



Share your information

eZ Publish 4.5
Technical Manual

©1999 – 2010 eZ Systems AS

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license can be downloaded from <http://www.gnu.org/copyleft/fdl.html>.

Corrections and/or suggestions might be sent to info@ez.no.

This PDF file is generated automatically from the online documentation available at <http://doc.ez.no>.

This version was generated on January 20, 2012.

Contents

1	Installation	16
1.1	Requirements 4.5	18
1.1.1	Install notes for eZ Publish on RHEL 6	25
1.2	Virtual host setup	26
1.2.1	eZ JS Core rewrite rules	30
1.2.2	Virtual host example	32
1.2.3	htaccess example	36
1.3	Normal installation	38
1.3.1	Requirements for doing a normal installation	39
1.3.2	Installing eZ Publish on a Linux/UNIX based system	44
1.3.3	Installing eZ Publish on Windows	48
1.4	Manual installation	52
1.4.1	Requirements for doing a manual installation	53
1.4.2	Manual installation on a Linux/UNIX based system	54
1.4.3	Manual installation on Windows	55
1.4.4	Manual configuration of eZ Publish	56
1.5	Automated installation	63
1.5.1	Requirements for doing an automated installation	64
1.5.2	Automated installation of eZ Publish	65
1.6	The setup wizard	69
1.7	Removing eZ Publish	85
1.8	Extensions	87
1.8.1	Extension load ordering	88
1.8.2	Extracting the files	91

1.8.3	Activating the extension	93
1.9	Troubleshooting	95
2	Concepts and basics	97
2.1	The internal structure of eZ Publish	98
2.1.1	Directory structure	100
2.2	Content and design	102
2.2.1	Storage	104
2.3	Content management	105
2.3.1	Datatypes	107
2.3.2	The content class	108
2.3.3	Class attributes	112
2.3.4	The content object	115
2.3.5	Multiple languages	119
2.3.6	The content node	123
2.3.7	The content node tree	126
2.3.8	Top level nodes	129
2.3.9	Node visibility	131
2.3.10	Object relations	134
2.3.11	Sections	136
2.3.12	URL storage	138
2.3.13	Information collection	139
2.4	Configuration	140
2.4.1	Site management	142
2.4.2	Extension siteaccess settings	146
2.4.3	Access methods	147
2.5	Modules and views	150
2.6	URL translation	153
2.7	Designs	157
2.7.1	Design combinations	159
2.8	Access control	161
2.9	Workflows	165
2.10	Webshop	167

Share your information

3	Templates	172
3.1	Template basics	173
3.1.1	Node templates	176
3.1.2	System templates	178
3.2	The pagelayout	180
3.2.1	The page head	184
3.2.2	Variables in pagelayout	188
3.3	The template language	196
3.3.1	Comments	198
3.3.2	Variable types	199
3.3.3	Variable usage	204
3.3.4	Array and object inspection	208
3.3.5	Control structures	212
3.3.6	Functions and operators	216
3.4	Basic template tasks	218
3.4.1	URL handling	221
3.5	Information extraction	224
3.5.1	Outputting node and object data	226
3.6	The template override system	229
3.6.1	Template override example	231
4	Features	234
4.1	Rest API	235
4.1.1	Installation	236
4.1.2	REST API configuration settings	237
4.1.3	Getting started with the eZ Publish REST API	240
4.1.4	Authentication	241
4.1.5	Resources	250
4.1.6	Extension functionality	266
4.2	Audit trailing	275
4.3	Policy functions	280
4.4	Multi-language	284
4.4.1	Configuring your site locale	288

4.4.2	Configuring the site languages	292
4.4.3	Managing the translation languages	297
4.4.4	Translatable class attributes	299
4.4.5	Translatable country names	304
4.4.6	Multi-lingual objects	306
4.4.7	Working with translations	309
4.4.8	The bit-field algorithm	314
4.4.9	Language-based permissions	316
4.5	Multi-language support for URL aliases	319
4.5.1	Managing URL aliases	326
4.5.2	URL transformation rules	332
4.5.3	Custom transformation commands	335
4.6	Clustering	337
4.6.1	Cluster File Handlers	342
4.6.2	Setting it up for an eZDBFileHandler	345
4.6.3	Setting it up for an eZDFSFileHandler	351
4.6.4	Reverting a cluster setup	357
4.6.5	Maintenance	360
4.6.6	Cluster Configuration Settings	362
4.7	Packages	368
4.7.1	Package types	370
4.7.2	Creating new packages	373
4.7.3	Exporting packages to files	383
4.7.4	Importing packages to the system	385
4.7.5	Removing packages from repository	386
4.7.6	Installing packages	387
4.7.7	Uninstalling packages	393
4.7.8	package.xml format	395
4.7.9	Custom install scripts	398
4.8	Cronjobs	402
4.8.1	The cronjob scripts	403
4.8.2	Configuring cronjobs	411

4.8.3	Running cronjobs	414
4.9	Advanced redirection after login	417
4.10	VAT charging system	421
4.10.1	Assigning VAT types to products	423
4.10.2	Three approaches to VAT charging	425
4.10.3	Product category	430
4.10.4	User country	432
4.10.5	Displaying VATs on the actual site	435
4.10.6	Managing VAT types	437
4.10.7	Managing product categories	440
4.10.8	Managing VAT rules	443
4.10.9	VAT settings	446
4.10.10	Creating new VAT handlers	448
4.11	Improved shipping handling	450
4.12	LDAP Login Handler	454
4.12.1	LDAP Group Mapping Type	457
4.12.2	Roles and Settings	460
4.12.3	Enhancements	461
4.13	Multi-currency	463
4.13.1	Custom prices and auto prices	464
4.13.2	Rounding auto prices	467
4.13.3	Currency rates	469
4.13.4	Creating a new currency	471
4.13.5	Editing a currency	476
4.13.6	Removing a currency	480
4.13.7	Preferred currency	481
4.13.8	Multi-price products	483
4.13.9	Products overview	487
4.13.10	Exchange rates update handlers	488
4.13.11	Upgrading your webshop	491
4.14	View caching	493
4.14.1	Configuring the view cache	496

4.14.2	Clearing the view cache	499
4.14.3	Smart view cache cleaning	502
4.14.4	Pre-generation of view cache	508
4.15	Notifications	509
4.15.1	Using the admin interface	511
4.15.2	Using an actual site	517
4.15.3	Adding a "Keep me updated" button	520
4.15.4	Customizing the E-mails	521
4.15.5	Granting access to notifications	522
4.15.6	Notification events	527
4.15.7	Notification handlers	530
4.15.8	Frequently Asked Questions	533
4.16	Search engine	535
4.17	WebDAV	539
4.17.1	Setting it up	545
4.18	User defined Object States	548
4.19	Language switcher	555
4.20	Queue handling in asynchronous publishing	558
4.20.1	Enable the publishing queue functionality	559
4.20.2	Setting up the asynchronous publishing service with an init script	560
4.20.3	Starting the daemon manually	562
4.20.4	Various configurations	563
4.20.5	Publishing objects asynchronously	564
4.20.6	Make prioritizations in the queue via filtering hooks	565
5	Reference	572

Share your information

List of Figures

1.1	Step 1: Welcome page	70
1.2	System finetuning	71
1.3	Step 2: Issues	71
1.4	Step 3: Outgoing E-mail	72
1.5	Step 4: Database choice	73
1.6	Step 5: Database initialization	74
1.7	Step 6: Language support	75
1.8	Step 7: Site selection	76
1.9	The list of imported packages	77
1.10	Package language options	78
1.11	Step 8: Site access configuration	80
1.12	Step 9: Site details	82
1.13	Step 10: Site administrator	83
1.14	Step 11: Site registration	83
1.15	Step 12: Finished	84
1.16	93
1.17	The debug output appears at the bottom of the page	96
2.1	Libraries, kernel and modules.	98
2.2	Content + Design = Web page	103
2.3	Storage overview	104
2.4	Example of a content class.	108
2.5	109
2.6	Datatypes, attributes, a content class and objects.	115
2.7	Content object structure (with versions and translations).	119

2.8	120
2.9	Object - node relation	123
2.10	Objects, nodes and the content node tree	126
2.11	Content node tree	127
2.12	Objects, node and the content node tree - multiple locations	127
2.13	Content node tree with multiple locations	128
2.14	Top level nodes	129
2.15	Hiding a visible node	132
2.16	Hiding an invisible node	132
2.17	Unhiding a node with a visible ancestor	133
2.18	Unhiding a node with an invisible ancestor	133
2.19	Example of sections.	137
2.20	Example of a setup with two siteaccesses.	142
2.21	Siteaccess directory example.	143
2.22	Configuration override example.	144
2.23	Objects, nodes and nice URLs.	155
2.24	The design fallback mechanism.	159
2.25	Users, groups, policies and roles.	161
2.26	The workflow system.	165
2.27	The integrated e-commerce solution.	168
3.1	Client - server cycle.	174
3.2	The module result as a part of the pagelayout.	174
3.3	Location of pagelayout and full view template in example design.	176
3.4	Pagelayout + node view full template.	177
3.5	The location of the pagelayout (main) template.	180
3.6	The structure of the "ezdate" object.	202
3.7	Typical components of a function call.	216
3.8	Typical components of a template operator call.	217
3.9	The override system.	229
3.10	Template override example.	230
3.11	Example content node tree.	231
3.12	Pagelayout + override templates in example design.	232

3.13	Template override example.	233
4.1	243
4.2	245
4.3	246
4.4	247
4.5	The language selection step in the setup wizard.	293
4.6	294
4.7	294
4.8	297
4.9	298
4.10	300
4.11	301
4.12	301
4.13	301
4.14	303
4.15	303
4.16	List of countries containing translated country names.	305
4.17	306
4.18	307
4.19	308
4.20	310
4.21	311
4.22	311
4.23	312
4.24	313
4.25	316
4.26	317
4.27	318
4.28	321
4.29	326
4.30	329
4.31	330

4.32	373
4.33	374
4.34	374
4.35	375
4.36	375
4.37	376
4.38	376
4.39	377
4.40	377
4.41	378
4.42	378
4.43	379
4.44	380
4.45	381
4.46	381
4.47	382
4.48	383
4.49	384
4.50	385
4.51	385
4.52	386
4.53	387
4.54	388
4.55	388
4.56	388
4.57	389
4.58	389
4.59	390
4.60	390
4.61	391
4.62	391
4.63	392

S
Your
information

4.64	393
4.65	394
4.66	Displaying a custom install script in the list of items during the package installation process	399
4.67	Displaying a custom wizard step during the package installation process	400
4.68	405
4.69	418
4.70	418
4.71	419
4.72	419
4.73	420
4.74	Setting the VAT type on the object level.	423
4.75	Setting the default VAT type on the class level.	424
4.76	Class attribute edit interface for the "Product category" datatype.	430
4.77	A fragment of the product edit interface.	431
4.78	Class attribute edit interface for the "Country" datatype.	432
4.79	The list of VAT types.	437
4.80	The newly added VAT type in the list of VAT types.	438
4.81	The confirmation dialog.	439
4.82	The list of product categories.	440
4.83	The newly added category in the list of product categories.	440
4.84	The confirmation dialog.	442
4.85	The list of VAT charging rules.	443
4.86	The VAT charging rule edit interface.	443
4.87	The newly created VAT rule in the list of VAT charging rules.	444
4.88	The base price in USD and two auto prices.	465
4.89	The base price in USD, non-base custom price in NOK and auto price in EUR. . .	465
4.90	The results of removing the base custom price.	466
4.91	The list of available currencies.	471
4.92	The currency edit interface.	471
4.93	The list of available currencies.	472
4.94	The currency edit interface.	473

4.95	Unknown currency name in the list of currencies.	473
4.96	Displaying inactive currency in the list of currencies.	475
4.97	The list of currencies with disabled possibility to update auto rates.	477
4.98	The list of currencies with updated auto rates.	477
4.99	The list of currencies with removed custom rates.	478
4.100	The list of currencies with one custom rate.	479
4.101	The class edit interface for a product class.	484
4.102	Class attribute edit interface for the "Multi-price" datatype.	484
4.103	The products overview interface.	487
4.104	The resulting prices after product upgrading.	491
4.105	500
4.106	A part of the site content structure.	505
4.107	510
4.108	512
4.109	513
4.110	513
4.111	513
4.112	514
4.113	514
4.114	The "Up" button	514
4.115	515
4.116	516
4.117	516
4.118	The "keep me updated" button.	517
4.119	The "notification added" confirmation for users.	517
4.120	Notification settings for users.	518
4.121	522
4.122	523
4.123	523
4.124	524
4.125	524
4.126	525

4.127	526
4.128	526
4.129	536
4.130	536
4.131	538
4.132	WebDAV - Virtual top folder	539
4.133	WebDAV - Login	540
4.134	WebDAV - Top level nodes	540
4.135	WebDAV - Content node tree	541
4.136	WebDAV - IE open dialog	546
4.137	WebDAV - Content node tree	547
4.138	550
4.139	551
4.140	552
4.141	553
4.142	553
4.143	554
4.144	554

- Installation
- Concepts and basics
- Templates
- Features
- Reference

Chapter 1

Installation

This chapter explains how to obtain and install eZ Publish using the different installation methods. In addition, it also describes how to upgrade or remove an existing eZ Publish installation. If you don't want to install eZ Publish yourself, you can always hire eZ Systems to install and setup the software for you. It is also possible to purchase a hosted eZ Publish solution from various providers and partners.

There are three ways of installing eZ Publish:

1. Normal installation
2. Manual installation
3. Automated installation

Normal installation

This option is the most common and recommended way of installing eZ Publish. It requires a system which already has the proper environment installed, most notably a web server and a database. eZ Publish needs to be downloaded and unpacked. A web-based setup wizard is initiated using a browser. The setup wizard asks a couple of questions and automatically configures eZ Publish. The method is explained under the "Normal installation" (page 38) section.

Manual installation

This option is for experienced users. No wizards or fancy dialogs, no bundled software, no installers, no nothing. This method requires a system which already has a web-server and a database set up and ready to go; eZ Publish needs to be downloaded and unpacked. The system is then configured by manually altering various configuration files and making manual changes to the database. This method is explained under the "Manual installation" (page 52) section.

Automated installation

This installation method (also named kick-start) is for experienced users. It is designed for system administrators who wish to do pre-configured installations of eZ Publish that require a minimum of interaction with the web based setup wizard. It requires a system which already has the proper environment installed, most notably a web server and a database. eZ Publish needs to be downloaded and unpacked. Instead of clicking through the setup wizard and manually providing configuration parameters, the system is installed based on a group of settings defined in a configuration file. This method is explained under the "Automated installation" section.

1.1 Requirements 4.5

This page provides the platform requirements for eZ Publish 4.5.

