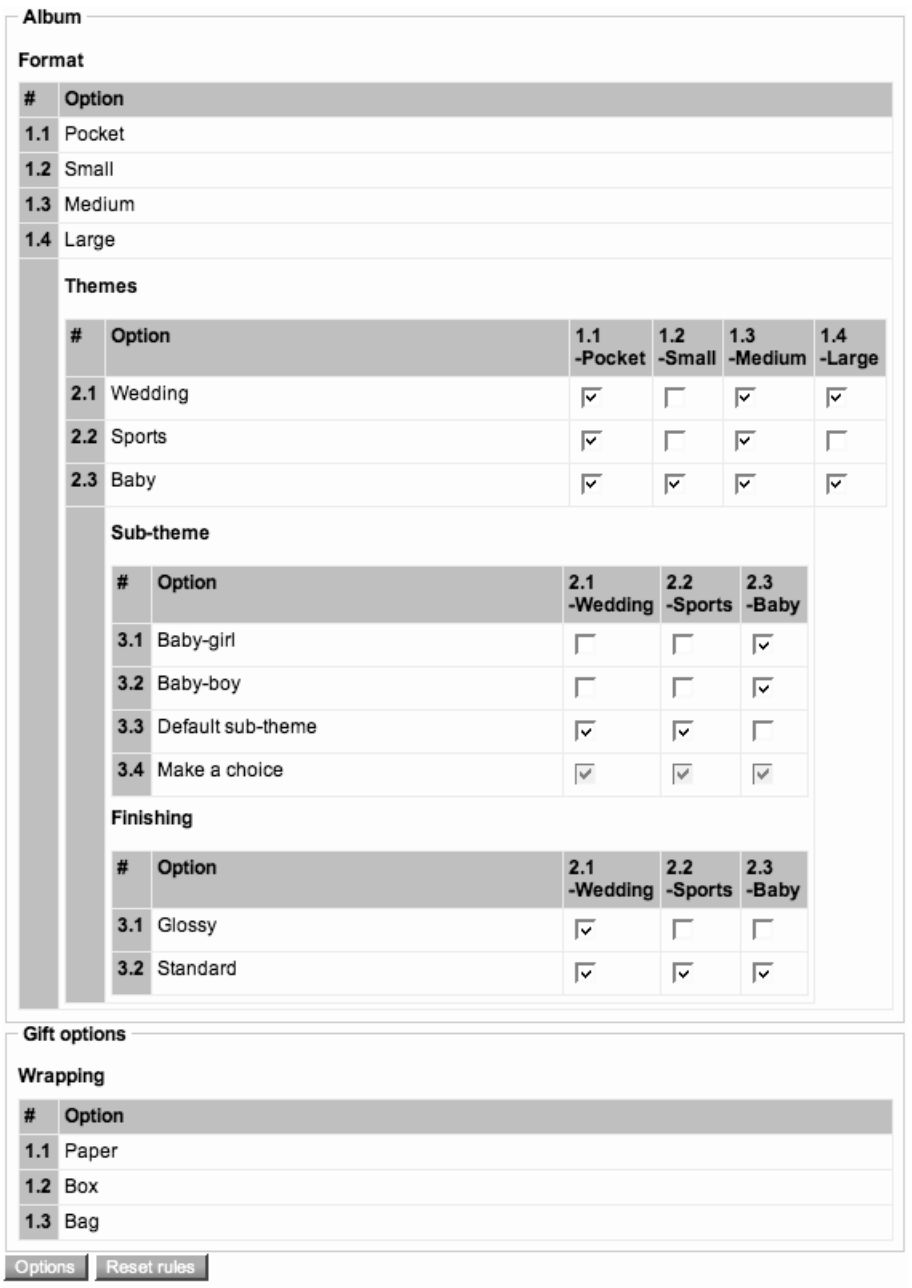
Our scrapbook has the following properties:

- It is available in 4 sizes (pocket, small, medium, large).
- It is available in 3 themes (wedding, sports, baby). The wedding theme is not available in the small size; the sports theme is only available in the pocket and medium sizes; and the baby theme is available in all sizes.
- Scrapbooks with the baby theme are available in 2 sub-themes: one for girls and one for boys.
- All themes come with standard finishing. However, the wedding theme can also come in glossy finishing.
- There are 3 wrapping options: paper, box and bag.

These properties correspond to a trait structure for our multi-option attribute as follows:

- There are 2 top-level groups: one for customizing the features of the scrapbook (we will name this the "Album" group) and one for the wrapping options (we will name this the "Gift options" group).
- Within the "Album" group, there will be separate sub-levels consisting of multi-options for the format, theme, sub-theme and finishing - the latter two will be on the same nesting level, since they both depend on the chosen theme.

Our final result is summarized in the screenshot below, which shows the groups, levels of multi-options, and dependency rules for the scrapbook in *rule mode* (this mode is described in Section 15.9.5, “Dependency rules for multi-options”):



Album

Format

#	Option
1.1	Pocket
1.2	Small
1.3	Medium
1.4	Large

Themes

#	Option	1.1 -Pocket	1.2 -Small	1.3 -Medium	1.4 -Large
2.1	Wedding	☑	☐	☑	☑
2.2	Sports	☑	☐	☑	☐
2.3	Baby	☑	☑	☑	☑

Sub-theme

#	Option	2.1 -Wedding	2.2 -Sports	2.3 -Baby
3.1	Baby-girl	☐	☐	☑
3.2	Baby-boy	☐	☐	☑
3.3	Default sub-theme	☑	☑	☐
3.4	Make a choice	☑	☑	☑

Finishing

#	Option	2.1 -Wedding	2.2 -Sports	2.3 -Baby
3.1	Glossy	☑	☐	☐
3.2	Standard	☑	☑	☑

Gift options

Wrapping

#	Option
1.1	Paper
1.2	Box
1.3	Bag

Options | Reset rules

Figure 15.54. Multi-option attribute - scrapbook example end result

15.9.1. Adding options

Initially we have one group, with one multi-option containing one option, all with blank fields. Before adding elements, let us simply name the group ("Album") and multi-option ("Format"). Then, we add our options to the "Format" multi-option. To add an additional option, simply click the **Add option** button for each extra option. The screenshot shows the normal view for a multi-option attribute in edit mode, where we have named the group and multi-option, named the first option "Pocket", added another option named "Small", and added but not named a third option.

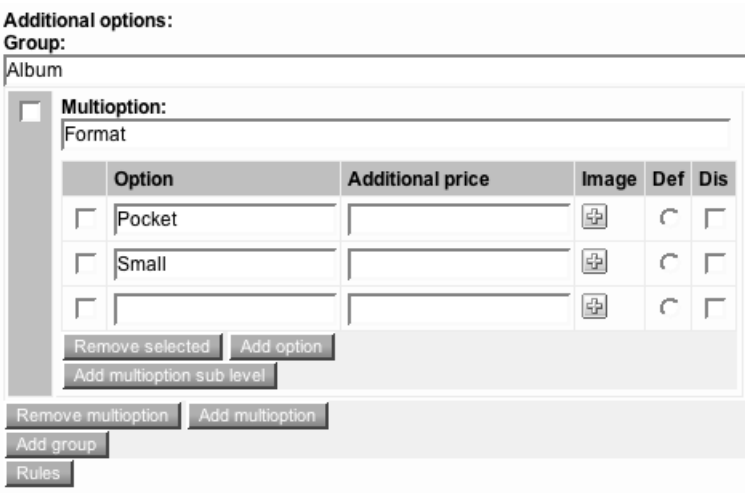


Figure 15.55. Add options to the "Format" multi-option

To complete the format options, the blank option is named "Medium" and we add a fourth option "Large". Before going any further with album-related options, the next section describes how to add the "Gift options" group.

Tip

Adding a new multi-option is similar to adding a single option. Simply click the **Add multioption** button.

Tip

The editing interface will reload each time you add or remove an element from the multi-option attribute. We recommend specifying the structure and providing names before starting to work with prices, dependency rules, default selections and disabling options.

Groups, multi-options and options are removed by marking the corresponding checkboxes and clicking the corresponding **Remove selected** button.

15.9.2. Adding a new multi-option group

New groups are added by clicking the **Add group** button, near the bottom of the attribute editing area. The newly created group will be added below the existing group, with blank input fields as shown below:

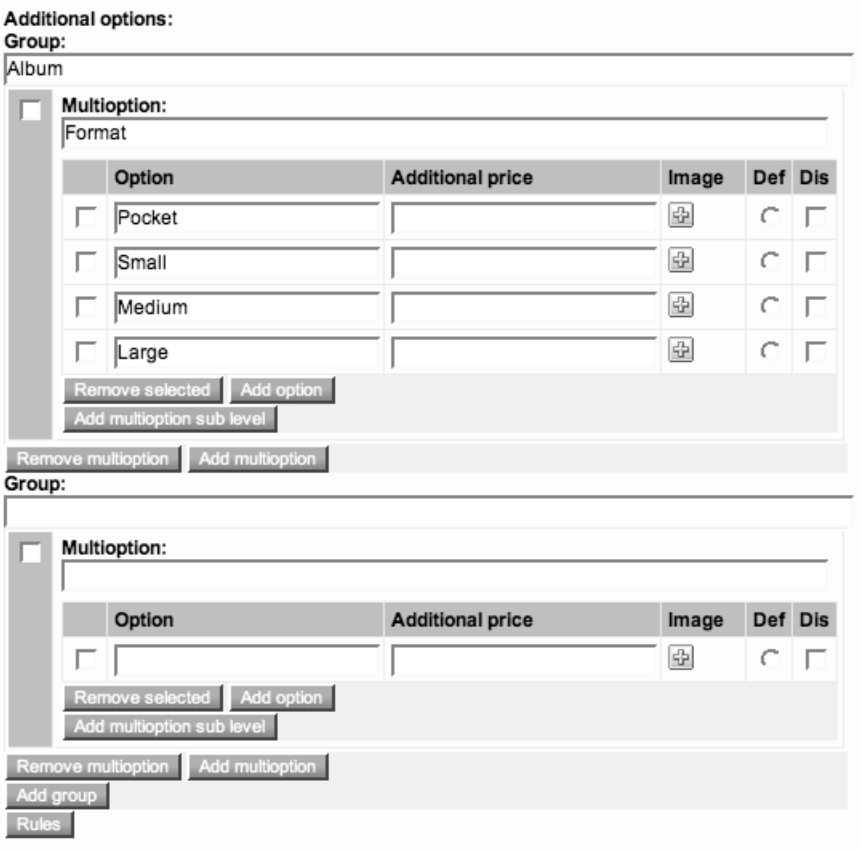


Figure 15.56. New multi-option group

We then name the new group "Gift options", name the single multi-option "Wrapping" and add options for paper, box and bag wrapping.

15.9.3. Multi-option nesting

The **Add multioption sub level** button is located at the bottom of a multi-option that does not yet have any sub-levels. Clicking this button will create a new sub-level below the targeted multi-option. In the case where the button is not shown for a level, it means that it already has a sub-level.

Each level, regardless of whether it has a sub-level, has **Remove multioption** and **Add multioption** buttons. They enable you to add and remove multi-options located at the different levels. Note that removing a multi-option will also remove all its options and sub-levels of multi-options and options.

Tip

The maximum number of levels is 15 but it is not recommended to use more than 10 levels.

15.9.3.1. Adding a multi-option sub-level

The key to adding a new multi-option sub-level is to select the correct multi-option to start with. You have probably observed the vertical gray areas on the left in the previous screenshots. Simply locate the name input field of the multi-option under which you want to add a sub-level, follow the gray area downward until it ends, and right next to it you will see the relevant button.

For our scrapbook, click the **Add multioption sub level** button underneath the "Format" multi-option. The newly created sub-level will be added where we can input the information for our wedding, sports and baby themes.

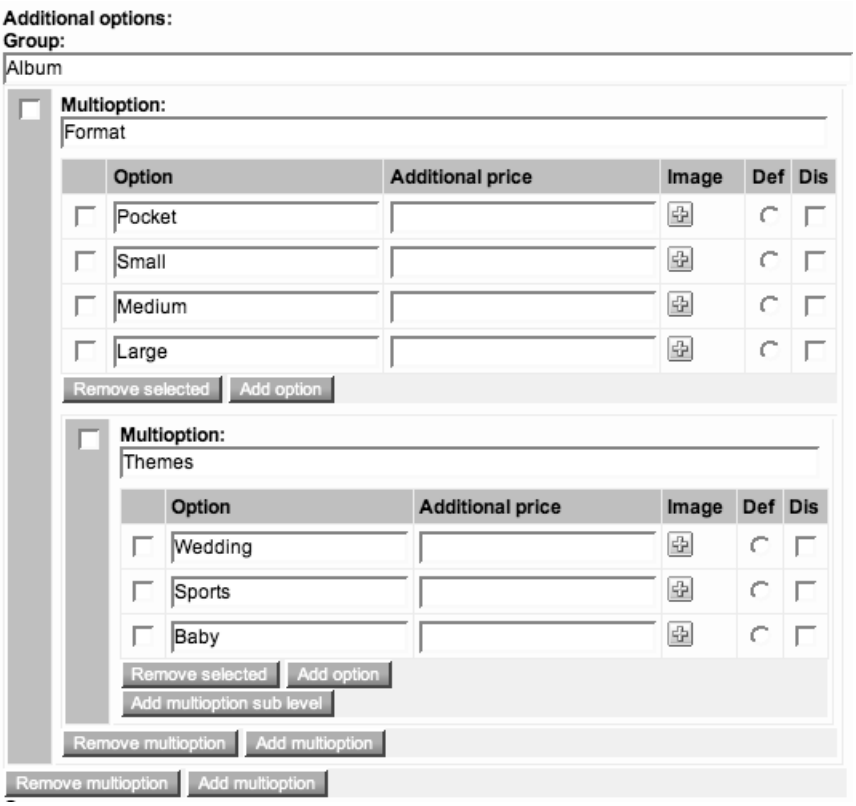


Figure 15.57. New multi-option sub-level with options

To add the remaining elements to our trait structure, the following is needed:

1. Another sub-level beneath "Themes", called "Sub-theme" (click the **Add multioption sub level** button for the "Themes" multi-option). The "Sub-theme" multi-option will contain the options "Baby-girl", "Baby-boy", "Make a choice" and "Default sub-theme". The latter two options will be explained later.

2. A "Finishing" multi-option at the same level as the "Sub-themes" multi-option (click the **Add multioption** button corresponding to the "Themes" level). The "Finishing" multi-option will contain the options "Glossy" and "Standard".

The result, except for the "Gift options" group, is shown in the screenshot below. At this time you should enter values in the **Additional price** input fields for the options that affect the price. In our example, only some size, finishing and gift options affect price, while the theme does not. You can optionally add an image corresponding to each option by clicking the relevant button under the **Image** column.

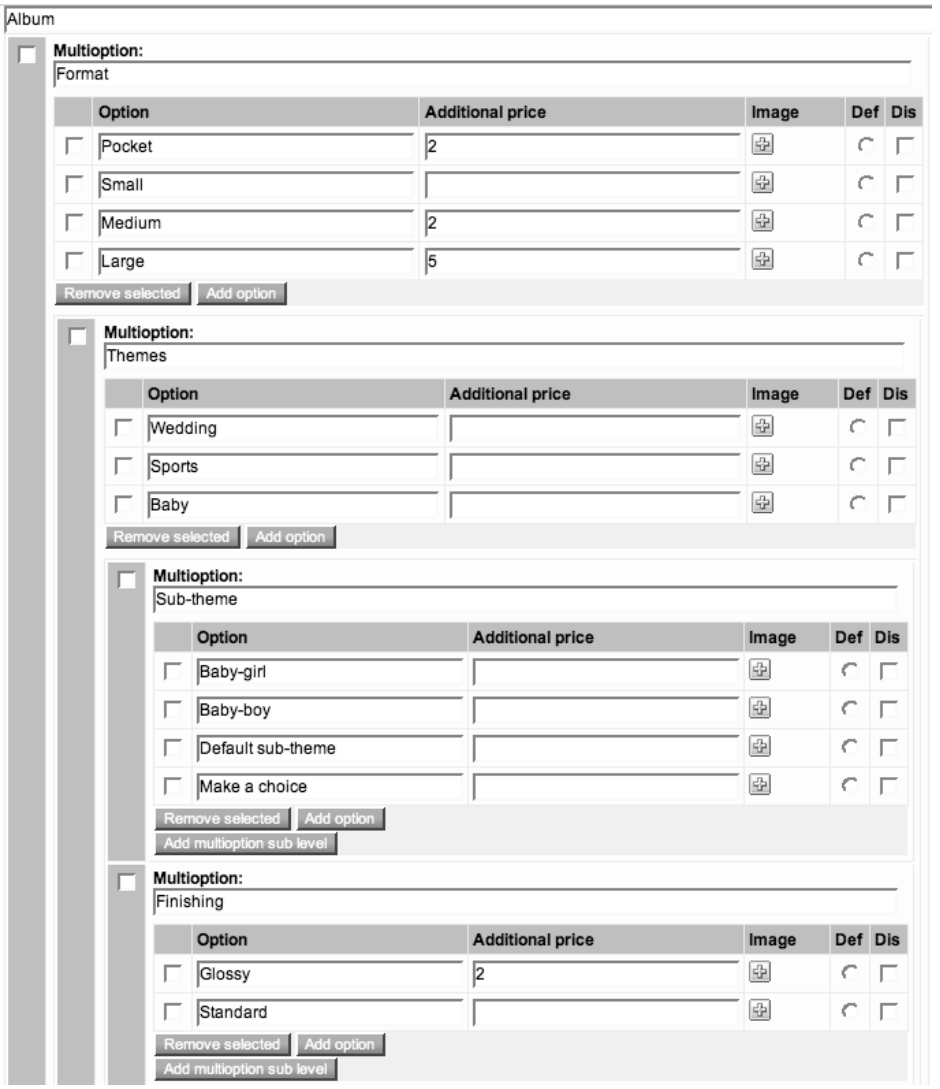


Figure 15.58. Structure and prices set

15.9.4. Default selection and disabled options

To the right of each option are the **Def** and **Dis** columns. The radio buttons of the **Def** column are used to optionally mark the *default selection* for a specific multi-option. For example, we could set the "Standard" option of the "Finishing" multi-option as the default. When customers are looking at the page for the scrapbook product on the front-end site, this option will be pre-selected in the corresponding dropdown list, although it can be changed.

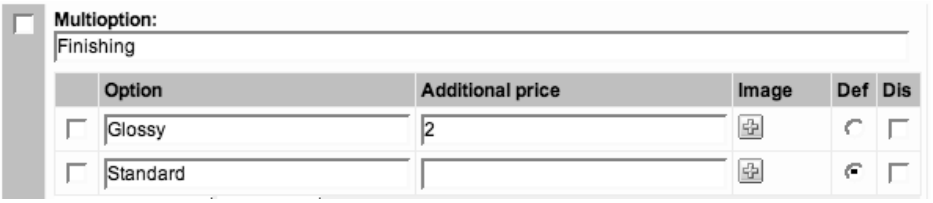


Figure 15.59. Default selection for finishing

Sometimes you want simply to ensure that customers do not omit important traits. In that case, marking an option as default sufficient. However, sometimes you want to force customers to make explicit choices. This is often used in combination with dependency rules (described later). For example, if the customer selects "Baby" as a theme, a sub-theme must be chosen, but favoring the boy or girl theme over the other as a default is not a good solution.

In this case it is better to add an additional dummy option "Make a choice" and set this as the default option. To ensure that the customer actually does make a choice, this option has to be disabled. A *disabled option* is an option that cannot be selected when the object is being viewed. When the default selection is disabled, it forces the customer to pick one of the other options.

Tip

You can disable several options of the same multi-option by marking the corresponding checkboxes. This is useful if, for example, some sizes or colors of your product are temporary sold out. To enable it again, simply unmark the checkbox.