Reference platforms

eZ Publish 4.5	
Operating system	<ul style="list-style-type: none"> • Debian 6.0, Linux 2.6
Web Server	<ul style="list-style-type: none"> • Official Apache 2.2.17 (pre-fork mode)
DBMS	<ul style="list-style-type: none"> • MySQL 5.0.51a
PHP (mod_php) + PHP CLI	<ul style="list-style-type: none"> • Official PHP 5.2.17
PHP + PHP CLI extensions	<ul style="list-style-type: none"> • APC 3.1.6 (external extension) • bz2 • Curl • dom • exif • fileinfo • ftp • gd • Iconv • json • MBstring • mysqli • pcntl • PCRE • pdo-mysqli • posix • reflection • simplexml • spl • ssl • xmlreader • zlib
Graphic Handler	<ul style="list-style-type: none"> • ImageMagick 6.4.9
eZ Components (Zeta Components)	<ul style="list-style-type: none"> • 2009.2.1
eZ Publish extensions	<ul style="list-style-type: none"> • eZ Online Editor 5.3.0 • eZ Website Interface 1.8.0

	<ul style="list-style-type: none"> • eZ Flow 2.3-0 • eZ Find 2.4-0 • (eZ Find 2.5.0 (with Solr 3.1)) • eZ Archive 1.1.0 • eZ Google Map Location 1.3.0 • eZ Star Rating 1.3.0 • eZ Website Toolbar 1.4.0 • eZ OpenOffice.org 2.6.0 • eZ MB Password Expiry 1.4-0 • eZ Multiupload 1.4-0 • eZ Survey 2.3-0 • eZ Comments 1.2-0 • eZ Teamroom 1.3.0 • eZ JSCore 1.3-0 • eZ Script Monitor 1.2-0 • eZ SI 1.4.0 • eZ Style Editor 1.3-0 • eZ XML Export 1.3-0 • eZ Image Editor 1.2.0 • eZ Network 1.2.1
Cluster mode	<ul style="list-style-type: none"> • eZDBFileHandler • eZDFSFileHandler + Linux NFS

Approved platforms

These platforms are also tested, but not as extensively as our reference platforms. These platforms still benefit from the full support and maintenance guarantees provided with the eZ Publish Enterprise Subscription, but more issues might occur during normal operations, performance might be lower and issues take longer to resolve.

	eZ Publish 4.5, Single Mode	eZ Publish 4.5, Cluster Mode
Operating system	<ul style="list-style-type: none"> • Linux 2.6 • Windows 2008 (IIS 7 only) • Red Hat Enterprise Linux (RHEL) 6 (see separate installation notes) (page 25) • Debian 6 • SUSE Linux Enterprise Server (SLES) 11 	<ul style="list-style-type: none"> • Linux 2.6 • Red Hat Enterprise Linux (RHEL) 6 • Debian 6 • SUSE Linux Enterprise Server (SLES) 11
Web Server	<ul style="list-style-type: none"> • Apache 2.2.x (prefork) 	<ul style="list-style-type: none"> • Apache 2.2.x (prefork)

	mode) <ul style="list-style-type: none"> • MS IIS 7 (with Microsoft URL Rewrite Module 1.1) 	mode)
DBMS	<ul style="list-style-type: none"> • MySQL 5.0.x • MySQL 5.1.x • PostgreSQL 8.4 • Oracle 11g 	<ul style="list-style-type: none"> • MySQL 5.0.x • MySQL 5.1.x • Oracle 11g
PHP (mod_php) + PHP CLI + apache	<ul style="list-style-type: none"> • PHP 5.2.1 -> PHP 5.2.17 • PHP 5.3.x 	<ul style="list-style-type: none"> • PHP 5.2.1 -> PHP 5.2.17 • PHP 5.3.x
PHP (FastCGI) + PHP CLI + IIS	<ul style="list-style-type: none"> • PHP 5.3.x 	
PHP	<ul style="list-style-type: none"> • APC 3.1.6 (external extension) • bz2 • Curl • dom • exif • fileinfo • ftp • gd • Iconv • json • MBstring • mysql, mysqli • pcntl • PCRE • pdo-mysqli • pdo-pgsql • pdo-ocllsl • pdo-oci • posix • reflection • simplexml • spl • ssl • xmlreader • zlib 	<ul style="list-style-type: none"> • APC 3.1.6 (external extension) • bz2 • Curl • dom • exif • fileinfo • ftp • gd • Iconv • json • MBstring • mysqli • PCRE • posix • reflection • simplexml • spl • xmlreader • zlib
Graphic Handler	<ul style="list-style-type: none"> • ImageMagick >= 6.4.x • PHP extension GD2 	<ul style="list-style-type: none"> • ImageMagick >= 6.4.x • PHP extension GD2
eZ Components		

	<ul style="list-style-type: none"> • 2009.2.1 	<ul style="list-style-type: none"> • 2009.2.1
eZ Publish extensions	<ul style="list-style-type: none"> • eZ Star Rating 1.3.0 • eZ Website Toolbar 1.4.0 • eZ OpenOffice.org 2.6.0 • eZ MB Password Expiry 1.4.0 • eZ Multiupload 1.4.0 • eZ Survey 2.3.0 • eZ Comments 1.2.0 • eZ Teamroom 1.3.0 • eZ JSCore 1.3.0 • eZ Script Monitor 1.2.0 • eZ SI 1.4.0 • eZ Style Editor 1.3.0 • eZ XML Export 1.3.0 • eZ Image Editor 1.2.0 • eZ Network 1.2.1 • eZ Oracle 2.2.0 	<ul style="list-style-type: none"> • eZ Star Rating 1.3.0 • eZ Website Toolbar 1.4.0 • eZ OpenOffice.org 2.6.0 • eZ MB Password Expiry 1.4.0 • eZ Multiupload 1.4.0 • eZ Survey 2.3.0 • eZ Comments 1.2.0 • eZ Teamroom 1.3.0 • eZ JSCore 1.3.0 • eZ Script Monitor 1.2.0 • eZ SI 1.4.0 • eZ Style Editor 1.3.0 • eZ XML Export 1.3.0 • eZ Image Editor 1.2.0 • eZ Network 1.2.1 • eZ Oracle 2.2.0 (eZ DFS only)
Cluster mode		<ul style="list-style-type: none"> • eZDBFileHandler (mysql only) • eZDFSFileHandler (mysql and Oracle) + Linux NFS

Compatible platforms

eZ Publish can run and execute on many more platforms than the ones listed above. However, eZ Systems doesn't insure and guarantee quality operation of an eZ Publish Enterprise installation if it is running on any platform not listed as either REFERENCE or APPROVED. eZ Publish Enterprise Subscriptions are still available for compatible platforms, but the guarantee and the product support will not apply and although you will receive various maintenance releases and services, no bug fix guarantee will apply to issues related to the platform. Maintenance and monitoring tools will not be available. eZ Systems does not advise merely compatible platforms for production use.

	eZ Publish 4.5, Single Mode	eZ Publish 4.5, Cluster Mode
Operating system	<ul style="list-style-type: none"> • Linux 2.6 • Solaris 10 Intel • Opensolaris 2008.11 	<ul style="list-style-type: none"> • Linux 2.6 • Solaris 10 Intel • Opensolaris 2008.11

Your information

	<ul style="list-style-type: none"> Intel • Opensolaris 2009.06 Intel • Windows 2000/XP • Windows 2008 (IIS 7 only) • Mac OSX server 	<ul style="list-style-type: none"> Intel • Opensolaris 2009.06 Intel • Mac OSX server
Web Server	<ul style="list-style-type: none"> • Apache 1.3.x • Apache 2.2.x (prefork mode) • MS IIS 7 (with Microsoft URL Rewrite Module 1.1) • LIGHTPTD • MS IIS 6 • Nginx 	<ul style="list-style-type: none"> • Apache 1.3.x • Apache 2.2.x (prefork mode)
DBMS	<ul style="list-style-type: none"> • MySQL 5.0.x • MySQL 5.1.x • PostgreSQL • Oracle 9/10/11g 	<ul style="list-style-type: none"> • MySQL 5.0.x • MySQL 5.1.x • Oracle 11g
PHP (mod_php) + PHP CLI + apache	<ul style="list-style-type: none"> • PHP 5.2.1 -> PHP 5.2.17 • PHP 5.3.x 	<ul style="list-style-type: none"> • PHP 5.2.1 -> PHP 5.2.17 • PHP 5.3.x
PHP (FastCGI) + PHP CLI + IIS	<ul style="list-style-type: none"> • PHP 5.3.x 	
PHP	<ul style="list-style-type: none"> • APC 3.1.6 (external extension) • bz2 • Curl • dom • exif • fileinfo • ftp • gd • Iconv • json • MBstring • mysqli, mysql • pcntl • PCRE • pdo-mysqli • pdo-pgsql 	<ul style="list-style-type: none"> • APC 3.1.6 (external extension) • bz2 • Curl • dom • exif • fileinfo • ftp • gd • Iconv • json • MBstring • mysqli • PCRE • posix • reflection • simplexml

	<ul style="list-style-type: none"> • pdo-ocsl • pdo-oci • posix • reflection • simplexml • spl • ssl • xmlreader • zlib • Wincache 1.1 (external extension) 	<ul style="list-style-type: none"> • spl • xmlreader • zlib • Wincache 1.1 (external extension)
Graphic Handler	<ul style="list-style-type: none"> • ImageMagick >= 6.4.x • PHP extension GD2 	<ul style="list-style-type: none"> • ImageMagick >= 6.4.x • PHP extension GD2
eZ Components	<ul style="list-style-type: none"> • 2008.2.2-2009.2.1 - 2009.2.1 	<ul style="list-style-type: none"> • 2008.2.2-2009.2.1 -2009.2.1
eZ Publish extensions	<ul style="list-style-type: none"> • eZ Star Rating 1.3.0 • eZ Website Toolbar 1.4.0 • eZ OpenOffice.org 2.6.0 • eZ MB Password Expiry 1.4.0 • eZ Multiupload 1.4.0 • eZ Survey 2.3.0 • eZ Comments 1.2.0 • eZ Teamroom 1.3.0 • eZ JSCore 1.3.0 • eZ Script Monitor 1.2.0 • eZ SI 1.4.0 • eZ Style Editor 1.3.0 • eZ XML Export 1.3.0 • eZ Image Editor 1.2.0 • eZ Network 1.2.1 • eZ Oracle 2.2.0 	<ul style="list-style-type: none"> • eZ Star Rating 1.3.0 • eZ Website Toolbar 1.4.0 • eZ OpenOffice.org 2.6.0 • eZ MB Password Expiry 1.4.0 • eZ Multiupload 1.4.0 • eZ Survey 2.3.0 • eZ Comments 1.2.0 • eZ Teamroom 1.3.0 • eZ JSCore 1.3.0 • eZ Script Monitor 1.2.0 • eZ SI 1.4.0 • eZ Style Editor 1.3.0 • eZ XML Export 1.3.0 • eZ Image Editor 1.2.0 • eZ Network 1.2.1 • eZ Oracle 2.2.0 (eZ DFS only)
Cluster mode		<ul style="list-style-type: none"> • eZDBFileHandler • eZDFSFileHandler
Distributed file system		<ul style="list-style-type: none"> • Linux NFS • Solaris NFS • Sun ZFS • Oracle OCFS

- Redhat GFS

Share your information

1.1.1 Install notes for eZ Publish on RHEL 6

Red Hat Network

- Log into Red Hat Network
- Click on €Channels€ in the menu bar.
- Click € Red Hat Optional Server 6€, €target systems€ and enable this channel for your server

Required RPMs

php-pear-Net-Curl	Required by ezfind
php-pear-Date	Red Hat's rpm for datetime
php-mbstring	Required by ezini
php-pecl-apc	The supported rpm equivalent to the pear install
php-process	Required for eZ scripts and cron jobs
php-mcrypt	
php-xml	
php-pgsql	PostgreSQL support
php-mysql	MySQL support
php-mbstring	Note: Available from the optional rpm channel

1.2 Virtual host setup

This section describes how to set up a virtual host for eZ Publish using the Apache web-server. A virtual host setup is only needed if eZ Publish has been configured to use the host access method, which is the most secure method.

By making use of virtual hosts, it is possible to have several sites running on the same server. The sites are usually differentiated by the name they are accessed. Apache will look for a specified set of domains and use different configuration settings based on the domain that is accessed.

Generic virtual host setup

Virtual hosts are usually defined at the end of "httpd.conf", which is the main configuration file for Apache. Adding a virtual host for eZ Publish can be done by copying the following lines and replacing the text encapsulated by the square brackets with actual values. Please refer to the next section for a real life example of using virtual hosts.

```
NameVirtualHost [IP_ADDRESS]

<VirtualHost [IP_ADDRESS]:[PORT]>
  <Directory [PATH_TO_EZPUBLISH]>
    Options FollowSymLinks
    AllowOverride None
  </Directory>

  <IfModule mod_php5.c>
    php_admin_flag safe_mode Off
    php_admin_value register_globals 0
    php_value magic_quotes_gpc 0
    php_value magic_quotes_runtime 0
    php_value allow_call_time_pass_reference 0
  </IfModule>

  DirectoryIndex index.php

  <IfModule mod_rewrite.c>
    RewriteEngine On
    RewriteRule ^/api/ /index_rest\.php [L]
    RewriteRule ^/([~/]+)?content/treemenu.* /index_treemenu\.php [L]
    RewriteRule ^/var/([~/]+)?storage/images(-versioned)?/. * - [L]
    RewriteRule ^/var/([~/]+)?cache/(texttoimage|public)/. * - [L]
    RewriteRule ^/design/[~/]+/(stylesheets|images|javascript)/. * - [L]
    RewriteRule ^/share/icons/. * - [L]
    RewriteRule ^/extension/[~/]+/design/[~/]+/
(stylesheets|flash|images|lib|javascripts?)/. * - [L]
```

```

RewriteRule ^/packages/styles/.+/(stylesheets|images|javascript)/[~/]+/
.* - [L]
RewriteRule ^/packages/styles/.+/thumbnail/.* - [L]
RewriteRule ^/var/storage/packages/.* - [L]

# Makes it possible to place your favicon at the root of your
# eZ Publish instance. It will then be served directly.
RewriteRule ^/favicon\.ico - [L]
# Uncomment the line below if you want you favicon be served
# from the standard design. You can customize the path to
# favicon.ico by changing /design/standard/images/favicon\.ico
#RewriteRule ^/favicon\.ico /design/standard/images/favicon\.ico [L]
RewriteRule ^/design/standard/images/favicon\.ico - [L]

# Give direct access to robots.txt for use by crawlers (Google,
# Bing, Spammers..)
RewriteRule ^/robots\.txt - [L]

# Platform for Privacy Preferences Project ( P3P ) related files
# for Internet Explorer
# More info here : http://en.wikipedia.org/wiki/P3p
RewriteRule ^/w3c/p3p\.xml - [L]

# Uncomment the following lines when using popup style debug
# RewriteRule ^/var/cache/debug\.html.* - [L]
# RewriteRule ^/var/[~/]+/cache/debug\.html.* - [L]

RewriteRule .* /index\.php
</IfModule>

DocumentRoot [PATH_TO_EZPUBLISH]
ServerName [SERVER_NAME]
ServerAlias [SERVER_ALIAS]

</VirtualHost>

```

[IP_ADDRESS]	The IP address of the virtual host, for example "128.39.140.28". Apache allows the usage of a wildcards here ("*").
[PORT]	The port on which the webserver listens for incoming requests. This is an optional setting, the default port is 80. The combination of an IP address and a port is often referred to as a socket. Apache allows the usage of a wildcards here ("*").
[PATH_TO_EZPUBLISH]	Path to the directory that contains eZ Publish.

	This must be the full path, for example <code>"/var/www/ezpublish-3.6.0"</code> .
<code>[SERVER_NAME]</code>	The host or the IP address that Apache should look for. If a match is found, the virtual host settings will be used.
<code>[SERVER_ALIAS]</code>	Additional hosts/IP addresses that Apache should look for. If a match is found, the virtual host settings will be used.

Please note that the `"mod_rewrite"` module must be enabled in `"httpd.conf"` in order to use the Rewrite Rules.

NameVirtualHost

The `"NameVirtualHost"` setting might already exist in the default configuration. Defining a new one will result in a conflict. If Apache reports errors such as `"NameVirtualHost [IP_ADDRESS] has no VirtualHosts"` or `"Mixing * ports and non-* ports with a NameVirtualHost address is not supported"`, try skipping the `NameVirtualHost` line. For more info about the `NameVirtualHost` directive, see <http://httpd.apache.org/docs/1.3/mod/core.html#namevirtualhost>.