15.9.5. Dependency rules for multi-options

A *dependency rule* is a rule specifying how multi-options at different sub-levels of the same group can be combined. For example, "Baby-boy" and "Baby-girl" sub-themes are only available with the "Baby" theme. Valid combinations are specified in a special-purpose editing interface accessed by clicking the **Rule** button at the bottom of the multi-option attribute editing area. This is a bit like toggling translator mode of the **Object Edit Interface**. As was shown in Figure 15.54, “Multi-option attribute - scrapbook example end result”, *rule mode* shows a multi-level table containing all the groups, multi-options and options, given hierarchical numbers, and checkboxes for marking valid combinations.

Initially all checkboxes are marked, meaning that all combinations are possible. Your job is to eliminate the invalid options by unmarking the checkboxes. For example, we unmarked the checkboxes for the "Baby-girl" and "Baby-boy" sub-themes for the "Wedding" and "Sports" themes, since they do not apply. Note that the disabled "Make

a choice" option is grayed out - you cannot specify dependency rules for disabled options. This is shown in the screenshot below, showing the dependency rules for the "Sub-theme" multi-option only:

Sub-theme

#	Option	2.1 -Wedding	2.2 -Sports	2.3 -Baby
3.1	Baby-girl	☐	☐	☑
3.2	Baby-boy	☐	☐	☑
3.3	Default sub-theme	☑	☑	☐
3.4	Make a choice	☑	☑	☑

Figure 15.60. Dependency rules for "Sub-theme" multi-option

We added the "Default sub-theme" option because otherwise, no option would have passed the input validation (described next) for the "Wedding" and "Sports" themes.

To turn off rule mode and return to standard edit mode, click the **Options** button at the bottom. Clicking the **Reset rules** button next to it will restore the initial configuration in which all combinations are selectable, except for those explicitly marked as disabled.

15.9.6. Multi-option validation

Recall that some datatypes support input validation. The dependency rules, default selection, and disabled markings are enforced by default when the customer customizes the product to purchase (by selecting items in dropdown lists), and when he or she tries to add the product to the basket or the wish list. This validation is handled by JavaScript, ensuring that incorrect customizations cannot be made. Notice in the screenshot below, which shows a part of the scrapbook product page on the front-end of the site, that the "Sports" theme is grayed out and cannot be selected when the "Large" format is chosen. This is due to the dependency rule that was specified.

Figure 15.61. Selection disabled due to dependency rule

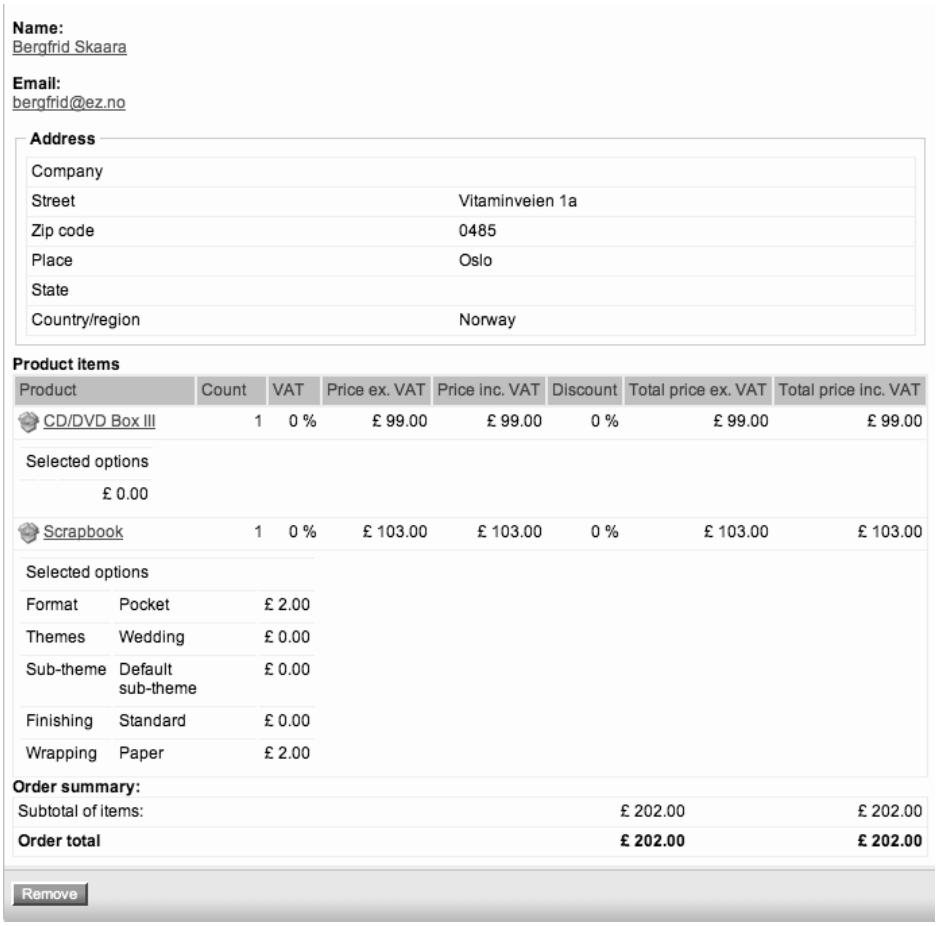
If the customer uses a browser without JavaScript support, validation is handled as part of checkout process. As a result, the customer might have to re-specify the options for

the product because an invalid combination was made, such as selecting glossy finishing for a sports album.

15.10. Managing orders

As explained in Section 15.1.6, “Orders”, Webshop orders are created when the checkout process of a purchase is completed. *Order management* includes Administration Interface tasks such as working with the **Order list** interface (accessed by clicking the **Orders** link in the left menu of the **Webshop** tab) and sales statistics. Billing, packing and dispatching are not part of order management as discussed in this book.

Recall that orders are stored using a special type of object. Each order is assigned a status. You cannot edit it, but you can change its status, view it, and archive it. The following sections will show you to do this, but let us first look at some of the information stored for an individual order. The screenshot below shows one of the orders in the **Order list** interface.



Name:
Bergfrid Skaara

Email:
bergfrid@ez.no

Address

Company	
Street	Vitaminveien 1a
Zip code	0485
Place	Oslo
State	
Country/region	Norway

Product items

Product	Count	VAT	Price ex. VAT	Price inc. VAT	Discount	Total price ex. VAT	Total price inc. VAT
CD/DVD Box III	1	0 %	£ 99.00	£ 99.00	0 %	£ 99.00	£ 99.00
Selected options £ 0.00							
Scrapbook	1	0 %	£ 103.00	£ 103.00	0 %	£ 103.00	£ 103.00

Selected options

Format	Pocket	£ 2.00
Themes	Wedding	£ 0.00
Sub-theme	Default sub-theme	£ 0.00
Finishing	Standard	£ 0.00
Wrapping	Paper	£ 2.00

Order summary:

Subtotal of items:	£ 202.00	£ 202.00
Order total	**£ 202.00**	**£ 202.00**

Remove

Figure 15.62. Single order

The top part contains some customer information, followed by information about the product(s) purchased. The bottom window shows the *status log* for this particular order. Customer information is collected as part of the checkout process, as described in Section 15.3.2, “Shop account”, and can also be viewed by selecting a customer in the **Customer list** interface (accessed by clicking the **Customers** link in the left menu of the **Webshop** tab). Products, prices, VAT, discounts and options were discussed earlier in this chapter.

Clicking the **Remove** button when viewing an order will delete that particular order.

15.10.1. Order statuses

An *order status* has a name (for example "Pending" or "Delivered") and identification number, as well as information about whether it is active or inactive. Websites with the

Website Interface version 1.2 and higher have three built-in order statuses: "Pending", "Processing" and "Delivered":

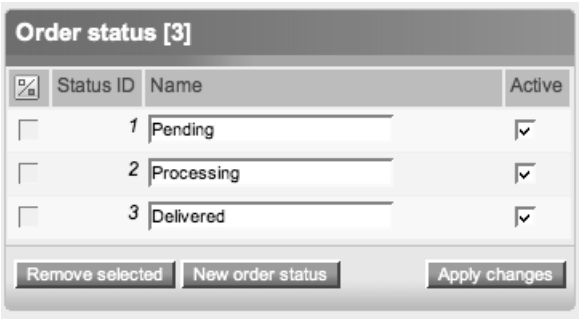


Figure 15.63. Order statuses

You can add, remove, rename, activate and deactivate order statuses in the interface in the screenshot above, which is reached by clicking the **Order status** link in the left menu of the **Webshop** tab. We recommend consulting your site administrator before making changes here if your site makes use of some form of shipping or billing system.

15.10.2. Placed orders

Placed orders are available in the **Order list** interface, accessed by clicking the **Orders** link in the left menu of the **Webshop** tab:

Figure 15.64. Order list interface

From within this interface you can:

- View a specific order by clicking on the order number
- View the customer information associated with the order by clicking on the customer's name

- Change the order status by selecting from the **Status** dropdown list, then clicking the **Apply changes** button
- Archive selected orders by marking the corresponding checkboxes, then clicking the **Archive selected** button
- Change the sorting type (by time or customer) and the sort order (ascending or descending)

15.10.3. Sales statistics

The **Product statistics** interface accessed from the left menu of the **Webshop** tab gives an overview of all sales:

Figure 15.65. Product statistics interface

Here you can quickly view which products have sold, and in what quantity. By comparing this list to the one in the **Products overview** interface, you can identify unpopular products. As times goes by, it might be handy to pull out monthly or yearly statistics. This is easily achieved by selecting from the dropdown lists and clicking the **Show** button.

15.11. Summary

The following list contains a summary of Webshop-related concepts:

- The **Webshop** tab offers special-purpose interfaces to view and manage your shop.
- A *product* is a content object with product information and a price. It is created, edited and stored in the node tree as other content objects and can be a simple-price or multi-price product.
- *Value Added Taxes* (VAT) affect the price of a product by a percentage. Prices are entered including or excluding VAT.
- *Discount rules* reduce the final price of products by given percentages.