SOAP and WebDAV

If you would like to use the SOAP and/or the WebDAV features of eZ Publish, you'll have to add the following lines in the virtual host configuration:

```
RewriteCond %{HTTP_HOST} ^webdav\..*
RewriteRule ^(.*) /webdav.php [L]

RewriteCond %{HTTP_HOST} ^soap\..*
RewriteRule ^(.*) /soap.php [L]

ServerAlias soap.example.com
ServerAlias webdav.example.com
```

Optional re-write rules to improve performance

```
# Everything below is optional to improve performance by forcing
# clients to cache image and design files, change the expires time
# to suite project needs.
<IfModule mod_expires.c>
  <LocationMatch "^/var/[^/]+/storage/images/.*">
    # eZ Publish appends the version number to image URL (ezimage
    # datatype) so when an image is updated, its URL changes to
    ExpiresActive on
```

```
ExpiresDefault "now plus 10 years"
</LocationMatch>

<LocationMatch "^/extension/[^/]+/design/[^/]+/
(stylesheets|images|javascripts?|flash)/.*">
# A good optimization if you don't change your design often
ExpiresActive on
ExpiresDefault "now plus 5 days"
</LocationMatch>

<LocationMatch "^/extension/[^/]+/design/[^/]+/lib/.*">
# Libraries get a new url (version number) on updates
ExpiresActive on
ExpiresDefault "now plus 90 days"
</LocationMatch>

<LocationMatch "^/design/[^/]+/
(stylesheets|images|javascripts?|lib|flash)/.*">
# Same as above for bundled eZ Publish designs
ExpiresActive on
ExpiresDefault "now plus 7 days"
</LocationMatch>

<LocationMatch "^/share/icons/.*">
# Icons as used by admin interface, barly change
ExpiresActive on
ExpiresDefault "now plus 7 days"
</LocationMatch>

# When ezjscore.ini/[Packer]/AppendLastModifiedTime=enabled
# so that file names change when source files are modified
#<LocationMatch "^/var/[^/]+/cache/public/.*">
# Force ezjscore packer js/css files to be cached 30 days
# at client side
#ExpiresActive on
#ExpiresDefault "now plus 30 days"
#</LocationMatch>
</IfModule>
```


(optional) Speedup ajax calls

Copy or symlink `index_ajax.php` from this extension to the root folder of eZ Publish (next to `index.php`). Add the following rewrite rule:

- `.htaccess`

```
RewriteRule ezjscore/call.* index_ajax.php  
RewriteRule ^index_ajax\.php - [L]
```

- ` Virtual Host mode`

```
RewriteRule ^/([~/]+)?ezjscore/call.* /index_ajax\.php [L]
```


1.2.2 Virtual host example

This example demonstrates how to set up a virtual host on the Apache web server for an eZ Publish installation located in `"/var/www/example"`. Let's say that we want to access eZ Publish by using the following URLs:

- `http://www.example.com` (actual website for public access)
- `http://admin.example.com` (administration interface for the webmaster)

In order to achieve this, we need to set up both eZ Publish and the web server so that they respond correctly to the different requests.

eZ Publish configuration: siteaccess settings

eZ Publish needs to be configured to use the host access method. This can be done from within the web based setup wizard or by manually editing the global override for the `site.ini` configuration file: `"/settings/override/site.ini.append.php"`. A typical configuration would look something like this:

```
...
[SiteAccessSettings]
AvailableSiteAccessList []
AvailableSiteAccessList []=example
AvailableSiteAccessList []=example_admin
MatchOrder=host

HostMatchMapItems []=www.example.com;example
HostMatchMapItems []=admin.example.com;example_admin
...
```

This configuration tells eZ Publish that it should use the `"example"` siteaccess if a request starts with `"www.example.com"` and `"example_admin"` if the request starts with `"admin.example.com"`. For more information about site management in eZ Publish, please refer to the `"Site management"` (page [142](#)) part of the `"Concepts and basics"` chapter.

Apache configuration: virtual host settings

Assuming that...

- eZ Publish is located in `"/var/www/example"`
- the server's IP address is `128.39.140.28`
- we wish to access eZ Publish using `"www.example.com"` and `"admin.example.com"`

...the following virtual host configuration needs to be added at the end of "http.conf":

```
NameVirtualHost 128.39.140.28

<VirtualHost 128.39.140.28>
  <Directory /var/www/example>
    Options FollowSymLinks
    AllowOverride None
  </Directory>

  <IfModule mod_php5.c>
    php_admin_flag safe_mode Off
    php_admin_value register_globals 0
    php_value magic_quotes_gpc 0
    php_value magic_quotes_runtime 0
    php_value allow_call_time_pass_reference 0
  </IfModule>

  DirectoryIndex index.php

  <IfModule mod_rewrite.c>
    RewriteEngine On
    RewriteRule ^/api/ /index_rest\.php [L]
    RewriteRule ^/([~/]+)?content/treemenu.* /index_treemenu\.php [L]
    RewriteRule ^/var/([~/]+)?storage/images(-versioned)?/.* - [L]
    RewriteRule ^/var/([~/]+)?cache/(texttoimage|public)/.* - [L]
    RewriteRule ^/design/[~/]+/(stylesheets|images|javascript)/.* - [L]
    RewriteRule ^/share/icons/.* - [L]
    RewriteRule ^/extension/[~/]+/design/[~/]+/
(stylesheets|flash|images|lib|javascripts?)/.* - [L]
    RewriteRule ^/packages/styles/.+/(stylesheets|images|javascript)/[~/]+/
.* - [L]
    RewriteRule ^/packages/styles/.+/thumbnail/.* - [L]
    RewriteRule ^/var/storage/packages/.* - [L]

    # Makes it possible to place your favicon at the root of your
    # eZ Publish instance. It will then be served directly.
    RewriteRule ^/favicon\.ico - [L]
    # Uncomment the line below if you want you favicon be served from the
    standard design.
    # You can customize the path to favicon.ico by replacing design/
    standard/images/favicon.ico
    # by the adequate path.
    #RewriteRule ^/favicon\.ico /design/standard/images/favicon\.ico [L]
    RewriteRule ^/design/standard/images/favicon\.ico - [L]
```

```

# Give direct access to robots.txt for use by crawlers (Google, Bing,
Spammers..)
RewriteRule ~/robots\.txt - [L]

# Platform for Privacy Preferences Project ( P3P ) related files for
Internet Explorer
# More info here : http://en.wikipedia.org/wiki/P3p
RewriteRule ~/w3c/p3p\.xml - [L]

# Uncomment the following lines when using popup style debug.
# RewriteRule ~/var/cache/debug\.html.* - [L]
# RewriteRule ~/var/[~/]+/cache/debug\.html.* - [L]

RewriteRule .* /index\.php
</IfModule>

DocumentRoot /var/www/example
ServerName www.example.com
ServerAlias admin.example.com
</VirtualHost>

```

Note that it isn't necessary to create a separate virtual host block for "admin.example.com", it can be added to the existing block using the "ServerAlias" directive.

You can have apache1 and apache2 part in the sample vhost. That way allows to use one vhost for both servers.

```

<IfModule mod_php5.c>
# If you are using Apache 2, you have to use <IfModule sapi_apache2.c>
# instead of <IfModule mod_php5.c>.
# some parts/addons might only run safe mode on
php_admin_flag safe_mode Off
# security just in case
php_admin_value register_globals 0
# performance
php_value magic_quotes_gpc 0
# performance
php_value magic_quotes_runtime 0
#http://www.php.net/manual/en/
ini.core.php#ini.allow-call-time-pass-reference
php_value allow_call_time_pass_reference 0
</IfModule>

<IfModule sapi_apache2.c>
# If you are using Apache 2, you have to use <IfModule sapi_apache2.c>
# instead of <IfModule mod_php5.c>.

```

```
# some parts/addons might only run safe mode on
php_admin_flag safe_mode Off
# security just in case
php_admin_value register_globals 0
# performance
php_value magic_quotes_gpc 0
# performance
php_value magic_quotes_runtime 0
#http://www.php.net/manual/en/
ini.core.php#ini.allow-call-time-pass-reference
php_value allow_call_time_pass_reference 0
</IfModule>
```

1.2.3 htaccess example

These rewrite rules can be used if you don't have direct access to alter the virtual host configuration, which is common in a shared hosting setup. The file ".htaccess_root" is included in the installation package of your eZ Publish installation. In order to activate these rewrite rules in your hosted environment, you must change the name of the file to ".htaccess" and keep in the same folder as the original ".htaccess_root" file.

Please find an example of the ".htaccess" rewrite rules below:

```
# Copy this file to a new file called .htaccess in your eZ Publish root
# to secure your installation by turning on .htaccess based virtual host mode.
```

```
DirectoryIndex index.php
```

```
RewriteEngine On
```

```
RewriteRule ^api/ index_rest.php [L]
```

```
RewriteRule ^index_rest\.php - [L]
```

```
RewriteRule ^([\w/]+)?content/treemenu.* index_treemenu\.php [L]
```

```
RewriteRule ^var/([\w/]+)?storage/images(-versioned)?/.* - [L]
```

```
RewriteRule ^var/([\w/]+)?cache/(texttoimage|public)/.* - [L]
```

```
RewriteRule ^design/[\w/]+/(stylesheets|images|javascript)/.* - [L]
```

```
RewriteRule ^share/icons/.* - [L]
```

```
RewriteRule ^extension/[\w/]+/design/[\w/]+/
```

```
(stylesheets|flash|images|lib|javascripts?)/.* - [L]
```

```
RewriteRule ^packages/styles/.+/(stylesheets|images|javascript)/[\w/]+/.* - [L]
```

```
RewriteRule ^packages/styles/.+/thumbnail/.* - [L]
```

```
RewriteRule ^var/storage/packages/.* - [L]
```

```
# Makes it possible to placed your favicon at the root of your
```

```
# eZ Publish instance. It will then be served directly.
```

```
RewriteRule ^favicon\.ico - [L]
```

```
# Uncomment the line below if you want you favicon be served from the standard
design.
```

```
# You can customize the path to favicon.ico by replacing design/standard/
images/favicon.ico
```

```
# by the adequate path.
```

```
#RewriteRule ^favicon\.ico /design/standard/images/favicon\.ico [L]
```

```
RewriteRule ^design/standard/images/favicon\.ico - [L]
```

```
# Give direct access to robots.txt for use by crawlers (Google, Bing,
Spammers..)
```

```
RewriteRule ^robots\.txt - [L]
```

```
# Uncomment the following lines when using popup style debug.
```

```
# RewriteRule ^var/cache/debug\.html.* - [L]
```

```
# RewriteRule ^var/[\w/]+/cache/debug\.html.* - [L]
```

```
# Platform for Privacy Preferences Project ( P3P ) related files for Internet Explorer
# More info here : http://en.wikipedia.org/wiki/P3p
RewriteRule ^w3c/p3p\.xml - [L]

RewriteRule .* index\.php
```

1.3 Normal installation

The normal installation method is the most common and recommended way of deploying eZ Publish. It requires a system which already has the proper environment installed, most notably a web server and a database. The necessary requirements are explained in detail within the next section (page 39). A typical normal installation process consists of the following steps:

- Setting up / creating a database
- Downloading a packaged eZ Publish distribution
- Unpacking the eZ Publish distribution
- Initiating and going through the web based setup wizard

Once the web based setup wizard has completed, eZ Publish will be ready for use.

The "Installing eZ Publish on a Linux/UNIX based system (page 44)" and "Installing eZ Publish on Windows (page 48)" sections (depending on the target OS) will take you through the necessary steps.

1.3.1 Requirements for doing a normal installation

eZ Publish makes use of and depends on four important things:

1. A web server
2. A server-side PHP scripting engine
3. The eZ Components library
4. A database server
5. An image conversion system (optional)

The first three things should be in place before an eZ Publish installation is deployed. The image conversion system is optional and is only needed if you're planning to use eZ Publish with images. The web server and the server-side PHP scripting engine has to run on the same machine. The database server may run on a different computer.

Please also visit the requirements page (page 18) to check if your operation system is supported.

For the moment, the following software solutions can be used:

Web server

Currently, only the Apache web server is supported. On Linux/UNIX based systems, it is recommended to use the latest version of the 2.2.x branch (applies also to cluster mode). Note that it must run in "prefork" mode instead of "threaded" mode - the reason for this is because some of the libraries that PHP extensions use might not be thread-safe.

On Windows, it is recommended to use MS IIS 7 (with Microsoft URL Rewrite Module 1.1). (Apache 2.x for Windows is not supported since it only exists in "threaded" mode.)

The Apache web server is the most popular web server on the planet. It is free, open source and can be downloaded from <http://www.apache.org>.

Server-side PHP scripting engine

Since most of the eZ Publish system is written using the PHP scripting language, a PHP (hypertext pre-processor) server-side engine is needed. Make sure you have PHP 5.1.6 or later.

Note that it is strongly recommended to use the latest version of the 5.x branch, which is PHP 5.2.17 at the time of writing. The reason for this is that eZ Publish runs faster on PHP 5.2 than on PHP 5.1. In addition, some extensions may require PHP 5.2 (for example, the eZ Flow extension that comes together with eZ Publish). Make sure you use the PHP version that is required for your specific eZ Components version.

PHP is free software and can be downloaded from <http://www.php.net>. The following table reveals which functionality PHP needs to have compiled-in support for.

Name	Description
MySQLi extension (recommended) or MySQL functions	Required if a MySQL database will be used.
PostgreSQL functions	Required if a PostgreSQL database will be used.
Zlib compression functions	Required (see below).
DOM functions	Required (see below).
Session support	Required (enabled in PHP by default).
PCRE functions	Required (enabled in PHP by default).
GD2 support	Required if ImageMagick is not installed.
CLI support	Recommended (see below).
Client URL library functions	Recommended (see below).
Multibyte string functions	Required.
Exif functions	Recommended.

Zlib extension

Make sure that zlib support in PHP is enabled, otherwise the setup wizard (page 69) will not be able to unpack downloaded packages during the installation process.

DOM extension

In most cases, [DOM functions](#) are enabled by default as they are included in the PHP core. However, some Linux distributions have PHP without compiled-in support for DOM. Instead, they provide DOM as a shared module in a separate RPM package called "php-xml".

PHP CLI

It is strongly recommended to have [PHP CLI](#) installed, otherwise some features like notifications (page 509), delayed search indexing, upgrade scripts, the collaboration system (content approval), clearing caches from within the command line, etc. will not work.

CURL

It is recommended to enable [CURL](#) support, otherwise some features like outbound connections via proxy and [SSL support for eZSoap](#) will not work.

PHP memory limit issue

eZ Publish needs at least 64 MB in order to complete the setup wizard. If you are using PHP 5.2.0 or earlier version, you'll have to increase the default "memory_limit" setting which is located in the "php.ini" configuration file. (Don't forget to restart Apache after editing "php.ini".) Normal

operation requires about 16 MB. However, it is highly recommended that you keep the 64 MB setting since eZ Publish consumes a lot more memory as soon as you reindex the search, execute upgrade scripts, etc. Multilingual sites will also require at least 64 MB.

If you are using PHP 5.2.1 or later, there is no need to change the default "memory_limit" setting (it is set to 128 MB by default).

PHP timezone

You need to set the "[date.timezone](#)" value in the "php.ini" configuration file. If this setting is not specified, you will most likely receive error messages like "It is not safe to rely on the system's timezone settings" when running eZ Publish on PHP 5. The following example shows how the corresponding line in "php.ini" looks like:

```
date.timezone = <timezone>
```

Refer to the [PHP documentation](#) for the list of supported timezones. Don't forget to restart Apache after editing "php.ini".

eZ Components library

eZ Publish is an object-oriented application where each class definition is stored in a separate PHP source file. Instead of having a list of needed includes at the beginning of each source file, eZ Publish 4 makes use of the [_autoload\(\)](#) function. When eZ Publish is installed, all class definitions of the eZ Publish kernel will have their paths listed in the "autoload/ezp_kernel.php" file. In addition, the "autoload/ezp_extension.php" file will contain an array of paths for class definitions that are a part of the extensions that come with eZ Publish. These arrays will most likely need to be updated in the future (for example, when you install new extensions or configure existing ones using the "Setup - Extensions" part of the administration interface). The minimum eZ Components version required with eZ Publish 4.5 is "ezcomponents-ezp45", containing a fix to the package system. In particular, you need to install the File and Base components ("ezcBase" and "ezcFile"), otherwise eZ Publish will not be able to update autoload arrays.

eZ Components is an enterprise ready general purpose PHP components library used independently or together for PHP application development. eZ Components can be downloaded from <http://ezcomponents.org/download>. In the future, eZ Components will be bundled with eZ Publish. Refer to <http://ezcomponents.org/docs/install> for information about how to install eZ Components.

Important note

Starting from version 2008.1, the eZ Components library requires PHP version 5.2.1 or higher.

Database server

eZ Publish stores miscellaneous data structures and actual content using a database. This means that a database server has to be available for eZ Publish at all times. Follow this link to the [eZ Publish requirements page \(page 39\)](#) to find which database solutions eZ Publish is compatible with.

The setup wizard will automatically detect the database server as long as it is running on the same computer that functions as the web server. eZ Publish 4 requires a UTF-8 database.

Note that eZ Publish 4 does not support clustering (page 337) for PostgreSQL databases. The clustering code is optimized for best performance and focused on MySQL databases using the InnoDB [storage engine](#).

Even if you are not going to run eZ Publish in a clustered environment, the use of InnoDB is required. This storage engine makes it possible to use transaction-safe tables in a MySQL database. (Database transaction support is enabled by default in eZ Publish. This feature makes the system less vulnerable to database errors and inconsistencies due to aborted requests.) Contact your database administrator if you are unsure about whether InnoDB is available on your server.

If you want to use PostgreSQL, make sure the ["pgcrypto"](#) module is installed. On Linux/UNIX, you may need to install a separate package called ["postgresql-contrib"](#) (refer to the [PostgreSQL documentation](#) for more information), which contains the ["pgcrypto"](#) module. The ["pgcrypto"](#) module provides cryptographic functions for PostgreSQL, including the ["digest"](#) function, which is needed for eZ Publish. When setting up a PostgreSQL database for eZ Publish, you will have to register these functions in the database. Refer to the ["Setting up a database"](#) part of the ["Installing eZ Publish on a Linux/UNIX based system \(page 44\)"](#) and ["Installing eZ Publish on Windows \(page 48\)"](#) documentation pages (depending on the target OS) for more information.

Known issue with running PHP5.3 on MySQL: Some people (like Windows users with both IPv4 and IPv6 installed) experience problems connecting to the database server using host names like ["localhost"](#)... If you experience problems, try using IPv4 address like ["127.0.0.1"](#). This is due to a connectivity problem when running PHP5.3 on MySQL. So, please replace the database server name ["localhost"](#) with the IP address of the machine, or ["127.0.0.1"](#), which is reserved for the local host.

Session parameters

In order to support `ezpSessionHandlerDB`, `session.hash_function` and `session.hash_bits_per_character` in `php.ini` should be combined with the goal of never generating keys longer than 32 bytes.

Recommended settings:

```
session.hash_function = 1
session.hash_bits_per_character = 4
```

Oracle compatibility

The version 1.8 of the eZ Publish Extension for Oracle Database makes it possible to use Oracle as a database for eZ Publish 4.0.1 and higher. Note that earlier versions of the extension are not compatible with eZ Publish 4.

Image conversion system (optional)

In order to scale, convert or modify images, eZ Publish needs to make use of an image conversion system. One of the following software packages (both are free) can be used:

- GD2 (comes with PHP)
- ImageMagick (<http://www.imagemagick.org>)

ImageMagick supports more formats than GD and usually produces better results (better scaling, etc.). The setup wizard will automatically detect the pre-installed image conversion system(s).

The installation and setup of required software solutions (outlined above) is far beyond the scope of this document. Please refer to the homepage and documentation of the different software solutions.

Limitation on some file systems when storing large number of content files

eZ Publish stores all disc related content (eg Images, PDF's etc) in var/storage like the structure from content tree, creating one folder for each object. In most file systems used under Linux (especially ext2 + ext3) there exists a hard LIMIT TO 32000 directories per folder. So it is not possible to store more than 31999 objects under one folder.

To get around this limitation without changing the file system, you can split your content tree so that you don't have more than 32k content files (example: images) in the same folder.

Examples of file systems that supports more file/folder entries per folder.

- ReiserFS: roughly 1.2 million per directory
- ZFS: 2^{48} (a really big number: 281474976710656)!

1.3.2 Installing eZ Publish on a Linux/UNIX based system

The requirements for doing a normal installation must be met! Read the "Requirements for doing a normal installation (page 39)" section first. Proceed only if you have access to a Linux/UNIX based system with Apache, PHP, MySQL or PostgreSQL already installed and running. As mentioned earlier, the database server may run on a different computer than the web server. This section will guide you through the following steps:

- Setting up a database (MySQL or PostgreSQL)
- Downloading eZ Publish
- Unpacking eZ Publish
- Initiating the setup wizard

Setting up a database

A database must be created before running the setup wizard. The following text explains how to set up a database using either MySQL or PostgreSQL.

MySQL

1. Log in as the root user (or any other MySQL user that has the CREATE, CREATE USER and GRANT OPTION [privileges](#)):

```
$ mysql --host=<mysql_host> --port=<port> -u <mysql_user> -p<mysql_password>
```

Note that if MySQL is installed on the same server, the "--host" parameter can be omitted. If the "--port" parameter is omitted, the default port for MySQL traffic will be used (port 3306).

The MySQL client should display a "mysql>" prompt.

2. Create a new database:

```
mysql> CREATE DATABASE <database> CHARACTER SET utf8;
```

3. Grant access permissions:

```
mysql> GRANT ALL ON <database>.* TO <user>@<ezp_host> IDENTIFIED BY  
'<password>';
```

Note that if the specified user account does not exist, it will be created.

<mysql_host>	The hostname of the MySQL database server.
<port>	The port number that will be used to connect to the MySQL database server.
<mysql_user>	The MySQL user (if no user is set up, use "root").
<mysql_password>	The password that belongs to the <mysql_user>.
<database>	The name of the database, for example "my_new_database".
<user>	The username that will be used to access the database.
<ezp_host>	The hostname of the server on which eZ Publish will be running. (may be 'localhost' if MySQL is installed on the same server).
<password>	The password you wish to set in order to limit access to the database.

PostgreSQL

1. Log in as the postgres user (or any other PostgreSQL user that has sufficient [privileges](#) to create roles and databases):

```
$ psql -h <psql_host> -p <port> -U <psql_user> -W
```

Note that if PostgreSQL is installed on the same server, the "-h" parameter can be omitted. If the "-p" parameter is omitted, the default port for PostgreSQL traffic will be used (in most cases, port 5432).

The PostgreSQL client will ask you to specify the password that belongs to the <psql_user>. If the password is correct, the client should display a "<psql_user>=#" prompt.

2. Create a new database:

```
postgres=# CREATE DATABASE <database> ENCODING='utf8';
```

3. Create a new user:

```
postgres=# CREATE USER <user> PASSWORD '<password>';
```

4. Grant access permissions:

```
postgres=# GRANT ALL PRIVILEGES ON DATABASE <database> TO <user>;
```

5. Import the "pgcrypto" module into the new database:

```
postgres=# \c <database>
<database>=# \i '<path_to_pgcrypto>'
```

<psql_host>	The hostname of the PostgreSQL database server.
<port>	The port number that will be used to connect to the PostgreSQL database server.
<psql_user>	The PostgreSQL user (if no user is set up, use "postgres").
<database>	The name of the database, for example "my_new_database".
<user>	The username that will be used to access the database.
<password>	The password you wish to set in order to limit access to the database.
<path_to_pgcrypto>	The path to the "pgcrypto.sql" file, for example "/usr/share/pgsql/contrib/pgcrypto.sql".

Note for version 9.1 of PostgreSQL users: The following changes might be necessary for these users:

```
postgres=# \c <database>
<database>=# CREATE EXTENSION pgcrypto;
```

Downloading eZ Publish

The latest stable version of eZ Publish can be downloaded from <http://share.ez.no/download-develop/downloads>.

Unpacking eZ Publish

Use your favorite tool to unpack the downloaded eZ Publish distribution to a web-served directory (a directory that is reachable using a web browser). The following example shows how to do this using the tar utility (to unpack a tar.gz file, assuming that the "tar" and the "gzip" utilities are installed on the system):

```
$ tar zxvf ezpublish-<version_number>-gpl.tar.gz -C <web_served_directory>
```

<version_number>	The version number of eZ Publish that was downloaded.
<web_served_directory>	Full path to a directory that is served by the web server. This can be the path to the document root of the web server, or a personal web-directory (usually called "public_html" or

"www", and located inside a user's home directory).

The extraction utility will unpack eZ Publish into a sub-directory called "ezpublish-<version_number>". Feel free to rename this directory to something more meaningful, for example "my_site".

Initiating the setup wizard

The setup wizard can be started using a web browser immediately after the previous steps (described in this section) are completed. It will be automatically run the first time someone tries to access/browse the index.php file located in the eZ Publish directory. Let's assume that we are using a server with the host name "www.example.com" and that after unpacking, the eZ Publish directory was renamed to "my_site".

Document root example

If eZ Publish was unpacked into a directory called "my_site" under the document root, the setup wizard can be initiated by browsing the following URL: http://www.example.com/my_site/index.php.

Home directory example

If eZ Publish was unpacked to a web-served directory located inside the home directory of a user with the user name "peter", (usually called "public_html", "www", "http", "html" or "web"), the setup wizard can be initiated by browsing the following URL: http://www.example.com/~peter/my_site/index.php.

Refer to the "The setup wizard (page 69)" section for a detailed description of the web based setup wizard.

1.3.3 Installing eZ Publish on Windows

The requirements for doing a normal installation must be met! Read the "Requirements for doing a normal installation (page 39)" section first. Proceed only if you have access to a Windows based system with Apache, PHP, MySQL or PostgreSQL already installed and running. (Do not use Apache 2.x for Windows.) As mentioned earlier, the database server may run on a different computer than the web server. This section will guide you through the following steps:

- Setting up a database (MySQL or PostgreSQL)
- Downloading eZ Publish
- Unpacking eZ Publish
- Initiating the setup wizard

Setting up a database

Known issue with running PHP5.3 on MySQL: Some Windows users with both IPv4 and IPv6 installed experience problems connecting to the database server using host names like "localhost"... If you experience problems, try using IPv4 address like "127.0.0.1". This is due to a connectivity problem when running PHP5.3 on MySQL. So, please replace the database server name "localhost" with the IP address of the machine, or "127.0.0.1", which is reserved for the local host.

A database must be created before running the setup wizard. The following text explains how to set up a database using either MySQL or PostgreSQL.

MySQL

1. Log in as the root user (or any other MySQL user that has the CREATE, CREATE USER and GRANT OPTION [privileges](#)):

```
mysql --host=<mysql_host> --port=<port> -u <mysql_user> -p<mysql_password>
```

Note that if MySQL is installed on the same server, the "--host" parameter can be omitted. If the "--port" parameter is omitted, the default port for MySQL traffic will be used (port 3306).

The MySQL client should display a "mysql>" prompt.

2. Create a new database:

```
mysql> CREATE DATABASE <database> CHARACTER SET utf8;
```

3. Grant access permissions:

```
mysql> GRANT ALL ON <database>.* TO <user>@<ezp_host> IDENTIFIED BY
'<password>';
```

 Note that if the specified user account does not exist, it will be created.

<mysql_host>	The hostname of the MySQL database server.
<port>	The port number that will be used to connect to the MySQL database server.
<mysql_user>	The MySQL user (if no user is set up, use "root").
<mysql_password>	The password that belongs to the <mysql_user>.
<database>	The name of the database, for example "my_new_database".
<user>	The username that will be used to access the database.
<ezp_host>	The hostname of the server on which eZ Publish will be running. (may be 'localhost' if MySQL is installed on the same server).
<password>	The password you wish to set in order to limit access to the database.

PostgreSQL

1. Log in as the postgres user (or any other PostgreSQL user that has sufficient [privileges](#) to create roles and databases):

```
psql -h <psql_host> -p <port> -U <psql_user> -W
```

 Note that if PostgreSQL is installed on the same server, the "-h" parameter can be omitted. If the "-p" parameter is omitted, the default port for PostgreSQL traffic will be used (in most cases, port 5432).

The PostgreSQL client will ask you to specify the password that belongs to the <psql_user>. If the password is correct, the client should display a "<psql_user>=#" prompt.

2. Create a new database:

```
postgres=# CREATE DATABASE <database> ENCODING='utf8';
```

3. Create a new user:

```
postgres=# CREATE USER <user> PASSWORD '<password>';
```

4. Grant access permissions:

```
postgres=# GRANT ALL PRIVILEGES ON DATABASE <database> TO <user>;
```

5. Import the "pgcrypto" module into the new database:

```
postgres=# \c <database>
<database>=# \i '<path_to_pgcrypto>'
```

<psql_host>	The hostname of the PostgreSQL database server.
<port>	The port number that will be used to connect to the PostgreSQL database server.
<psql_user>	The PostgreSQL user (if no user is set up, use "postgres").
<database>	The name of the database, for example "my_new_database".
<user>	The username that will be used to access the database.
<password>	The password you wish to set in order to limit access to the database.
<path_to_pgcrypto>	The path to the "pgcrypto.sql" file, for example "C:\\Program Files\\PostgreSQL\\8.2\\share\\contrib\\pgcrypto.sql".

Downloading eZ Publish

The latest stable version of eZ Publish can be downloaded from <http://share.ez.no/download-develop/downloads>. Windows users should download the ".zip" archive.

Unpacking eZ Publish

Use your favorite utility to unpack the downloaded eZ Publish archive to a web-served directory (a directory that is reachable using a web browser). The extraction utility will unpack eZ Publish into a subdirectory called "ezpublish-4.x.y". Feel free to rename this directory to something more meaningful, for example "my_site".

Initiating the setup wizard

The setup wizard can be started using a web browser immediately after the previous steps (described in this section) are completed. It will be automatically run the first time someone tries to access/browse the index.php file located in the eZ Publish directory. Let's assume that we are using a server with the hostname "www.example.com" and that after unpacking, the eZ Publish directory was renamed to "my_site".

Document root example

If eZ Publish was unpacked into a directory called "my_site" under the document root, the setup wizard can be initiated by browsing the following URL: http://www.example.com/my_site/index.php.

Refer to "The setup wizard (page 69)" section for a detailed description of the web based setup wizard.

1.4 Manual installation

This installation method is for advanced users who know what they are doing, all other users should use the "Normal installation method" (page 38). The manual installation method requires an environment which already has a web server, a database and etc. setup and ready to go; eZ Publish needs to be downloaded and unpacked. Instead of running the setup wizard, all configuration is done manually using the command line interface of the target operating system. The following sections (depending on the target OS) will take you through the necessary steps.

1.4.1 Requirements for doing a manual installation

The requirements for doing a manual installation are the same as for the normal installation. Please refer to the "Requirements for doing a normal installation" (page [39](#)) section.

1.4.2 Manual installation on a Linux/UNIX based system

The requirements for doing a manual installation must be met. Please read the requirements page (page [39](#)) if you're not sure about the requirements. Proceed only if you have access to a UNIX based environment with Apache, PHP, MySQL or PostgreSQL already installed and running. As mentioned earlier, the database server may run on a different computer than the web server. A manual installation consists of the following steps:

- Setting up a database (MySQL or PostgreSQL)
- Downloading eZ Publish
- Unpacking eZ Publish
- Manual configuration of eZ Publish

The only difference between a normal and a manual installation is the last step. Instead of running the web based setup wizard, eZ Publish is manually configured by editing a couple of files. The first three steps are explained under the "Installing eZ Publish on a Linux/UNIX based system" (page [44](#)) section. The last step is explained under the "Manual configuration of eZ Publish" (page [52](#)) section.

1.4.3 Manual installation on Windows

The requirements for doing a manual installation must be met. Please read the requirements page if you're not sure about the requirements. Proceed only if you have access to a Windows based system with Apache, PHP, MySQL or PostgreSQL already installed and running. As mentioned earlier, the database server may run on a different computer than the web server. A manual installation consists of the following steps:

- Setting up a MySQL database
- Downloading eZ Publish
- Unpacking eZ Publish
- Manual configuration of eZ Publish

The only difference between a normal and a manual installation is the last step. Instead of running the web based setup wizard, eZ Publish is manually configured by editing a couple of files. The first three steps are explained under the "Installing eZ Publish on Windows" (page [48](#)) section. The last step is explained under the "Manual configuration of eZ Publish" (page [52](#)) section.