- A *shop account handler* handles and stores customer information for webshop orders.
- *Wish lists* and *shopping baskets* are used by customers to indicate interest in a product or to purchase it.
- An *order* is created when the checkout process completes.
- Webshop-specific datatypes include the "Price", "Multi-price", "Range option", "Option", "Multi-option2", "Country" and "Product category" datatypes. A *range option* is single group of enumerated options without price differentiation. An *option* is a single group of named options with additional price information. A *multi-option* has multiple and distinctive groups of options with multiple levels.

Glossary

A

Access control	In general, the term "access control" refers to the practice of granting rights to access data or perform actions based on the configuration of a user account. The four access control components are policies, roles, users, and user groups. Without permissions, access to everything on a site is completely denied; it is only by the cumulative assignment of permissions that users are permitted to view content and use site functionality. See Also: Policy, Role, User account, User group.
Access control panel	The **Access control** panel is located at the bottom of the left menu of the **User accounts** tab, and contains a single **Roles and policies** link, which points to the **Role list** interface containing the **Roles** window. See Also: User accounts tab.
Administration Interface	The Administration Interface is the back-end interface used for providing powerful tools for content management and editing as well as site management, configuration, customization and development. It includes the **Object Edit Interface** for editing operations. It consists of six main components: the main menu, path, left / secondary menu, main area, right area and **Search** interface. See Also: Object Edit Interface, Siteaccess, Website Interface.
Administrator user	By default, a user belonging to the Administrator user group has unlimited access to manage the site and perform editorial tasks. The default username for the Administrator user is "admin" and the default password is "publish" (unless the user specifies a custom password during the installation process). See Also: User account.
Ajax	*Ajax* (Asynchronous JavaScript And XML) is a web development technique that uses JavaScript to further improve the user experience. For example, Ajax is used to enhance the performance of the dynamic tree menu in the Administration Interface. If your browser supports JavaScript, it also supports Ajax.
Always available	Some objects need to be displayed even if they do not exist in any of the languages that are configured for a siteaccess. Such objects include user objects and, depending on your site's needs, containers such as folders. The *always available* property can be toggled on and off for each object. See Also: Content object.

Anonymous user
: An Anonymous user is a site visitor that has not explicitly logged in. By default, Anonymous users can access the public siteaccess with read-only privileges.
See Also: User account.

Assigned roles window
: The **Assigned roles** window is associated with, and can be toggled on and off using the **Roles** switch of the **User accounts** tab. It lists the roles that are assigned to the current object, which is normally a user or user group. You can access the **Role editing** and **Role management** interfaces from this window by clicking on one of the role names or the corresponding **Edit** button.
See Also: Role, User accounts tab.

Attribute
: See: Content object attribute.
See: Content class attribute.

Audit trailing
: The system provides a set of audit functions that make it possible to generate audit logs for different types of activities. At a minimum, for every operation, the system logs when it happened (timestamp), where the request came from (IP address) and which user did it (username and ID number). Activation and configuration of the audit system is managed through overrides of the `audit.ini` configuration file.

Auto price
: Prices in the currencies for which you do not specify a custom price are called *auto prices*. These are calculated from the base price using the currency rates.

Available policies window
: The **Available policies** window is associated with, and can be toggled on and off using the **Policies** switch of the **User accounts** tab. The listing shows what role, module, function and limitation each policy regulates.
See Also: Policy, User accounts tab.

B

Base currency
: In order to obtain one unit of another currency, all currency rates in the system must be specified relative to the *base currency*. This makes it possible to automatically calculate auto prices for each product regardless of the currencies in which the base custom prices are specified.
See Also: Currency rate, Custom price.

Base price
: See: Custom price.

Binary file indexing
: The *binary file indexing* feature makes it possible to include contents of uploaded files like PDFs or word processor documents in the information searched by the default engine. This feature requires external handlers to

do the actual indexing. Such handlers provide keywords and / or convert binary files to plain text recognizable by the built-in search engine.
See Also: Search engine.

Blog

The term "blog" is short for "web log" and refers to a collection of posts displayed in descending order by date (reverse chronological order). *Blogs* provide visitors with a more personal and informal information channel from the organization and the ability to comment and interact on posts.

Bookmarks

The **Bookmarks** window contains a list of the current user's bookmarks, and is located in the right area of the Administration Interface. These bookmarks are specific to eZ Publish and should not be confused with the bookmark mechanism of your web browser. In eZ Publish, you can bookmark content nodes, but not special items like the cache management page or external pages. Using bookmarks is an alternative way of navigating content. These bookmarks can be managed through the **Bookmarks** interface, accessed from the **My account** tab.

Browse interface

The **Browse** interface makes it possible to select objects that have been placed somewhere in the node tree for various content management tasks. Such tasks include adding secondary locations, moving and copying content, and creating object relations.

C

Class edit interface

The **Class edit** interface is used mainly by web developers to modify and create content classes. In this interface, you can edit class information such as the class name, attributes and whether or not the class is a container.
See Also: Content class, Content class attribute.

Class parameter

The `class` parameter can assign a specific appearance or behavior to elements (such as headings, tables, and paragraphs) in "XML block" attributes. The value in this parameter is used by the templates and usually corresponds to a specific CSS class for styling.
See Also: Online Editor, CSS.

CMS

A *content management system* (such as eZ Publish) is used to organize, store, collect, manage and publish website content. The main goal of such a system is to provide a well-structured, automated yet flexible solution that allows information to be freely distributed and instantly updated across various communication channels (such as the internet, intranets and miscellaneous front- and back-end systems). The system's ability to handle the distinction between structuring and displaying content (the separation of content and design) is one of the key features of eZ Publish.
See Also: Content, Design, eZ Publish.

Collaboration system

The *collaboration system* makes it possible to force certain content to be approved by an editor before it is published. For example, you could have multiple editors submit material for final approval by an editor-in-chief, or you could make sure that all content created by site visitors (like comments, forum messages and blog posts) is checked before it is published.

Container

Some content objects are of container classes and their nodes can contain other nodes below them in the content node tree. Only objects encapsulated by nodes located directly beneath the container in the content hierarchy are displayed. For example, you can add Article objects below Folder objects in the content structure. By default, when viewing the folder, the list of articles is displayed. How the sub items are displayed is defined in the templates and settings for the container objects.
See Also: Content class, Content object, Content node tree, Template.

Content

The separation of content and design is one of the key features of eZ Publish. *Content* is information that is organized and stored in a structured manner by eZ Publish. For example, content may be the components of a news article (such as the title, introduction and body) or the properties of a car (such as the model, year and color). All information that is stored for the purpose of later retrieval is referred to as content.
See Also: Design, CMS.

Content class

A content type is called a *content class* (or "class" for short), while a specific piece of content is called a *content object* (or "object" for short). A content class can be thought of as a structural blueprint for a particular type of content. The properties of the class are referred to as *class attributes*.
See Also: Content object, Content class attribute, Datatype.

Content class attribute

The properties of a content class are referred to as class attributes. A datatype describes the type of value that can be stored in an attribute and is the smallest possible entity of storage. Each attribute has the same set of generic controls, regardless (but not independent) of the datatype that represents the attribute.
See Also: Content class, Content object, Datatype.

Content diff

See: Version history interface.

Content Editing Interface

The **Content Editing Interface** in the Website Interface has the same purpose as the **Object Edit Interface** in the Administration Interface: to edit content. When you are viewing content in the **Content Editing Interface**, you are in edit mode. You enter edit mode by clicking the **Edit** button on the **Website Toolbar** when viewing content.
See Also: Object Edit Interface, Website Toolbar, Online Editor.

Content hierarchy

See: Content node tree.

Content node

In eZ Publish, a content object is wrapped inside (or encapsulated) and structured using a *content node* (or "node" for short). Nodes are organized in a node tree. A node encapsulates a content object as a wrapper, and it provides a location for the content by adding structuring capabilities and binding it together with other content. A node has its own internal structure (ID, sorting of sub items, pointers to object / parent node, and more).
See Also: Content object, Content node tree, Content structure, Version.

Content node tree

The *content node tree* (or "node tree" for short) is a hierarchical tree structure built up of nodes. It is also known as the *content hierarchy*. The content node tree structures content hierarchically, including both visible and hidden nodes. An object may be assigned to several nodes, thereby appearing at multiple places in the content node tree. The tree is divided into five branches, of which the Content, Media and Users branches are the major branches.
See Also: Content node, Content object, Version.

Content object

In eZ Publish, a content type is called a *content class* (or "class" for short), while a specific piece of content is called a *content object* (or "object" for short). A content object contains actual data stored within its attributes. In object-oriented terms, an object is an instance of a certain class.
See Also: Content object attribute, Content node, Content class.

Content object attribute

A *content object attribute* (or "attribute" for short) holds the actual data for a content object. These are the values you enter in the input fields when creating or modifying an object. In object-oriented terms, a content object attribute is an instance of a content class attribute.
See Also: Content object, Datatype, Content class attribute.

Content relationship

A *content relationship* implies that there is some form of connection between different pieces of content. In eZ Publish, there are three ways of building content relationships. First, the content node tree ties all published content together in a content hierarchy. Second, object relations are relationships between objects; for example, images can be embedded in the body of an article. Third, external links connect objects to other webpages on the internet (internal links connect objects within eZ Publish and are a type of object relation).
See Also: Container, Embed, Object relation.

Content structure

A *content structure* can refer to a couple of things in eZ Publish. First, it refers to the way material on your website is organized. Second, it refers to one of the tabs in the Administration Interface - the **Content structure** tab. This tab is associated with the Content branch of the node tree; the Content branch is a tree-structure representation of most of the content that site visitors see.
See Also: Section, Content node tree.

Context sensitivity

Context sensitivity means that the buttons and choices shown depend on the content currently displayed and the operations available under the given circumstances.

Context-sensitive pop-up menu

The *context-sensitive pop-up menu* (or "pop-up menu") contains functions that are specific to the item from which the menu was invoked, providing quick access to commonly used functions. The pop-up menu can be brought up by clicking one of the icons to the left of a node (in the tree menu or **Sub items** window), provided that JavaScript is supported and enabled in your browser.
See Also: JavaScript.