1.4.4 Manual configuration of eZ Publish

This section describes how to manually configure eZ Publish instead of using the setup wizard to do all the work. Keep in mind that the manual installation method is for expert users only. It should only be used by people who know what they are doing. The following steps will work on both Linux/UNIX and Windows environments.

Database initialization

A clean eZ Publish database is created using two very important SQL scripts: "kernel_schema" and "cleandata" (note that an empty database should be created before launching these scripts). The first of them initializes the necessary database structure and the second one imports the pre-defined data to the database. While the "kernel_schema" script differs for each database engine, the "cleandata" script is the same for all solutions.

MySQL

Use the following command to run the MySQL specific "kernel_schema" script:

```
$ mysql -u USERNAME -pPASSWORD DATABASE < PATH/kernel/sql/mysql/
kernel_schema.sql
```

In eZ Publish 4.0.1 and later versions, the script will use the InnoDB storage engine when creating new tables. This storage engine is recommended (and will be required in the future) for running eZ Publish on a MySQL database. Contact your database administrator if you are unsure about whether InnoDB is available on your server.

In eZ Publish 4.0.0, the CREATE TABLE statements in the "kernel_schema" script do not specify which storage engine to use (no ENGINE or TYPE option), and thus the default storage engine will be used. Normally, it is MyISAM. Because of this, it is highly recommended to set the default storage engine to InnoDB before you run the "kernel_schema" script (refer to the [MySQL documentation](#) for information about how to set the default engine). Alternatively, you can run the "kernel_schema" script first and then convert the newly created tables to InnoDB. You can either use the "bin/php/ezconvertmysqltabletype.php" script for database conversion (recommended) or convert the tables individually by using the following SQL query for each table:

```
ALTER TABLE <name_of_table> TYPE = innodb;
```

Use the following command to run the generic "cleandata" script:

```
$ mysql -u USERNAME -pPASSWORD DATABASE < PATH/kernel/sql/common/cleandata.sql
```

USERNAME	The MySQL user (if no user is set up, use
----------	---

	"root").
PASSWORD	The password that belongs to the username.
DATABASE	The name of the database, for example "my_database".
PATH	The full path to the root directory of your eZ Publish installation, for example "/opt/ezp".

File permissions

Windows users can skip this part. If eZ Publish is installed on a Linux/UNIX based system, some of the file permissions need to be changed. There is a shell script that takes care of this. This script must be run, otherwise eZ Publish will not function properly. The script needs to be run from within the eZ Publish directory:

```
$ cd /opt/ezp
$ bin/modfix.sh
```

Replace "/opt/ezp" with the full path to the root directory of your eZ Publish installation.

The modfix script recursively alters the permission settings of the following directories inside the eZ Publish installation:

- var/*
- settings/*
- design/*
- autoload/*

If you know the user and group of the web server it is recommended to use a different set of permissions:

```
# chown -R user.usergroup var/ settings/ design/ autoload/
# chmod -R 770 var/ settings/ design/ autoload/
```

The "user.usergroup" notation must be changed to user and group name that the web server runs as.

Configuring eZ Publish

The "site.ini.append.php" configuration file located in the "settings/override" directory of your eZ Publish installation must be changed, or else eZ Publish will not function properly. This file is the global override for the site.ini configuration file. There are a lot of things that need to be configured (database, mail transport system, var directory, etc.). The following text shows a generic example of a configuration that can be used:

```
<?php /* #?ini charset="utf-8"?

[DatabaseSettings]
DatabaseImplementation=eZmysql
Server=localhost
User=root
Password=
Database=my_database

[FileSettings]
VarDir=var/example

[Session]
SessionNameHandler=custom

[SiteSettings]
DefaultAccess=example
SiteList []
SiteList []=example

[SiteAccessSettings]
CheckValidity=false
AvailableSiteAccessList []
AvailableSiteAccessList []=example
AvailableSiteAccessList []=example_admin
RelatedSiteAccessList []
RelatedSiteAccessList []=example
RelatedSiteAccessList []=example_admin
MatchOrder=host;uri

# Host matching
HostMatchMapItems []=www.example.com;example
HostMatchMapItems []=admin.example.com;example_admin

[InformationCollectionSettings]
EmailReceiver=webmaster@example.com

[MailSettings]
Transport=sendmail
AdminEmail=webmaster@example.com
EmailSender=test@example.com

[RegionalSettings]
Locale=eng-GB
ContentObjectLocale=eng-GB
TextTranslation=disabled
```

```
*/ ?>
```

In the example above the "AvailableSiteAccessList[]" array located in the "[SiteAccessSettings]" section of this file determines the available siteaccesses called "example" and "example_admin". The "CheckValidity" setting located in the same section should be set to false, otherwise the setup wizard will be initiated when trying to access the site.

In addition, two siteaccess configurations must be created, a public siteaccess ("example") and an administration siteaccess ("example_admin"). The following sub-directories have to be created in the root of your eZ Publish installation:

- settings/siteaccess/example
- settings/siteaccess/example_admin

Both siteaccesses must have a file called "site.ini.append.php".

The public siteaccess

The following text shows a generic solution for the "example" siteaccess:

```
<?php /* #?ini charset="utf-8"?

[SiteSettings]
SiteName=Example
SiteURL=www.example.com
LoginPage=embedded

[SiteAccessSettings]
RequireUserLogin=false
ShowHiddenNodes=false

[DesignSettings]
SiteDesign=example

[ContentSettings]
ViewCaching=disabled

[TemplateSettings]
TemplateCache=disabled
TemplateCompile=disabled
#ShowXHTMLCode=enabled
#Debug=enabled

[DebugSettings]
```

```
DebugOutput=enabled
Debug=inline
#DebugRedirection=enabled

[RegionalSettings]
SiteLanguageList[]
SiteLanguageList[]=eng-GB
ShowUntranslatedObjects=disabled

*/ ?>
```

The admin siteaccess

The following text shows a generic solution for the "example_admin" siteaccess:

```
<?php /* #?ini charset="utf-8"?

[SiteSettings]
SiteName=Example
SiteURL=admin.example.com
LoginPage=custom

[SiteAccessSettings]
RequireUserLogin=true
ShowHiddenNodes=true

[DesignSettings]
SiteDesign=admin

[ContentSettings]
CachedViewPreferences[full]=admin_navigation_content=0;
admin_navigation_details=0;admin_navigation_languages=0;
admin_navigation_locations=
0;admin_navigation_relations=0;admin_navigation_roles=0;
admin_navigation_policies=0;admin_navigation_content=0;
admin_navigation_translatio
ns=0;admin_children_viewmode=list;admin_list_limit=1;
admin_edit_show_locations=0;admin_url_list_limit=10;admin_url_view_limit=10;
admin_sec
tion_list_limit=1;admin_orderlist_sortfield=user_name;
admin_orderlist_sortorder=desc;admin_search_stats_limit=1;admin_treemenu=1;
admin_boo
kmarkmenu=1;admin_left_menu_width=13

[DebugSettings]
```

```
DebugOutput=disabled
Debug=inline

[RegionalSettings]
SiteLanguageList []
SiteLanguageList []=eng-GB
ShowUntranslatedObjects=enabled

*/ ?>
```

Note that database settings, mail settings, regional and other settings defined in "settings/override/site.ini.append.php" will be used for each siteaccess regardless of what is specified in the siteaccess settings. In the example above, the "Database=my_database" is specified under the "[DatabaseSettings]" section of this file so this database will be used for both "example" and "example_admin" siteaccesses. Refer to the "Site management" and "Configuration" sections of the "Concepts and basics" chapter for more information.

Unicode support

There is no need to override the "i18n.ini" configuration file since Unicode support is enabled by default in eZ Publish 4.

Languages

Available languages and their priorities can be controlled per siteaccess using the "SiteLanguageList" configuration setting located under the "[RegionalSettings]" section of the siteaccess "site.ini.append.php" file. If this setting is not specified, the system will use the old "ContentObjectLocale" setting and thus only the default language will be shown. Refer to the "Configuring the site languages" section for more information and examples.

The "cleandata.sql" script creates only one language which is the British English (eng-GB). All other languages should be added using the "Setup - Languages" part of the administration interface (<http://admin.example.com> in the example above).

Dynamic tree menu

If you have a large site with many nodes, it is strongly recommended to enable the "Dynamic" switch for your administration siteaccess. This will make the left tree menu in the administration interface work much faster and decrease the usage of network bandwidth.

Administrator's log-in and password

The following user name and password are set by the "cleandata.sql" script and can be used for logging in to the administration interface.

- User name: admin
- Password: publish

It is strongly recommended to change this password as soon as possible. Note that if you need another user name for site administrator, you can create a new administrator user, log in as this user and remove the old one.

1.5 Automated installation

The automated installation method (also known as "kick-start") is for experienced users. It provides an automated version of the "Normal installation method (page 38)" and is designed for system administrators who wish to roll out pre-configured installations of eZ Publish. This method requires minimum interaction with the web-based setup wizard and thus it can be used to rapidly deploy eZ Publish on a massive scale. This method has the same requirements as the "Normal installation" method. A typical automated installation process consists of the following steps:

- Setting up / creating a database
- Downloading a packaged eZ Publish distribution
- Unpacking the eZ Publish distribution
- Configuring the "kickstart.ini" file
- Initiating the web based setup wizard

Once the web-based setup wizard has completed, eZ Publish will be ready for use.

1.5.1 Requirements for doing an automated installation

The requirements for an automated installation are the same as for the normal installation method. Please refer to the "Requirements for doing a normal installation" page for more information.

At the minimum, a web server, a PHP engine, and a database server must be installed. Additional server-side software is only necessary if the kick-start configuration file instructs the system to make use of such software. For example, "ImageMagick" has to be available if it has been specified as the primary image manipulation solution.

New requirements since version 4.5

In order to do an offline installation with the use of kickstart.ini, all required files must be downloaded from <http://packages.ez.no/ezpublish/> and extracted to var/storage/packages/eZ-systems

ezwebin:

 index.xml ezwebin_site.ezpkg ezwt_extension.ezpkg ezgmaplocation_extension.ezpkg
ezstarrating_extension.ezpkg ezwebin_classes.ezpkg ezwebin_democontent.ezpkg ezwebin_
design_blue.ezpkg ezwebin_design_cleangray.ezpkg ezwebin_design_gray.ezpkg ezwebin_
extension.ezpkg ezwebin_banners.ezpkg

ezflow:

 index.xml ezflow_site.ezpkg ezwt_extension.ezpkg ezgmaplocation_extension.ezpkg
ezstarrating_extension.ezpkg ezwebin_extension.ezpkg ezflow_classes.ezpkg ezflow_
democontent.ezpkg ezflow_design.ezpkg ezflow_extension.ezpkg

The following extraction method is used for each .ezpkg:

1. Create a new directory with the same name, except for the .ezpkg suffix
2. Extract the .ezpkg file into it's directory with "tar -xzf"

Note that the index.xml file should be stored to var/storage/packages/eZ-systems

The next section (page 65) explains how eZ Publish can be configured to do an automated installation of itself.

1.5.2 Automated installation of eZ Publish

The requirements for doing an automated installation must be met. Please read the previous section if you're not sure about the requirements. This section will guide you through the following steps:

- Setting up a database (MySQL or PostgreSQL)
- Downloading eZ Publish
- Unpacking eZ Publish
- Configuring the kickstart system
- Starting the installation by initiating the web based setup wizard

Depending on the target system, please refer to either "Installing eZ Publish on a Linux/UNIX based system" or "Installing eZ Publish on Windows" for information about the first three steps (database setup, download and unpacking). The rest of the steps are explained below.

Configuring the kickstart system

The behavior of the automated installation is controlled by the "kickstart.ini" configuration file. This file makes it possible to specify parameters for each installation step of the web based setup wizard. For example, by providing the database connection parameters, the corresponding setup wizard step will have the input forms pre-filled. It is also possible to instruct the wizard to skip certain steps.

Initialization

Create a copy of the "kickstart.ini-dist" file (located in the root of your eZ Publish installation) and make sure that the copy is named "kickstart.ini" (located in the root of eZ Publish).

The following example shows how this can be done on a Linux/UNIX based system:

1. Navigate into the eZ Publish directory:

```
$ cd /path/to/ezpublish/
```

2. Copy and rename the configuration file:

```
$ cp kickstart.ini-dist kickstart.ini
```

Security issues

The web server must have read access to the "kickstart.ini" file during the installation process. This might become a security problem at a later stage if the file contains usernames, passwords, etc. To prevent this from happening, it is recommended to do one of the following:

- Remove the file when the installation has completed.
- Use rewrite rules to make sure that it is not readable from outside.

Configuration blocks

The "kickstart.ini" file contains a configuration block for every step of the setup wizard. The block names are encapsulated by square brackets. The following list shows an overview of the available blocks.

- [email_settings]
- [database_choice]
- [database_init]
- [language_options]
- [site_types]
- [site_access]
- [site_details]
- [site_admin]
- [security]
- [registration]

In the default kickstart file, everything is commented out. The blocks and the corresponding settings have to be uncommented in order to take effect. This can be done by removing the hash ("#") characters from the start of the lines that you should be activated. Make sure that there are no leading whitespace characters at the start of the lines.

Configuration parameters

Each parameter takes a text string as an input value. Some parameters are able to handle an array of strings. The following examples demonstrate the two parameter types.

- Single parameter:

```
Server=www.example.com
```

- Array parameter:

```
Title[]
Title[news]=The news site
Title[forums]=The forum site
```

Documentation and examples

The "kickstart.ini" file contains documentation in the file itself. Please refer to the embedded instructions and examples for a detailed explanation of the steps. The following table shows how the examples / inline instructions deal with required and optional parameters.

Syntax	Description
<value>	Angle brackets indicate that the parameter value is required, example: #Server=<hostname>
[value]	Squared brackets indicate that the parameter value is optional, example: #FirstName=[string]

A parameter will only take effect if it has been uncommented. Remove the leading hash ("#") and make sure that there are no whitespace characters at the start of the lines that include the uncommented parameters.

Skipping steps

A step can be skipped by uncommenting and setting the "Continue" parameter to "true". This parameter can be used for each step / block. The following table shows the outcome for the different configurations of the "Continue" parameter.

Assignment	Result
Continue=false	The step will be shown and the input values will be pre-filled with the values (if any) defined in the "kickstart.ini" configuration file. This is the same as when the "Continue" parameter is missing or if it has been commented out.
Continue=true	The system will automatically use the values defined in the kickstart file and thus the step will not be shown. However, if something goes wrong (missing or wrong values, etc.), the step will be shown.

Starting the installation

The installation can be started by initiating the web based setup wizard. Please refer to the "Initiating the setup wizard" part of the "Normal installation" section.

Share your information

1.6 The setup wizard

This section contains a comprehensive guide through the web-based setup wizard of eZ Publish. The setup wizard is designed to ease the initial configuration of the system. It can be started using a web browser when the necessary installation steps (described in the previous sections) are completed. The setup wizard will automatically start the first time the "index.php" file (located in the root of the eZ Publish directory) is accessed/browsed.

The setup wizard does not store or modify any data before the final step; thus, it can be safely restarted by reloading the URL containing only the "index.php" part. The back button (located at the bottom) can be used to jump back to previous steps in order to modify settings. A typical setup cycle consists of 12 steps:

1. Welcome page
2. System check
3. Outgoing E-mail
4. Database choice (optional)
5. Database initialization
6. Language support
7. Site selection
8. Access method
9. Site details
10. Site security
11. Site registration
12. Finish

Note that some of the steps will be omitted when an eZ Publish bundle is being installed.

Welcome page

(see figure 1.1)

This is the initial page of the setup wizard. This step allows the user to select which language that will be used during the installation process. In addition, the wizard also checks the system configuration and displays a note if it is not optimal (in this case, an additional button called "Finetune" will be available at the bottom of the page).

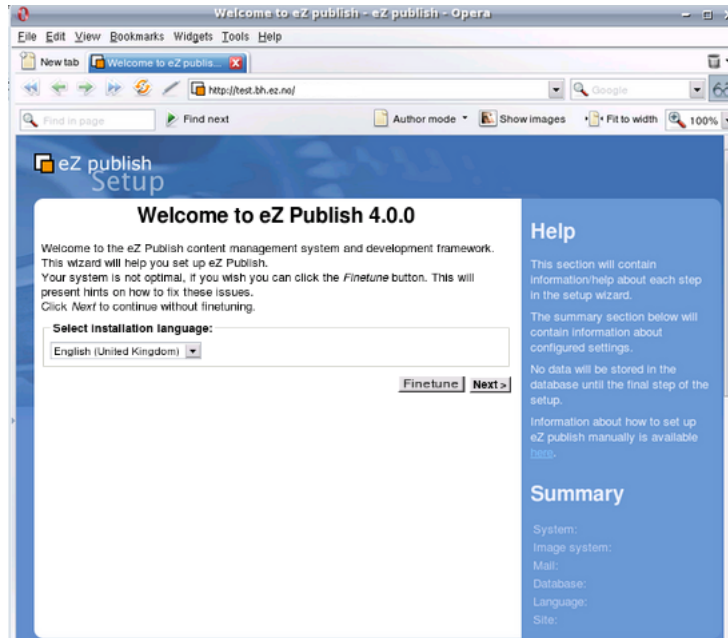


Figure 1.1: Step 1: Welcome page

The system automatically pre-selects one of the languages according to your browser's language settings. You can choose another language by selecting the desired language using the drop-down list. (The list of available languages is built using the INI files located in the "share/locale" directory).

After you click "Finetune" (if available), the wizard will load the "System finetuning" page, which contains information about configuration issues. The following screenshot shows an example of this page.

(see figure 1.2)

After you click "Next", the wizard will either load the "System check" page (if some critical issues need to be fixed) or the "Outgoing E-mail" page (if everything is okay).

System check

(see figure 1.3)

This page usually appears if critical issues/problems are detected. The setup wizard will display information about the issues that need to be fixed and suggestions describing how they can be fixed.

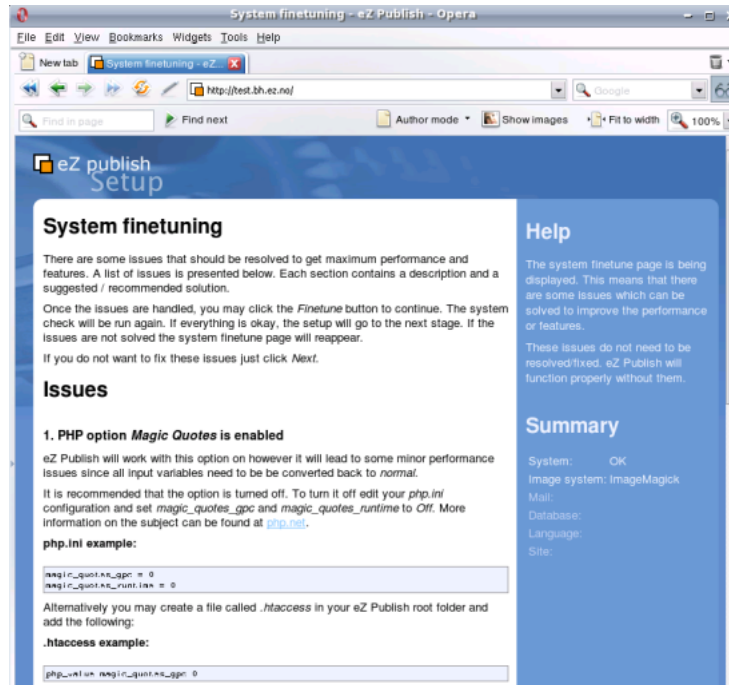


Figure 1.2: System finetuning

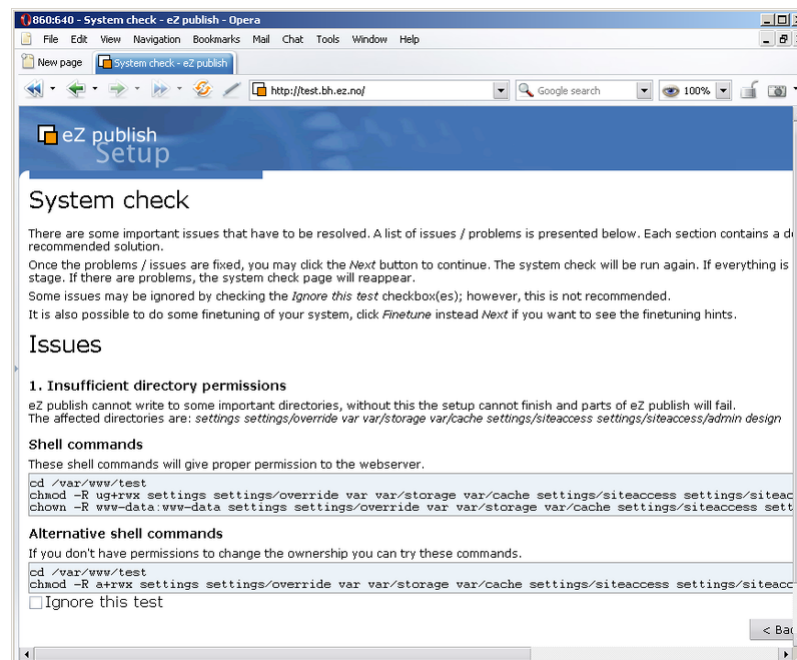


Figure 1.3: Step 2: Issues

Issues

There may be several issues/problems. A suggestion to each problem is presented below the description of the problem itself. The setup wizard will probably suggest the execution of miscellaneous shell commands (in order to fix ownerships, permissions, etc.). These commands must be executed using a system shell. Simply copy the commands from the browser window and paste them into an open shell. The setup wizard will run the system check again when the "Next" button is clicked. The "System check" page will keep reappearing until all issues have been fixed (or ignored, see the next section). Once everything is okay, the setup wizard will display the next step.

Ignoring tests

Some issues/problems may be ignored using a check-box labeled "Ignore this test". However, it is recommended to fix all issues rather than ignoring them.

Outgoing E-mail

(see figure 1.4)

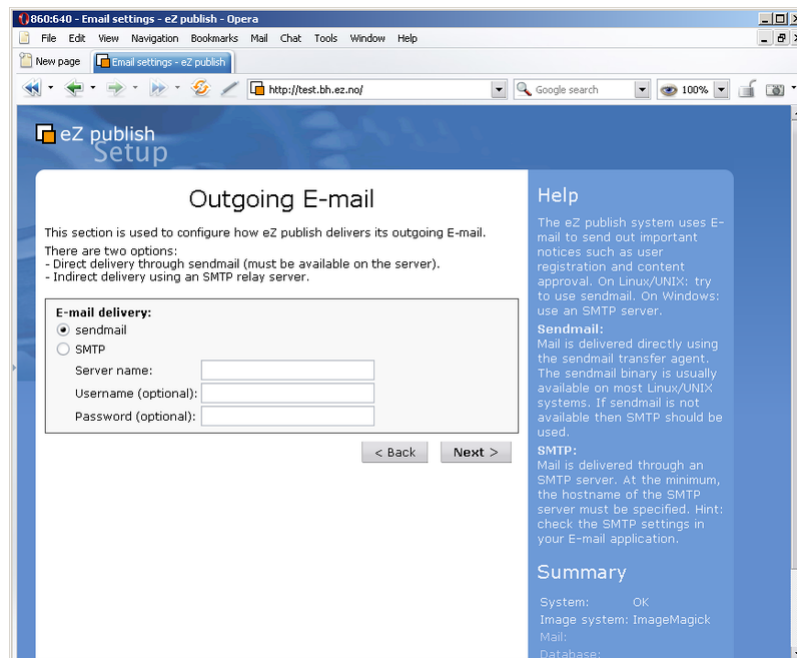


Figure 1.4: Step 3: Outgoing E-mail

eZ Publish uses E-mail to send out miscellaneous notices. This step is used to configure how eZ Publish delivers outgoing E-mail. There are two options:

- Direct delivery through sendmail (must be available on the server)
- Indirect delivery using an SMTP (Simple Mail Transfer Protocol) relay server

On Linux/UNIX: try to use sendmail; use SMTP if sendmail is unavailable. On Windows: use the SMTP setting.

Sendmail

Mail is delivered directly using the sendmail transfer agent. The agent must be running on the same host as the webserver is running on. The sendmail binary is usually available on most Linux/UNIX systems. If sendmail is not available then SMTP should be used.

SMTP

Mail is delivered through an SMTP server. At the minimum, the hostname of the SMTP server must be specified.

Database type

(see figure 1.5)

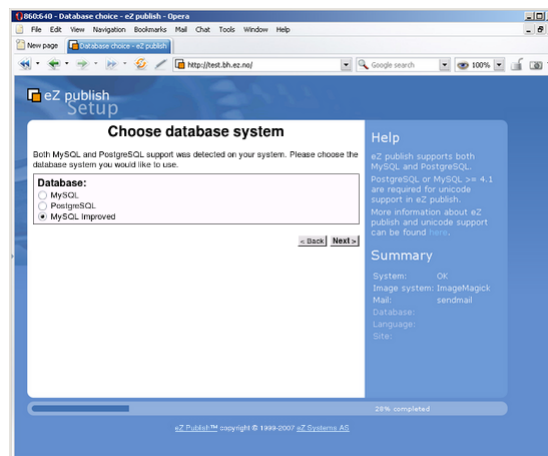


Figure 1.5: Step 4: Database choice

The setup will automatically detect database support that has been made available for the PHP scripting engine. If both MySQL and PostgreSQL are supported, the database choice dialog will appear. If PHP is setup only to support one type of database, eZ Publish will automatically use it and thus the database choice dialog will not be displayed.

Note that if the [MySQLi extension](#) is enabled in PHP, the "MySQL Improved" option will be available on the list. If you are going to use a MySQL database, it is recommended to select "MySQL Improved" instead of "MySQL".

Database initialization

(see figure 1.6)

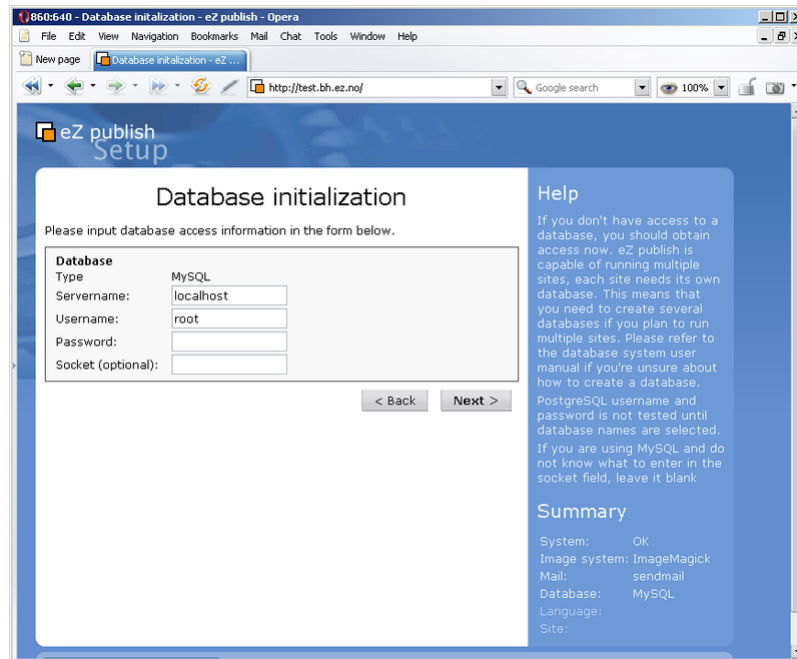


Figure 1.6: Step 5: Database initialization

Information about the host name of the server running the database engine, and a user name/password combination needs to be provided. After you click "Next", if MySQL or MySQL Improved are used, the setup wizard will attempt to connect to the database. The setup will only continue if it is able to connect to the specified MySQL server with the specified user name/password combination. PostgreSQL parameters are tested at a later stage during the setup wizard. (Note that even if the eZ Publish Extension for Oracle Database is installed, the setup wizard will not let you use an Oracle database. The configuration must be done manually as described in the documentation of the database extension.)

Known issue with running PHP5.3 on MySQL: Some people (like Windows users with both IPv4 and IPv6 installed) experience problems connecting to the database server using host names like "localhost"... If you experience problems, try using IPv4 address like "127.0.0.1". This is due to a connectivity problem when running PHP5.3 on MySQL. So, please replace the database server name "localhost" with the IP address of the machine, or "127.0.0.1", which is reserved for the local host.

Language support

(see figure 1.7)

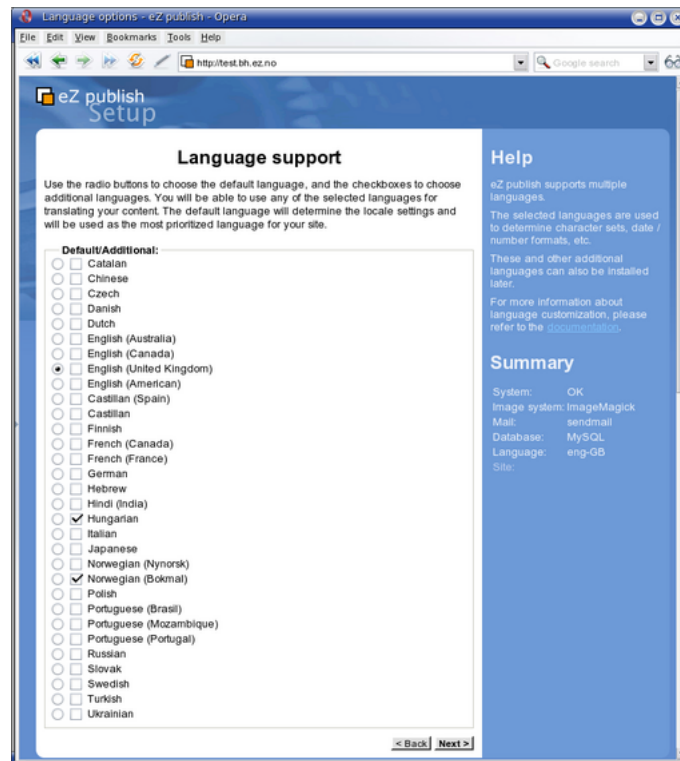


Figure 1.7: Step 6: Language support

This step allows the user to choose a language configuration for the site that is being installed. The setup wizard automatically pre-selects one of the languages according to your browser's language settings. Use the radio buttons to choose the default language (required), and the check-boxes to choose the additional languages (optional). All the selected languages will be added to the system and put on the list of prioritized languages. You will be able to use any of these languages for creating and translating your content after the setup wizard is finished.

Note that choosing the default language at this step will determine default language, system locale and the most prioritized language for your site. If you select for example "German" as default language, then both locale and default language will be set to "ger-DE", your administration interface will be translated into German, and this language will be recorded as the most prioritized one for your site. Languages can be reconfigured at any time (even when a site is up and running) using the administration interface.

Note that regardless of the selected language configuration, the site will be created using UTF-8 as the character set.

Site selection

(see figure 1.8)



Figure 1.8: Step 7: Site selection

This step allows the user to select one of the standard site packages. These packages are intended to provide basic examples mostly for the purpose of demonstration and learning. However, it is possible to use them as a basic framework which you can extend/tweak in order to make it suitable for a specific purpose. A demo site usually contains some artwork (images), CSS code, actual content and template files. The plain type should be used when starting from scratch.

The setup wizard automatically fetches the list of available site packages from remote and internal repositories and asks the user to choose one. The default remote repository is <http://packages.ez.no/ezpublish/4.0>. Note that it only contains the following three site packages:

- Plain site
- Website Interface
- eZ Flow

Older site packages such as "News site", "Shop site" and "Gallery site" are currently not available for eZ Publish 4.