Country (datatype)

Attributes of the "Country" datatype store a country, and enable you to use country-dependent VAT as your VAT charging system. The available countries are defined in the `country.ini` configuration file, with a standards-compliant name (such as "Norway"), code (such as "NO") and International Dialing Code (such as "47").

Cronjob

In eZ Publish, a *cronjob* refers to maintenance tasks that can be run periodically by the `runcronjobs.php` script. The usual procedure is for a system administrator to specify when `runcronjobs.php` should be run, then the operating system will process the job in the background at the scheduled intervals or times. On UNIX / Linux systems, this can be done by making use of the "cron" system. On Windows, the script can be run by the "Scheduled Tasks" service.
See Also: Workflow system.

CSS

A *Cascading Style Sheet* (CSS) is a construct that is most commonly used with HTML and XHTML to provide more control over page display. CSS allows designers to create stylesheets that define how different elements (such as headers, links and images) should appear. The stylesheets can then be applied to any webpage. Modifying a stylesheet changes the display for all webpages that use the stylesheet.
See Also: Design, Class parameter.

Currency rate

A *currency exchange rate* (or "currency rate" for short) specifies the amount of some currency that must be given up in order to obtain one unit of the base currency.
See Also: Base currency.

Current draft window

The **Current draft** window is a part of the **Object Edit Interface** and displays information about the version that is currently being edited.
See Also: Object Edit Interface.

Custom price

The *base custom price* (or "base price" for short) is the price of a multi-price product on which the auto prices are based. Additional custom prices

(set by content editors) are called *non-base custom prices*. Non-base custom prices are fully independent of the currency rates and the base price.

D

Datatype

A *datatype* is the smallest entity of storage in eZ Publish and determines the validation, storage and retrieval for the value held by the attribute.
See Also: Content object attribute, Input validation.

Default language

The default language corresponds to the "ContentObjectLocale" setting in the `site.ini` configuration file. The default language is used in one aspect of the language fallback system, as well as by developers. What content managers might describe as the "default" language is more likely either the highest prioritized site language for a siteaccess, or the main or initial language of some content object.
See Also: Language fallback system.

Delayed publishing

Normally, a draft version transforms into a published version directly when you submit it. The *delayed publishing* mechanism enables content to be automatically published at a specified time in the future. Using this feature, the draft is given the "Pending" status when submitted, and remains so until it is published at the specified time.

Design

The separation of content and design is one of the key features of eZ Publish. Design deals with the visual presentation of content, data markup and layout. It includes the things that make up a web interface: HTML, stylesheets, images that are not part of the content, typography, fonts, and so on.
See Also: Content, CMS, CSS, HTML, Template.

Design tab

The **Design** tab of the Administration Interface displays an area used to view and modify the visual elements and layout of the site.
See Also: Administration Interface.

Details window

The **Details** window is shown in the main area of the Administration Interface when viewing some content (and when the corresponding switch is enabled). This displays additional information about the selected node and the object that it encapsulates.
See Also: Main area.

Discount rule

A *discount rule* reduces the final price of a product by a given percentage.

DNS

A *Domain Name System* (or Service or Server) is an internet-based service that translates domain names (such as `www.example.com`) into IP addresses (such as "60.70.12.130"). While humans prefer to work with do-

main names, computers work with IP addresses; the DNS provides an interface between the two.

Draft

A *draft* is a newly created but not yet published piece of content stored in the system. Drafts are managed through the **User profile** page of the Website Interface or the **My account** page of the Administration Interface.
See Also: Version, Version status, Content object.

E

E-commerce

See: Webshop.

Editing conflicts

Editing conflicts happen when several editors are editing the same piece of content, making eZ Publish uncertain of which version of the content you want to edit. The three main scenarios that create an editing conflict are "Unpublished draft - same user", "Unpublished draft - different user", and "Unpublished draft - multiple users including you". Preventative measures include always exiting edit mode properly and regularly managing your drafts.
See Also: Edit mode, Draft.

Editor user

By default, a user belonging to the Editor user group has access to add, edit, move and remove most of the content on the site.
See Also: User account.

Edit mode

When you are viewing some content in the **Content Editing Interface** (in the Website Interface) or the **Object Edit Interface** (in the Administration Interface) you are in *edit mode*. You usually enter edit mode by clicking either the **Edit** or **Create here** button.
See Also: Online Editor.

Embed

Content objects can be displayed within other content through the embedding feature. For example, you can insert images into the rich text "XML block" attributes (such as article bodies).
See Also: Rich text content, Edit mode.

Encapsulation

All content objects are usually encapsulated by (or wrapped inside) one or more nodes.
See Also: Content node.

Extension

An eZ Publish extension is an add-on that provides extra functionality to the system. Examples include the Online Editor, the eZ Find search engine, and eZ Flow.

eZ Find

eZ Find is an extension that enhances the search feature by providing, among other things, more search term options and relevancy ranking of results.

eZ Flow
: *eZ Flow* is an eZ Publish extension aimed at media sites with rapid and frequent content changes. In eZ Flow, a *layout* is a combination of *zones* placed on a frontpage. A *block* is an area within a zone that contains a particular type of content. The **eZ Flow** interface is a special-purpose edit mode look for managing frontpages.

eZ Network
: *eZ Network* is an automatic maintenance service for eZ Publish.
See Also: eZ Publish.

eZ Publish
: eZ Publish is a highly flexible and customizable content management system. It can be used to build everything from personal home pages to multi-national corporate web solutions with role-based multi-user access, online shopping, discussion forums and other advanced functions. Based on open source technologies and principles, it can be easily extended and customized to interact with other solutions.
See Also: CMS.

eZ Systems
: eZ Systems is the creator of eZ Publish. Based on a philosophy of openness and information sharing, eZ Systems combines enterprise software with open source code and Total Product Responsibility.
See Also: eZ Publish.

F

Formatted text
: Formatting can be as simple as marking a word bold or italic, or using more advanced elements such as tables, lists, inline graphics, links and so on. This kind of content is often referred to as *rich text*. Formatted text within "XML block" attributes is edited through the Online Editor or manually with XML tags.

Front-end editing
: Content can be edited from the website's front-end through the Website Interface and its **Website Toolbar** (when logged in with an Editor or Administrator user account).

G

Gallery
: Objects of the `Gallery` content class are containers that are usually used to organize images.
See Also: Content class, Container.

Global content translation list
: The *global content translation list* corresponds to the entire set of translation languages (the set of languages in which you can create and translate content on a site).
See Also: Translation languages.

Group
: See: User group.

H

Help — Usage assistance is available via mouse hover tooltips and through the **Help** buttons on the **Website Toolbar** and **OE toolbar**.

HTML — *HyperText Markup Language* is the language designed for the creation of webpages and other information viewable in a browser. HTML is typically used to structure information, such as headings, paragraphs, lists and images. This is done with tags such as `<p>` and `<li>`.

HTTP — *Hypertext Transfer Protocol* is the primary method used to transfer data between web servers and browsers.

I

Information collection — The *information collection* feature makes it possible to gather user input on a site. This means that some part of a specific page is set up to gather data. This feature is typically useful when it comes to the creation of feedback forms, surveys and polls.

Initial language — See: Main language.

Input validation — *Input validation* is supported by most (but not all) of the built-in datatypes. The validation feature of a datatype cannot be turned on or off. If a datatype supports validation, it will always try to validate the incoming data and the system will never allow the storage of incorrectly formatted input. If there are any validation errors, you will be informed when attempting to store or publish a draft.
See Also: Datatype.

Interactive tree — The left menu of the Administration Interface, shown when accessing the **Content structure**, **Media library** or **User accounts** tabs, behaves in a similar way as the interface of a typical file browser in a modern operating system. It enables you to view and explore the depths of the tree by expanding the different nodes. A node is selected by clicking its name. Whenever a different node is selected, the interface will reload and the main area will display the selected node. This behavior is in contrast to expanding and collapsing the branches, which does not reload the main area.
See Also: Content node tree, Tree menu.

J

JavaScript

JavaScript is a lightweight client-side programming language that extends the functionality of HTML. It is used to create a more user-friendly environment in eZ Publish. Among other things, JavaScript is used to create an interactive tree-representation of content in the Administration Interface (in the tree menu), enables the usage of the pop-up menu both in the Website Interface and Administration Interface, and is involved in important features of the Online Editor.

L

Language fallback system

The integrated *language fallback system* handles language filtering and prioritizing for multilingual sites in order to ensure that requested content is displayed appropriately. The language fallback system is used every time a content object is requested, to select one of the object's translations to display. Filtering is done based on the list of site languages for a particular siteaccess. Filter overrides applied as a safety net for cases when important content objects do not exist in any of those languages. Siteaccesses can be configured to display untranslated content, and objects can be set to be always available. In both cases, the object's main language is used.
See Also: Translation, Untranslated content, Site languages.

Language management interface

The **Language management** interface is located within the **Setup** tab and accessed by clicking the **Languages** link in the left menu. In this interface, you can view (but not edit) the specific settings of a locale by clicking on the corresponding language name. You can also add and remove translation languages.
See Also: Translation languages.

Language selection interface

The *Language selection interface* is presented whenever you create or edit an object. When adding a new translation, it is possible to choose an existing translation on which it will be based. You can select one of the existing languages or choose "None". When a language is chosen, edit mode will be transformed into a special translator mode.
See Also: Translator mode, Translation.

LDAP

Lightweight Directory Access Protocol is a relatively simple protocol for updating and searching directories running over TCP/IP.

Left menu

See: Secondary menu.

Locale

A *locale* is a set of country-specific settings like language, character set, number format, currency format, and date and time format. These are

defined in configuration files, and have standards-compliant identifiers like "eng-US" (for English, USA).

The *site locale* refers to the value of the "Locale" setting within the "[RegionalSettings]" block of `site.ini`. The specified locale will affect how some content is displayed on the site. It is common to use different site locales for different language-specific siteaccesses.
See Also: Site.ini.

Location — A node provides a location for a content object within the tree structure. A node refers to exactly one content object.
See Also: Content node, Content node tree, Content object.

Locations window — The **Locations** window is shown in the main area of the Administration Interface while viewing content (and when the corresponding switch is enabled). The purpose of this window is to reveal the different nodes that encapsulate the current object.
See Also: Main area.

M

Main area — The *main area* is the most dynamic and important part of the Administration Interface, as it displays the actual content and / or interfaces that are associated with the current action. For the three left-most tabs, it includes a row of horizontally-aligned switches for toggling the display of various windows when viewing content.
See Also: Administration Interface.

Main edit window — The **Main edit** window in the **Object Edit Interface** is where you modify the attributes of the current object.
See Also: Object Edit Interface.

Main language — An object can be created in any of the languages found in the list of translation languages. When an object is created, its *main language* is set to the language that was used during creation. When you create an object, its initial and main language are the same, but you can change the main language later if the object contains multiple translations.
See Also: Translation, Translation languages.

Main menu — The *main menu* of the Administration Interface is a horizontal group of tabs located below the eZ Publish logo and the **Search** interface.
See Also: Administration Interface.

Media library — The **Media library** tab is one of the tabs in the Administration Interface. It is associated with the Media branch (also called the *media library*) and section of the node tree. The media library is used to store media content

(including images, animations, movies and documents) outside the Content branch. The media library can be considered a repository for often re-used material and material uploaded to the system through the Online Editor.
See Also: Content structure, Section, Administration Interface.

MIME type

Multipurpose Internet Mail Extensions (MIME) is an internet standard that defines content types. Its purpose is to extend the format of emails. It is common to associate a MIME type with a file extension such as `.jpg`.

Modal dialog

When some **OE toolbar** buttons or context-sensitive pop-up menu items are clicked, modal dialogs / windows are shown. A *modal dialog* provides access to additional editing tools depending on the current setting.
See Also: Online Editor, Context-sensitive pop-up menu.

Module

A *module* offers an interface for web-based interaction, providing access to eZ Publish core functionality. The most important and most utilized module is the "content" module. A module has a set of *views* that enable you to reach the functions provided by the module. In other words, a view represents an interface to a module. For example, the "edit" view of the "content" module lets you modify the contents of an object.
See Also: View.

MTA

A *Mail Transfer Agent* is an application that sends and receives emails to and from the system on which it is installed.

Multi-price

The "Multi-price" datatype stores multiple prices in different currencies per product, including or excluding VAT.

Multilingual site

eZ Publish sites with more than one language installed and can thus have content in multiple languages are referred to as *multilingual sites*.

My account tab

The **My account** tab in the Administration Interface provides access to user-specific settings (such as the user profile and the **Notification settings** interface); a user's drafts and pending items; the **Collaboration** interface; and the Webshop basket and wish list.
See Also: Administration Interface, User profile, Draft.

N

Node

See: Content node.

Non-translatable attributes

Non-translatable attributes are marked as such in the class definition. When creating additional translations (beyond the main translation) the value in a non-translatable attribute is copied to the new translation and cannot be edited. This is useful for attributes such as email addresses, dates and numbers.
See Also: Content class.

Node visibility
: *Node visibility* refers to whether or not some published content can be viewed on the front-end of a site. The effect of hiding a node is that the node in question, and all of its descendants, are no longer displayed.

Notification
: The email notification service sends an email to subscribed users every time a specific part of the website (as specified by each user) is changed. Users can also sign up for notifications about actions in the collaboration system. Notifications can be managed through the **My account** tab in the Administration Interface and the **User profile** page in the Website Interface.
See Also: Collaboration system, Content node tree, Notification settings interface.

Notification settings interface
: The **Notification settings** interface is used to modify three aspects of the notifications feature: the frequency of notifications; whether or not notifications are sent individually or combined into one digest; and to remove the notifications setting from pages for which you previously requested notifications.
See Also: Notification.

O

OASIS
: The *Organization for the Advancement of Structured Information Standards* is a not-for-profit, international consortium that drives the development, convergence, and adoption of e-business standards such as the Open Document Format for Office Applications.

Object
: See: Content object.

Object Edit Interface
: The **Object Edit Interface** is a special-purpose interface within the Administration Interface where content is created and edited.
See Also: Content Editing Interface, Administration Interface.

Object information window
: The **Object information** window is a part of the **Object Edit Interface** and reveals information about the object that is being edited. It provides access to manage the object's versions.
See Also: Object Edit Interface.

Object-oriented technology
: The eZ content model is based on ideas borrowed from object-oriented programming languages like Java and C++. Superficially, *object orientation* means to look at the world in terms of objects. A data structure is described using a content class made up of attributes. Content is managed through objects, which are instances of classes.
See Also: Content object, Content class, Attribute, Datatype.

Object relation

An *object relation* is an association between two content objects. Relations can be specified within XML blocks via the `embed` and `link` tags. Some content types also allow relations to be specified as attributes to content objects, and you can also add object relations of the Common type for unspecified purposes. The referred object is called a *related object*.
See Also: Related object.

OE status bar

The **OE status bar** is part of the Online Editor and located directly below the **OE text area**. It displays useful information about the element where the cursor is currently located.
See Also: Online Editor.

OE text area

The **OE text area** is part of the Online Editor and is an editable multi-line text area showing a visualization of your content as formatted text.
See Also: Online Editor.

OE toolbar

The **OE toolbar** is part of the Online Editor and contains tools for editing the content that is displayed in the **OE text area**. Its buttons are very similar to those found in many word processors.
See Also: Online Editor.

Online Editor

The Online Editor simplifies the editing process by providing assisted text editing with a WYSIWYG interface. In edit mode, you can use the Online Editor to apply formatting characteristics in the text fields for "XML block" attributes. You can also create links, insert tables and lists and more.

The Online Editor is an extension to eZ Publish that is included and enabled by default, meaning that it is automatically displayed when editing attributes of the "XML block" datatype.
See Also: Datatype, XML, CSS, Class parameter, Content Editing Interface, Object Edit Interface, Edit mode, Administration Interface.

Open Document Format

Open Document Format is an XML-based file format specification by OASIS for office applications.
See Also: OASIS.

OpenOffice.org

OpenOffice.org is a multi-platform and multilingual office suite and an open source project. Compatible with all other major office suites, the product is free to download, use and distribute.

eZ Publish can import and export text documents based on the OASIS Open Document Format (ODF) via the eZODF extension, which is included and enabled by default. This is the format used by OpenOffice.org Writer documents.

Open source

Open source refers to computer software that has publicly available source code (as opposed to proprietary programs where the source code is not

available). Open source software is distributed under licenses that permit modification and re-distribution.

Options — Attributes of the "Option", "Multi-option2" and "Range option" datatypes are used to enable customers to customize the product being purchased. For example, they can select a shoe size, t-shirt color, brand, or whether or not to add gift wrapping. The "Option" and "Multi-option2" datatypes make it possible to specify additional prices for particular types of products (such as an extra $5 for the large size). As a result, you can have different prices for one and the same product in the same currency.

Order — An *order* is created when the checkout process from the shopping basket is completed. Orders are stored in the database as special-purpose objects.

P

Pagelayout — The main template is called the *pagelayout*. It contains the `<html>`, `<head>` and `<body>` tags; together with the stylesheets (CSS), these dictate the overall look of a site. Among other things, the pagelayout specifies the title, logo, menu and footer that is presented for each webpage the system generates.
See Also: Template.

Path — The interactive *path* in the Administration Interface is located below the tabs. It is a representation of your current position in the navigation hierarchy of the site (in the Content, Media and Users branches), or the module or view used. For the three left-most tabs, the last path element shows the name of the content you are currently viewing, prefixed by links to the higher-level content. The path is also known as *breadcrumbs*.
See Also: Administration Interface.

PDF — *Portable Document Format* is Adobe Systems' widely used file format that allows the electronic representation of documents in a paged form. It is supported on all major computer platforms.

PHP — *PHP: Hypertext Preprocessor* is an open source, server-side HTML-centric scripting language that can be used to create dynamic webpages.
See Also: HTML.

Plain text content — Attributes of the "Text block" and "Text line" datatypes are shown as is, since these datatypes do not permit formatting to be specified inside the text. Therefore, this kind of content is often referred to as *plain text*.
See Also: XML, Datatype, Rich text content.

Policy

A *policy* is a single rule that grants access to specific or all functionality of a module. It is part of the access control mechanism of eZ Publish. A role is a set of policies that is assigned users and user groups.

A *limitation*, or more precisely a module function limitation, restricts the domain a policy regulates. Limitations dictate under which circumstances a policy applies, for example, to allow blog posts and comments to be made only within the "Community" section.
See Also: Access control, Role.

Pop-up menu

See: Context-sensitive pop-up menu.

Preview window

The **Preview** window is shown at the top of the main area of the Administration Interface (below the horizontally-aligned row of switches) when viewing some content object. This window displays the contents of the object's attributes.
See Also: Main area, Administration Interface.

Price

The "Price" datatype stores a single price per product, including or excluding VAT. A multi-price product will always have as many prices as there are currencies.

Product

A *product* is content object with a price. The price is represented by an attribute of the "Price" or "Multi-price" datatype. Depending on which of these datatypes is used, the product is referred to as a *simple-price product* or *multi-price product*.

Product category

Attributes of the "Product category" datatype store product categories to be used by the VAT charging system.
See Also: Datatype.

Publish

To publish a content object is the transition from the object having the "Draft" status to having the "Published" status. This happens when you click the **Send for publishing** button for the first time in an object's life cycle.

To publish a version means to make changes to a content object publicly available, except if it is the first version of the object. In that case, publishing the version is equivalent to publishing the object.

R

Related object

Content objects can have references to other objects, called *related objects*. Related objects are listed in the **Related objects** window (at the bottom of the **Object Edit Interface**) and the **Relations** window (near the bottom of the main area of the **Content structure** tab in the Administration Interface).