The wizard will automatically download the selected site package and all its dependent packages, import them to the system and display a list of successfully imported packages as shown in the following screenshot. (This step will be omitted if all these packages are already stored under internal repositories.)

Note: You may download and extract these packages and required files for offline installation. Read the requirements for such an offline installation by clicking this link.

(see figure 1.9)

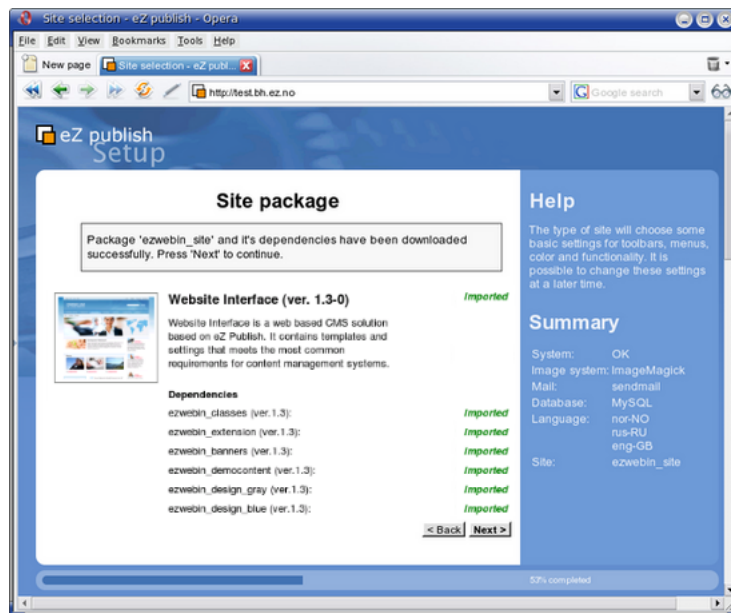


Figure 1.9: The list of imported packages

All dependent packages except for the site style package will be automatically installed.

Package language options

(see figure 1.10)

If the language configuration selected at the "Language support" step doesn't match the languages used in the packages being installed, the "Package language options" interface will appear as shown on the screenshot above. For example, the "Website interface" site package makes it possible to have demo content created in 2 languages: English (United Kingdom) and French. If the same languages are selected at the "Language support" step, the packages will be installed silently. Otherwise, the user will need to specify how the system should act towards the "superfluous" languages (i.e. languages that exist in the package but aren't present in the selected language configuration for the site). Possible actions are:

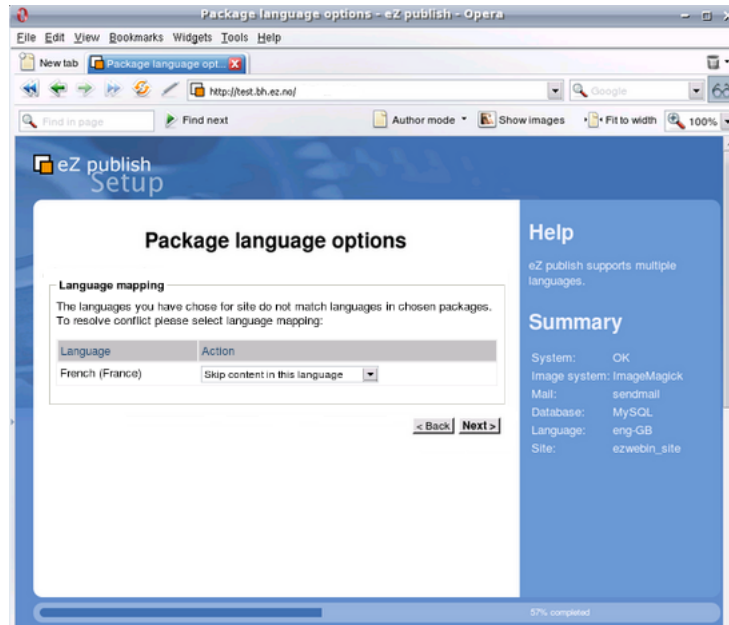


Figure 1.10: *Package language options*

- Skip content in this language
- Create language (extend the language configuration of the site and create demo content in this language)
- Map to another language (use demo data to create content in another language)

Dealing with possible problems

If the web server is not able to contact the remote repository (due to firewall rules for example), the setup wizard will display an error message at the "Site selection" step. To fix this, allow outbound connections to `http://packages.ez.no` in your firewall (port 80) or download the packages manually.

Outbound connections via proxy

If you allow only outbound connections via a proxy server, then you need to configure eZ Publish in the following way:

1. Create a file called "site.ini.append.php" in the "settings/override" directory and make sure it contains the following lines:

```
[ProxySettings]
ProxyServer=proxy.example.com:3128
User=myuser
Password=secret
```

Replace "proxy.example.com:3128" with the actual address and port number that can be used to access the web through the proxy server. If the proxy server requires authentication, you will also need to provide a valid username/password combination.

2. Restart the setup wizard.

Note that [CURL](#) support must be enabled in PHP, otherwise outbound connections via proxy will not work.

Manual download of packages

If the wizard fails to connect the external packages repository, you can manually download the desired site package and all the dependent packages it requires and then upload/import them via the setup wizard. The following instructions reveal how this can be done.

1. Go to the [packages download page](#). The "Sites" section of this page contains the list of available site packages including the following information for each of them:
 - Name
 - Description
 - Dependencies (if any)

Click on the name of the desired site package to download it. (A package is downloaded as an ".ezpkg" file.)

2. Download all the dependent packages required by this site package (these are listed under "Dependencies"). You can download a package by clicking on its name. The packages are downloaded as ".ezpkg" files.
3. Use the package import interface located at the bottom of the page in the setup wizard to upload/import the downloaded site package (click the "Choose" button, select the downloaded ".ezpkg" file that contains the site package and click the "Upload" button). The imported site package will appear on the list.
4. Upload/import all the dependent packages using the same import interface.

Note: it is also possible to download packages manually from the remote repository. The following instructions reveal how this can be done.

1. Go to the [packages repository](#), find the desired site package and download it manually. (A package is downloaded as an ".ezpkg" file.)

- Unpack the ".ezpkg" file into a temporary folder and view the "package.xml" file in order to figure out which dependent packages are required (these are listed between the <dependencies> and </dependencies> XML tags as described here). Download all the dependent packages that are required.

Additional functionality

In eZ Publish 3.7 and earlier versions, the setup wizard included one more step called "Site functionality" that allowed to select additional features that should be installed. This step is no longer used. Additional functionality can be added after the setup wizard is finished by downloading the desired packages from the "Content objects" section of the [packages download page](#), importing the packages and installing them.

Access method

(see figure 1.11)

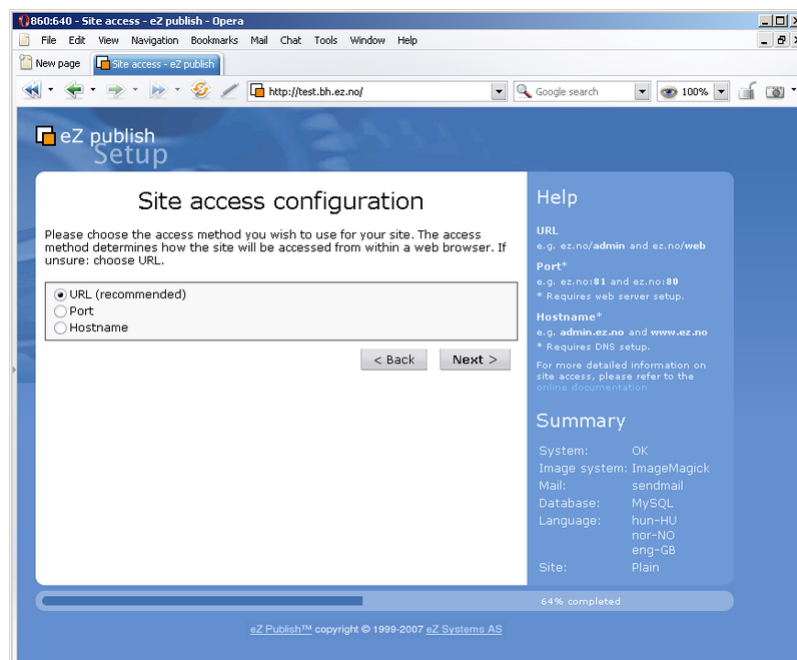


Figure 1.11: Step 8: Site access configuration

This step allows the configuration of the access method that should be used when eZ Publish receives a request. There are three options:

- URL
- Port

- Hostname

URL

When the URL access method is used, eZ Publish selects the site that should be accessed based on the contents of the URL (in particular the part that comes right after "index.php"). This is the default and most generic option. It doesn't require any additional configuration. Use this setting when installing eZ Publish for the first time.

Port

When the port access method is used, eZ Publish selects the site that should be accessed based on a port number that is specified in the URL. The port number must be appended to the host name of the web server: "http://www.example.com:81/index.php". This option requires additional web server and firewall configuration. Use this setting only if you know what you're doing.

Host name

When this access method is used, each site is assigned a unique host name. For example, "www.example.com" and "admin.example.com" can be assigned to the public and the administration interface respectively. This option requires additional web and DNS server configuration. Use this setting only if you know what you're doing.

Site details

(see figure 1.12)

 This step allows the modification of settings related to the site that is being installed. Note that the "User path" and "Admin path" access values depend on which access method you choose. When the port access method is used these values are port numbers. If you use the URL access method then "User path" and "Admin path" should only contain letters, digits and underscores. If the host name access method is used then some additional symbols like dashes, dots and colons are allowed whereas underscores aren't.

The available databases will be displayed in the database drop-down menu. The "Refresh" button can be used to update the list (if a database is being created at this point). It is required that the database uses UTF-8 as character set.

If the selected database already contains data, the "Site Details" page will reappear and ask what to do. Possible actions are:

- Leave the data and add new
- Remove existing data
- Leave the data and do nothing

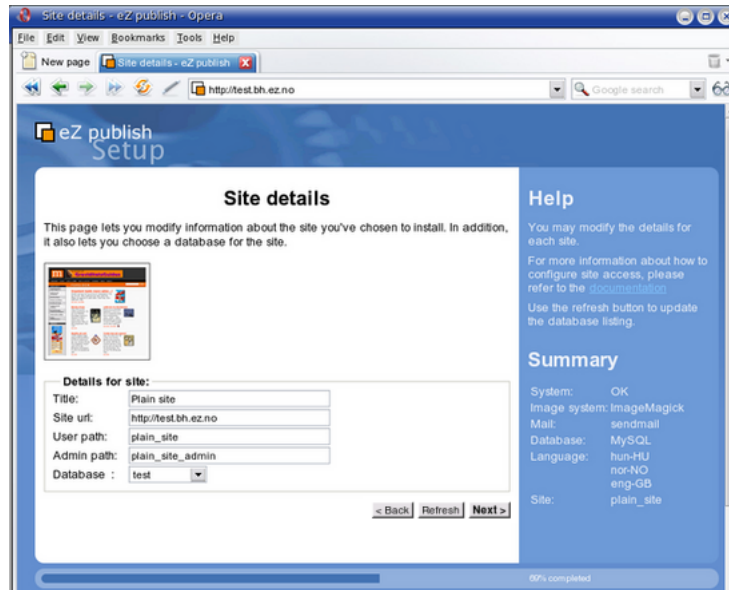


Figure 1.12: Step 9: Site details

- I've chosen a new database

Use the last option if another database has been chosen.

Site security

(see figure 1.13)

This step suggests some basic modifications that should be carried out in order to secure the site being installed. The suggested security tweak protects the configuration files from unwanted access. Don't worry about this unless you're setting up a site for public use.

Note that the administrator's user name (log-in) is set to "admin" by default and can not be changed. If you need another user name for site administrator, you can install eZ Publish, create a new administrator user, log in as this user and remove the old one.

Site registration

(see figure 1.14)

This step allows you to control whether the setup should send an information E-mail to eZ Systems or not. The information will be used internally for statistics and for improving eZ Publish. No confidential data will be transmitted and eZ Systems will not misuse or sell these details. The following information will be sent:

- System details (OS type, etc)

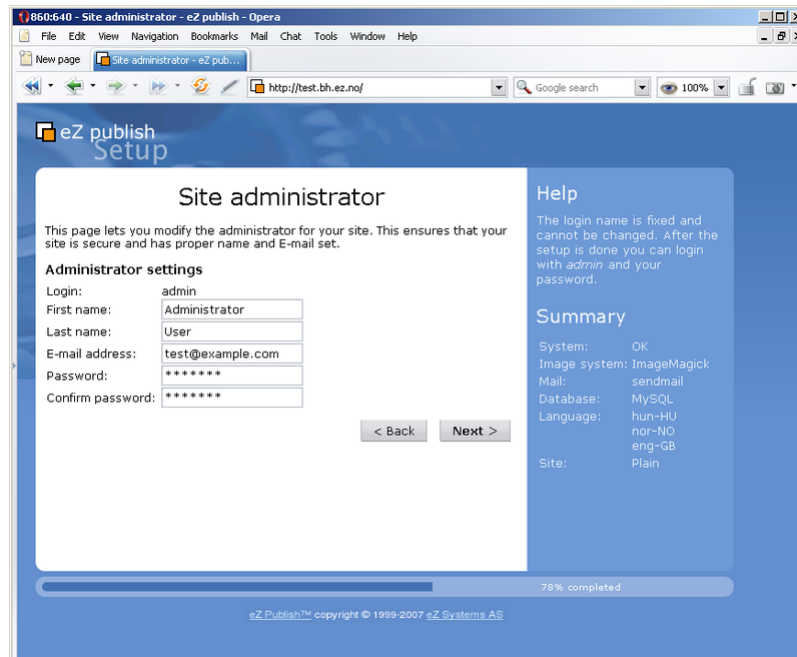


Figure 1.13: Step 10: Site administrator

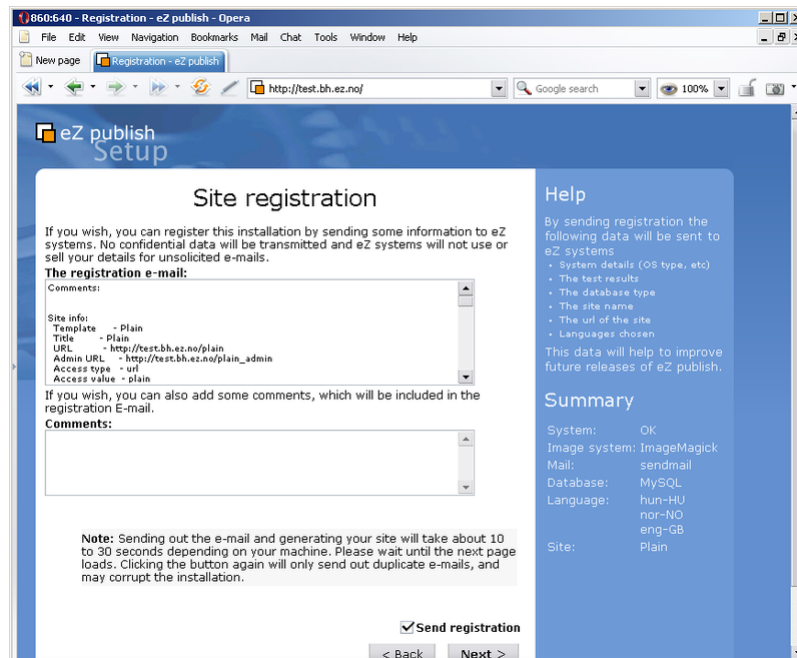


Figure 1.14: Step 11: Site registration

- The test results

- The type of database that is being used
- The name of the site
- The URL of the site
- The languages that were chosen

Finished

(see figure 1.15)

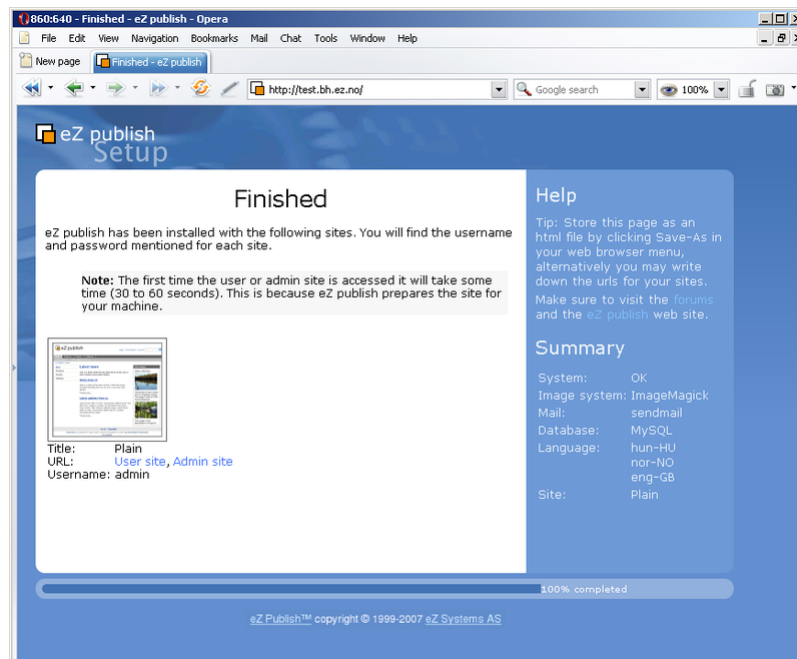


Figure 1.15: Step 12: Finished

The setup wizard has finished, eZ Publish is ready for use. Click on one of the links to access the various interfaces (public site, administration interface, etc.).