	See Also: Content object, Object relation, Embed.
Relations window	The **Relations** window is shown in the main area of the Administration Interface when viewing some content (and when the corresponding switch is enabled). The purpose of this window is to reveal information about objects that either use or are used by the current object. See Also: Main area.
Revert	Reverting content means to go back to (and publish) an older version of some content object. This is useful if you accidentally published a draft (instead of storing or discarding it) or if the published version is no longer valid. See Also: Undo.
Rich text content	In contrast to the plain text content you get from the "Text block" and "Text line" datatypes, the "XML block" datatype enables you to add formatting. Formatting can be as simple as marking a word to be shown in bold or italics, or more advanced like tables, lists, inline graphics and links. This kind of content is often referred to as *rich text*. See Also: XML, Datatype, Plain text content.
Right area	The *right area* of the Administration Interface displays information specific to the user who is currently logged in. Among other things, this is where you find the **Bookmarks** window, which contains a list of the current user's bookmarks. See Also: Bookmarks, Administration Interface.
Role	A *role* is a component of the access control mechanism, and is a group of policies that apply concrete access rules to specific users and user groups. See Also: Access control, Policy.
Role editing interface	The **Role editing** interface is a special-purpose editing interface for roles. It is accessed by clicking the **Edit** button in the **Role management** interface, the **Role list** interface, or the **Assigned roles** window. See Also: Policy, Role, Role list interface.
Role list interface	The **Role list interface** of the **User accounts** tab is accessed by clicking the **Roles and policies** link in the **Access control** panel at the bottom of the left menu. It contains the **Roles** window, which displays a list of existing roles. From here you can add, remove, modify and view roles. See Also: Access control panel.
Role management interface	The **Role management** interface is a special-purpose interface within the **User accounts** tab. The top window is a simplified version of the **Preview** window normally shown for content objects, instead showing the details of the current role. The bottom window is for assigning a role to users and user groups.

See Also: Role editing interface.

RSS

Really Simple Syndication (RSS) is a standard for information importing and exporting, commonly applied for news updates. An *RSS feed* is an XML file representing content from a particular part of a site. By including your feed, other sites can display content from your site on their site. When the contents of an RSS feed change (when, for example, someone creates or edits a content object), the sites using the feed are updated. You can do the same on your site by importing RSS feeds from other sites. Also, with RSS readers, site visitors can get automatic updates whenever content on an RSS feed has been updated.

S

Search engine

A *search engine* is a system for information retrieval on a computer system. eZ Publish has a built-in search engine. There is also an enhanced search engine extension called eZ Find.

Search interface

The **Search** interface is located in the top right corner of all eZ Publish sites by default. The default behavior is that the system will search for the specified word(s) within the entire node tree. You can limit a search by using the **Current location** radio button (only within the Administration Interface) or by specifying advanced search criteria.

Secondary menu

The *secondary menu* (also called the "left menu") of the Administration Interface provides access to content and interfaces that are associated with the tab that was selected from the main menu. If your browser has JavaScript support enabled, the secondary menu of the three left-most tabs is shown as an interactive tree.
See Also: JavaScript, Administration Interface.

Section

A *section* is a virtual collection of nodes that belong together, either conceptually or functionally. A *section ID* is an identification number that can be assigned to an object and denotes which section that object belongs to. A section can be assigned to many objects, but a single object can have only one section. Using sections makes it possible to, among other things: segment the node tree into different subtrees; set up custom template override rules; limit and control access to content; and assign discount rules to a group of products.

Sections interface

The **Sections** interface is a tool for managing sections. It is accessed from the **Setup** tab by clicking the **Sections** link in the left menu.

Send for publishing button

When the **Send for publishing** button is clicked, the system attempts to validate the contents of the attributes (if their datatypes support validation)

	and publish the draft. The draft will then become the published version of the object, and will, by default, be viewable on the site. See Also: Draft.
Session	A *session* is a user- and application-specific set of connections between a client and a server. It is used to, among other things, ensure that a site visitor only has to log in once when browsing restricted content.
Single-language site	eZ Publish sites with content in only one language are referred to as *single-language sites*.
Site	A *site* (or "website") encapsulates and includes everything that belongs to a particular area on the web (for example http://ez.no). An eZ Publish site includes configuration settings, a database with content structure and actual content, content-related files and design-related files. See Also: HTML.
Siteaccess	A *siteaccess* is a collection of (override) configuration settings. The Administration Interface corresponds to the *admin* siteaccess, and the public front-end interface (possibly containing the Website Interface) corresponds to the *public* siteaccess. There is a one-to-one relationship between a *site interface* and a siteaccess. See Also: Site, Site interface, Administration Interface, Website Interface.
Site.ini	The main configuration file in eZ Publish is *site.ini*. Here, and in the overrides of it, developers and administrators configure important information, including which database and design to use; how your site may be accessed; the name and path to the directories your installation uses; available extensions; and various configuration about templates, users, caching, siteaccesses and so on.
Site interface	A *site interface* refers to the visual presentation of a site, and is recognized through the address in the browser. There is a one-to-one relationship between a site interface and a *siteaccess*. See Also: Siteaccess, Administration Interface, Website Interface.
Sitemap	An eZ Publish *sitemap* is a two-level structural guide to your contents, or more specifically, a visual representation of the node tree structure. A sitemap is similar to a table of contents in a book.
Site languages	The list of *site languages* is a set specifying the languages in which the contents of a site should be displayed. It can be controlled per siteaccess based on a prioritized list in the "SiteLanguageList" setting within the "[RegionalSettings]" block of the `site.ini` configuration file. See Also: Translation, Siteaccess.

Site settings

Site settings are only available to Administrator users, and can be accessed via the **Site settings** link in the top right corner. From there, you can edit information such as the company logo, the title of the site, and more.
See Also: Website Interface, Administrator user.

SMTP

Simple Mail Transfer Protocol (SMTP) is a protocol used to send email messages between computers.

SOAP

SOAP is an XML-based, lightweight protocol for exchanging information.

Solr

The enterprise search server *Solr* (http://lucene.apache.org/solr/) is an open source project based on the Apache Lucene Java search library. This library is what enables the new and enhanced functionality of eZ Find.

Sort

By default, content objects are automatically sorted in the order that they were published, with the newest material at the top. Sorting is managed through the **Sorting** controls in the **Sub items** window of the Administration Interface.
See Also: Sub items window.

Sub items window

The **Sub items** window is shown at the bottom of the main area of the Administration Interface when viewing some content. This window displays a node's children (if there are any).
See Also: Main area, Administration Interface.

Switches

The horizontally-aligned *switches* at the top of the main area of the Administration Interface determine which windows (also called *panels*) are shown in the main area. A blue background indicates that a switch is enabled and thus the window containing the related information is currently being displayed. The **Content structure** tab has the **Preview**, **Details**, **Translations**, **Locations** and **Relations** switches.
See Also: Administration Interface.

T

Tab

The main menu of the Administration Interface is a collection of horizontally-aligned *tabs*. A tab is basically a menu item. Each tab provides access to content and / or interfaces and represents a group of administration functions, such as user or Webshop management. The main menu contains the **Content structure**, **Media library**, **User accounts**, **Webshop**, **Design**, **Setup** and **My account** tabs.
See Also: Administration Interface.

Tag cloud

Tags represent categories, or keywords, that relate to specific content objects. This helps to create interfaces where a user can quickly browse other pages with related content (for example "see also" or "related pages"

list of hyperlinks, and tag clouds). *Tag clouds* are a quick way for visitors to get a visual idea of the most popular topics on a site, and to quickly focus on topics of interest.

Template — An eZ Publish *template* is a custom HTML file that describes how some particular type of content should be visualized. It is also used for information extraction. The template system is component-based and the fundamental unit of site design. The main template is called the *pagelayout*.
See Also: Design, CSS, HTML, Pagelayout.

Translate from window — The **Translate from** window is part of the **Object Edit Interface** and displays information about the existing languages. It also enables you to select the language on which a new translation should be based.
See Also: Object Edit Interface, Translation.

Translation — A *translation* is a representation of the information in a specific language. The one-to-one translation solution makes it possible to represent the same content in multiple languages. This mechanism is completely independent of the datatypes. Each version consists of at least one translation. Translations are also known as the third dimension of content objects.
See Also: Content object, Version, Translation languages.

Translation languages — The list of *translation languages* is a set of languages in which content can be created and translated. The number of languages that can be used simultaneously is 30. You can add and remove languages within this limit as required.
See Also: Translation.

Translations window — The **Translations** window is shown in the main area of the Administration Interface when viewing some content (and when the corresponding switch is enabled). The purpose of this window is to show the languages in which the published version of an object exists. You can access translation-related operations from this window.
See Also: Main area, Administration Interface.

Translator mode — *Translator mode* is a special-purpose sub-mode of edit mode used for working with multilingual content and translations. If you select a language using the radio buttons located in the **Translate from** window and click the **Translate** button, the **Main edit** window is switched to a special translator mode with the source text from the selected existing language displayed above the relevant input fields.
See Also: Translate from window, Object Edit Interface, Translation, Edit mode, Administration Interface.

Trash — By default, deleted content is moved to a trash container, and is not permanently deleted from the system. You can access the trash through the

corresponding link at the bottom of the left menu when viewing content in the Administration Interface.
See Also: Undo, Secondary menu, Administration Interface.

Tree menu

The secondary / left menus of the first three tabs (the **Content structure**, **Media library** and **User accounts** tabs) look and behave in the same way, and are also referred to as *tree menus*. These menus provide access to the parts of the node tree.
See Also: Secondary menu, Administration Interface.

Trigger

A *trigger* initiates a workflow. It is associated with a function of a module, and starts the workflow before or after the function is completed.
See Also: Workflow, Workflow system.

U

Undo

Some content management tasks can be undone. Deleted items can be recovered from the trash, provided that they are still in the trash. Unwanted changes to content objects can be reverted by copying and re-publishing a previous version. Some text editing and formatting in the Online Editor can be undone through the integrated undo feature.
See Also: Trash, Revert, Online Editor.

Unpublish

The term "unpublish" is commonly used to describe taking an object off the public site. This is accomplished by deleting the content object, hiding it (altering its visibility status) or moving it to a restricted section. We recommend that you use the terms "delete", "remove", "hide" or "move" rather than "unpublish".
See Also: Content object, Publish, Section.

Untranslated content

Untranslated content is a content object that does not have a translation in any of the site languages for a siteaccess, and is excluded, by default, by the language fallback system. By enabling the "ShowUntranslatedObjects" setting in the "[RegionalSettings]" block of `site.ini`, untranslated content is shown in the object's main language.
See Also: Language fallback system, Site languages, Main language.

URL

A *Uniform Resource Locator* is a way of specifying the location of something on the internet, for example `http://www.example.com/hello.html`.

In eZ Publish, two types of URLs co-exist. System URLs are the raw identifiers used internally and normally not displayed to site visitors when viewing content. Virtual URLs are more user friendly and represent simplified versions of system URLs.

User account
: A *user account* is a content object that represents a registered user on the system and is associated with a *username* and a *password*. It provides access to privileged areas and operations.

User accounts tab
: The **User accounts** tab enables you to browse and manage nodes within the Users branch of the content node tree. This branch is associated with the Users section.

User group
: Registered users in the system belong to *user groups*. This concept is part of the access control mechanism and simplifies the task of managing users and assigning common privileges to multiple users.
See Also: Access control, User account.

User profile
: A *user profile* denotes the information stored in the actual User object, such as first and last name, image and signature. In other words, editing your user profile means to edit the contents of the object holding your user account.
See Also: User account.

UTF-8
: *UTF-8* is an encoding scheme for storing Unicode code points in terms of 8-bit bytes. Characters are encoded using sequences of 1, 2, 3 or 4 bytes. Characters in the ASCII character set are all represented using a single byte.

V

Value Added Taxes
: *Value Added Taxes* (VAT) affect the price of a product by a percentage. A price can be entered including or excluding VAT. If entered excluding VAT, VAT will be calculated and added to the displayed price, whereas if entered including VAT, the price stored in the object is the one displayed.

Version
: A content object consists of one or more *versions*. Versions are also known as the second dimension of content objects. Every time you, or someone else with editing permissions, click either the **Create here** or **Edit** buttons for some content, a new version is automatically created by the system. It is always the new version that will be edited; old versions remain untouched. A version includes a version number used to identify it and a version status (draft, published, archived, pending or rejected).

  Each version contains at least one translation.
  See Also: Version number, Version status, Translation, Content object.

Version history interface
: The **Version history** interface makes it possible to view and manage the versions that belong to the object that is being edited. It can be accessed by clicking the **Manage versions** button in the **Object information** win-

dow. Via this interface you can remove, copy and edit versions, as well as compare the differences between versions.
See Also: Version, Object information window.

Version number

Every version has a *version number*. This number is used by the system to organize and keep track of the different versions of an object. The version number is automatically incremented for each version that is created inside an object.
See Also: Version, Content object.

Version preview interface

The **Version preview** interface within the **Object Edit Interface** displays a preview of any version that exists on the system. In contrast to the **Preview** window that simply shows the attribute contents, this interface lets you view the contents with the design and layout of a particular siteaccess.
See Also: Object Edit Interface, Administration Interface.

Version status

In addition to existing in several versions, each version of a content object is located somewhere in the publishing life cycle; at any given time, a specific version has a version status. The terms "state" and "status" are often used interchangeably. A newly created version has a "Draft" status, and this status will remain until that version is published. "Archived" is the status set for previously published versions that have been replaced by an updated version. As a rule of thumb, a version can only be edited if it is a draft and it can only be edited by the same user who initially created it. The "Pending" status is used by the collaboration system and the delayed publishing mechanism.
See Also: Version, Content object, Draft, Collaboration system.

View

A *view* can be thought of as an interface to a module. By using views, it is possible to reach various functions that a module provides. For example, among other things, the "content" module provides views for displaying, editing, searching and translating the contents of objects. The name of the view that should be accessed appears after the name of the module (separated by a slash) in the URL.
See Also: Module.

Visibility status

Visibility status refers to whether a node (and content located below it) should be displayed and if not, if it has been explicitly or implicitly hidden. A node has one of three visibility statuses: "visible", "hidden", or "hidden by superior".

W

Web 2.0

Web 2.0 refers to the ability for users to interact with and even create content, in order to customize and enrich the user experience. RSS feeds,

polls, forums, wikis, Ajax, comments, blogs, and tags are all Web 2.0 features that are built into eZ Publish.

WebDAV — *Web-based Distributed Authoring and Versioning* (WebDAV) is a set of extensions to the HTTP protocol that enables users to collaboratively view, edit and manage files on remote web servers.

Webshop — eZ Publish comes with an integrated shop mechanism for e-commerce, called the *Webshop*, which plugs directly into the content model. The Webshop's components include: products; Value Added Taxes (VAT) and discounts that affect prices; wish lists and shopping baskets for customers; and orders.

Webshop tab — The **Webshop** tab of the Administration Interface displays an area used to view and modify information that is related to the built-in e-commerce engine (the Webshop).

Website — See: Site.

Website Interface — The Website Interface is a front-end interface available through the "ezwebin" extension. It is an enhancement to public siteaccesses, providing additional functionality to editors and administrators when logged in. In other words, it is what site visitors see, plus more. It includes the **Website Toolbar** for easy access to content editing operations related to the content that is being viewed, and the **Content Editing Interface** for edit operations.
See Also: Website Toolbar, Administration Interface, Content Editing Interface.

Website Toolbar — The **Website Toolbar** is part of the Website Interface, but only available when logged in with at least Editor user privileges. This is the most important tool for editing and managing content through the front-end of the site. An important feature of the **Website Toolbar** is context sensitivity. This means that the buttons and choices shown on the toolbar depend on the content currently displayed and the operations available under the given circumstances.
See Also: Website Interface, Editor user.

Workflow — A *workflow* is a sequential list of events that is started by a trigger. The initiation can be done before or after a function of a module completes, depending on the event. It is commonly used for automated periodic and scheduled maintenance.
See Also: Module, Trigger, Workflow system.

Workflow system — The *workflow system* makes it possible to perform different tasks with or without user interaction. It lets you specify a sequence of actions that will be executed before or after the completion of a module's function and

consists of four components: events, workflows, workflow groups and triggers. An *event*, or more precisely a *workflow event* (not to be confused with calendar events) is the smallest entity of the workflow system. It carries out a specific task.
See Also: Trigger, Workflow.

X

XHTML — *eXtensible HyperText Markup Language* is a reformulation of HTML version 4.0 in XML 1.0. Whereas HTML applies relatively loose standards, XHTML is much stricter. Among other things, the strictness of XHTML results in a faster and more consistent display across different browsers.

XML — *eXtensible Markup Language* is a rich and highly portable format for defining complex documents and data structures.
See Also: Rich text content, Online Editor.

Appendix A. How do I...

The following table presents an overview of how to perform common tasks in both the **Website Interface** and the **Administration Interface**. It can be used as a quick reference guide and also to compare the two interfaces.

Table A.1. Overview of common procedures for both interfaces

Task	**Website Interface (WI)**	**Administration Interface (AI)**
Get editorial access	Acquire a valid user account from your site administrator and access the login page: click the **Login** link (top right corner), enter your username and password, and log in.	Same as for the WI except that you must access the AI siteaccess or mark the corresponding checkbox on the WI login page.
Navigate	Browse the front-end site as normal.	Use the horizontally-aligned tabs, path, secondary / left menu, **Bookmarks** window and **Sub items** window.
View content	Navigate to content.	Locate the content in the secondary menu or main area. Click its name, or select "View" from the context-sensitive pop-up menu. Select the **Preview** switch.
Preview content	Click the **Preview** button on the **Website Toolbar** when in edit mode.	Click the **View** button when in edit mode or select a version number or translation in the **Version history** interface.
Enter edit mode	Click the **Edit** button on the **Website Toolbar** when viewing content, or create new content.	Locate content and select "Edit" from the pop-up menu, or click the **Edit** button in the main area. Alternatively, create new content.

Task	Website Interface (WI)	Administration Interface (AI)
Create new content	Navigate to the desired location and click the **Create here** button on the **Website Toolbar**.	Navigate to the desired location and click the **Create here** button in the **Sub items** window. Alternatively, select "Create here" from the relevant pop-up menu.
Input formatted text	Edit the XML input field using the Online Editor or manually.	Same as for the WI.
Access my drafts	Access your **User profile** page and click the **My drafts** link.	Access the **My account** tab and click the **My drafts** link.
Undo changes	When editing formatted text, use the **Undo** button in the **OE toolbar**. After saving a draft, delete the draft and restart editing. After publishing content, revert to a previous version.	Same as for the WI.
Copy	Not available.	Locate content and either click the **Copy** button for that item in the **Sub items** window or select "Copy" from the pop-up menu.
Move	Locate content and click the **Move** button on the **Website Toolbar**.	Locate content and either click the **Move** button for that item in the **Preview** window or select "Move" from the pop-up menu.
Remove	Locate content and click the **Remove** button on the **Website Toolbar**.	Locate content and either click the **Remove** button for that item in the **Preview** window or select "Remove" from the pop-up menu.

Task	Website Interface (WI)	Administration Interface (AI)
Recover from trash	Not available.	Click the **Trash** link at the bottom of the secondary menu and select the items you want to recover.
Solve an editing conflict	Select an existing draft or create a new one.	Same as for the WI, except that there is a different location to access drafts.
Compare versions	Bring up the **Version history** interface by clicking the **Manage versions** button on the **Website Toolbar** when in edit mode.	Click the **Manage versions** button (in the **Object Edit Interface**), or select "Advanced - Manage versions" from the pop-up menu.
Translate content	Access the content in the siteaccess for the new language and click **Edit** on the **Website Toolbar**.	Enter edit mode as normal, except first choose a new language instead of an existing language.
Sort content	Not available.	Specify the sort order and method for a container object either when creating the content or later using the **Sub items** window.

Index

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Z