Note that it is possible to restart the installation wizard after its successful finishing by specifying "CheckValidity=true" in the "settings/override/site.ini.append.php" file so that the setup wizard will be initiated when trying to access the site.

1.7 Removing eZ Publish

This section describes how to completely remove an eZ Publish installation from a system.

Removing eZ Publish is done in four steps:

1. Deleting the eZ Publish directory
2. Removing the database
3. Reconfiguring Apache (optional)
4. Removing the cronjobs (optional)

WARNING! By following these steps, you will remove both eZ Publish and all the data/content that you have put into the system. Everything will be lost.

Deleting the eZ Publish directory

Remove the eZ Publish directory using your favorite tool.

Linux/UNIX

On Linux/UNIX systems, the removal would most likely be carried out using the "rm" command:

```
$ rm -Rf /path/to/ez_publish
```

Please note that some file/directory permissions might be messed up. If this is the case, it will prevent a regular user from removing all eZ Publish files. You'll probably have to gain root access to solve this problem.

Windows

Windows users may simply delete the eZ Publish directory using the "Explorer".

Removing the database

MySQL

1. Start the MySQL client, log in using your username and password:

```
$ mysql -u <username> -p
```

If the username/password is correct, the client will then present a "mysql>" prompt.

2. Delete/remove the database using the drop command followed by the name of the database used by eZ Publish:

```
mysql> drop database <database-name>;
```

PostgreSQL

1. Remove the database by executing the PostgreSQL dropdb command from shell:

```
$ dropdb <database-name>
```

Reconfiguring Apache (optional)

If a virtual host setup was used, it is likely that the Apache configuration file contains eZ Publish specific settings. These settings will not be needed anymore and thus they can be removed. Open the "httpd.conf" file using a text editor, scroll down to the bottom and remove the eZ Publish specific virtual host settings. Remember to restart Apache after altering the configuration file.

Removing the cronjobs (optional)

Windows users should skip this part. If cron was configured to run eZ Publish specific jobs, then these will have to be removed. You may have to edit a global cron file (under "/etc/cron*") or use the "crontab" command with the -e (edit) parameter to edit a user's private cron file. Remove the eZ Publish specific entries.

1.8 Extensions

Extensions are plug-ins to eZ Publish, providing additional custom functionality. Various extensions are available for eZ Publish. All of them require the same basic steps for an installation. This chapter will show how to perform the following:

1. Extract the compressed archive containing the extension
2. Activate the extension

Some extensions might require further action to make them fully functional, e.g. creating new database tables, adding certain content classes to eZ Publish, etc. Such additional measures are explained in the documentation for each extension.

As outlined before, this section deals with the basic steps only. For demonstration purposes, the installation will be exemplified by an imaginary extension called "ezfoo".

1.8.1 Extension load ordering

Introduction

In eZ Publish Fuji, there is a mechanism that enables extensions to define in which order they should be loaded in regards to other, related, extensions. The function automates the extension load ordering and improves the extension loading so that each request is quicker, as such eZ Publish scales better with many extensions. This is done either by adding cache in the processing of extensions, or move non-critical parts out of the code-path which is run on each request.

Requirements

For the extension load ordering mechanism to be active, extensions should be recent enough to contain the needed metadata stored in the "extension.xml" file, located at the root directory of the extension. The following is a list of extensions and versions that contain the necessary metadata:

Extension	Version
eZ Comments	1.1
eZ Find	2.3
eZ Flow	2.2
eZ Image Editor	1.1
eZ SI	1.3
eZ Style Editor	1.1
eZ Survey	2.1
eZ Teamroom	1.1
eZ XML Export	1.3
eZ ODF	1.1
eZ Online Editor	5.x
eZ Website Interface	1.7
eZ GMAP Location	1.1
eZ JS Core	1.1
eZ Multiupload	1.1

Configuration

To activate the extension load ordering, the following configuration setting should be enabled in "site.ini.append.php" in setting/override:

```
[ExtensionSettings]
  ExtensionOrdering=enabled
```

Extension ordering

The order in which active extensions appear in the "site.ini" under /settings/override/site.ini "[ExtensionSettings] ActiveExtensions=" is important for the selection of templates and, possibly, configuration settings. For this reason, users must insure that the correct order is used for extensions that have dependencies on other extensions. For example eZ Find must be activated before eZ Webin. The same rule applies for the eZ Teamroom extension: eZ Teamroom and all its dependent extensions, such as eZ Lightbox and eZ Event, need to be activated before eZ Webin.

Although extensions can be activated using the Administrator Interface, as for now the admin interface does not provide a way to re-order the extensions, manually or automatically. This implies that the order can only be defined in site.ini.

Defining the correct order can sometimes be tricky when many extensions need to be considered. By providing metadata to the extension loading, manual extension load ordering is now a thing of the past.

Metadata

By providing metadata for the extension load ordering we enable eZ Publish to correctly determine the correct order itself. The metadata about extension ordering is optional. These metadata are now declared using an XML file, that can also be used to replace what is currently in "ezinfo.php". This file is named "extension.xml", located at the root of the extension. It has the following content:

```
<?xml version="1.0" encoding="utf-8" ?>
<software>
  <metadata>
    <name>Extension name</name>
    <version>X.Y.Z</version>
    <copyright>Copyright text</copyright>
    <license>License type</license>
    <info_url>http://url/to/extension/site</info_url>
  </metadata>
</software>
<software>
  <uses>
    <name>Used library/software 1</name>
    <license>Used library/software 1 license</license>
    <copyright>Used library/software 1 copyright</copyright>
    <info_url>Used library/software 1 URL</info_url>
    <version>Used library/software 1 version</version>
  </uses>
  <uses>
    <name>Used library/software 2</name>
    <license>Used library/software 2 license</license>
    <copyright>Used library/software 2 copyright</copyright>
    <info_url>Used library/software 2 URL</info_url>
  </uses>
</software>
```

```
        <version>Used library/software 2 version</version>
    </uses>

</software>

</metadata>

<dependencies>
    <uses>
        <extension name="usedExtension" />
    </uses>
    <requires>
        <extension name="requiredExtension" />
    </requires>
    <extends>
        <extension name="extendedExtension" />
    </extends>
</dependencies>

</software>
```

The first block (software/metadata) replaces "ezinfo.php". The fields are equivalent to those found in the info() method of the 'old' ezinfo. The sub-node software/metadata/software replaces the previous 3rdparty_libraries entry. One software/metadata/software/uses node must be used for each library.

The second block (software/dependencies) contains 3 sub-nodes:

- **uses:** indicates that the referenced extension is used by this extension.
- **requires:** indicates that the referenced extension is required by this extension.
- **extends:** indicates that the referenced extension is overloaded by this extension.

Note: 'uses' and 'requires' currently have the same effect, but their behavior will change when real dependencies checking is developed in the future.

1.8.2 Extracting the files

Each extension is distributed as a compressed archive. The name of the archive file includes the name of the extension and its release version. Furthermore, the compression type is indicated by the file ending, either "tgz", "tar.gz", "bz2", or "zip". For example:

- ezfoo-extension-1.0.tgz
- ezfoo-extension-1.0.tar.gz
- ezfoo-extension-1.0.bz2
- ezfoo-extension-1.0.zip

Windows users should download the "zip" archive. Linux/UNIX users may download any archive format as long as the necessary unpacking tools are available.

Extension base directory

Copy the downloaded archive into the "extension" directory of your eZ Publish installation. If this directory does not exist yet, then create it. (Do not create the directory with the plural naming "extensions" - this is a common error.)

The following shell commands can be used to create the "extension" directory and copy the archive on a Linux/UNIX system:

```
mkdir /opt/ezp/extension/
cp /home/myuser/download/ezfoo-extension-1.0.tar.gz /opt/ezp/extension/
```

Replace "/opt/ezp/" with the actual path to your eZ Publish installation and "/home/myuser/download/ezfoo-extension-1.0.tar.gz" with the actual path to the downloaded archive.

Unpack the archive

The archive should be unpacked inside the "extension" directory. When done correctly, an "ezfoo" directory will be created inside the "extension" directory.

See the following table for the correct shell command to use on a Linux/UNIX system, depending on the compression type:

Archive type	Command to extract
tar.gz or tgz	<pre>tar -zxvf ezfoo-extension-1.0.tar.gz</pre> <p>or</p> <pre>tar -zxvf ezfoo-extension-1.0.tgz</pre>
bz2	

	<pre>tar -jxvf ezfoo-extension-1.0.bz2</pre>
zip	<pre>unzip ezfoo-extension-1.0.zip</pre>

On Windows, you can just unzip the "zip" file using the built-in zip features.

At this point, the unpacked files should be available under "extension/ezfoo".

1.8.3 Activating the extension

Each extension needs to be activated, which means that it is being registered for eZ Publish to be available from within the eZ Publish framework. Every extension can either be activated in the eZ Publish administration interface or in a configuration file. Furthermore, the activation can be done either for the whole eZ Publish installation or for only certain site accesses.

Note: As of the 4.4 release of eZ Publish we have implemented a feature for automatic loading of extensions in the required sequence. Please go to Extension load ordering (page 88) for more information on how to do this. If you are running on releases prior to 4.4, please continue reading below.

Administration interface

Log in to your eZ Publish administration interface, click on the "Setup" tab, and then click "Extensions" on the left. You will see the list of available extensions with check-boxes. Simply select the ones you need as shown in the screenshot below and click the "Apply changes" button.

(see figure 1.16)

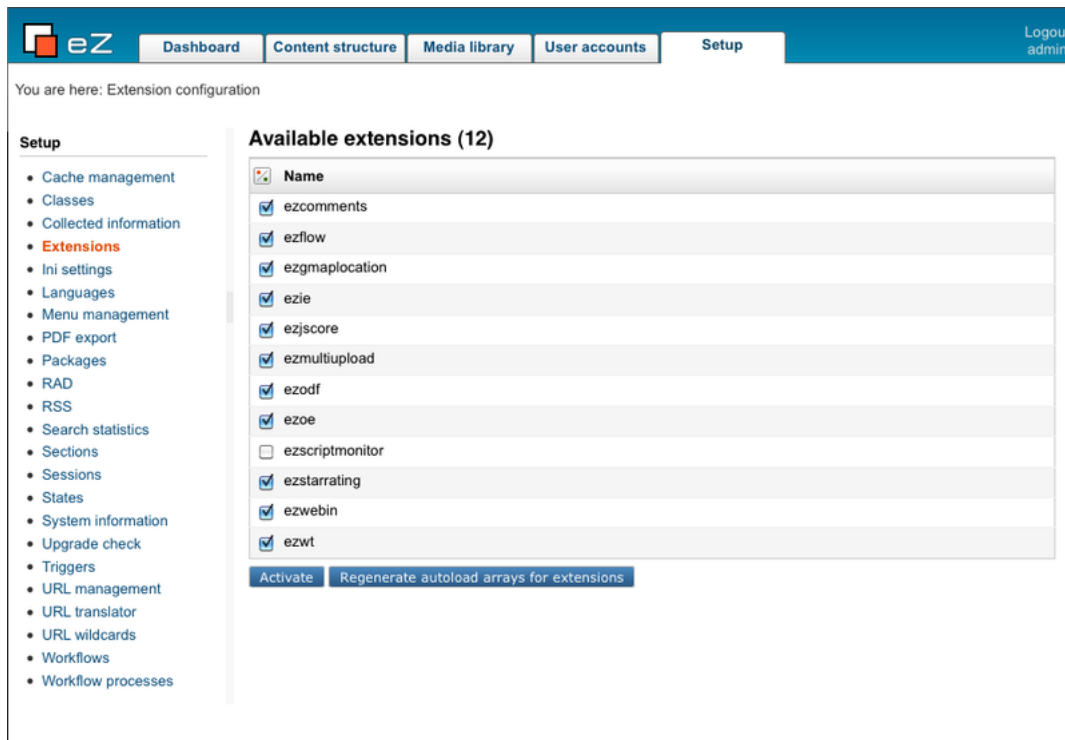


Figure 1.16:

This will activate the extension for all siteaccesses of your eZ Publish installation.

Configuration file

Alternatively, an extension can be enabled manually in the site.ini configuration file.

Activating for the whole installation

To enable the sample extension for all of your site accesses, edit the "site.ini.append.php" file located in the "settings/override" directory of your eZ Publish installation. Add the following line under the "[ExtensionSettings]" configuration block (page 140):

```
ActiveExtensions []=ezfoo
```

Multiple extensions can be present within the "[ExtensionSettings]" block. You'll have to manually create the file and/or the section if they do not exist.

Activating for certain siteaccesses

To enable the sample extension for only a single site access called "example", edit the "site.ini.append.php" file located in the "settings/siteaccess/example" directory of your eZ Publish installation. Add the following line under the "[ExtensionSettings]" configuration block:

```
ActiveAccessExtensions []=ezfoo
```

Note that the line registering the extension is not called "ActiveExtensions", but "ActiveAccessExtensions". You'll have to manually create the file and/or the section if they do not exist.

Updating the autoload arrays

After updating the configuration file(s), you need to run the "ezpgenerateautoloads.php" script, in order to add the information about all PHP class definitions of this extension to the "autoload/ezp_extension.php" file, otherwise eZ Publish might not be able to execute the PHP code of the newly added extension. The following example shows how to run the script.

1. Navigate into the eZ Publish directory.
2. Run the script using the following shell command:

```
bin/php/ezpgenerateautoloads.php --extension
```

The script will look for class definitions in the "extensions" directory and update the "autoload/ezp_extension.php" file accordingly.

1.9 Troubleshooting

This section will explain what can be done if installation fails because of some unknown reason. First of all, make sure that all the requirements (page 39) without exception are met. The requirements are strict and extremely important. Please read them very carefully.

If all the requirements are met but you still have problems, it is recommended to check the debug information during the installation process. To enable the debug output, do the following:

1. Go to the "settings/override" directory of your eZ Publish installation.
2. Create a new file called "site.ini.append.php" and put the following lines to it:

```
[DebugSettings]
DebugOutput=enabled
```

The debug output will appear at the bottom of the page as shown in the following screenshot.

(see figure 1.17)

The debug output will be displayed in the setup wizard, in the administration interface and on the actual site. This option can be disabled at any time by replacing "enabled" with "disabled" in the same place of the configuration file.

Note that the "CheckValidity" setting located in the "[SiteAccessSettings]" section of the same file controls if the setup wizard should automatically start the first time the site is accessed/browsed. If you want to restart the wizard after its successful finishing, you can specify "CheckValidity=true" in the "settings/override/site.ini.append.php" file so that the setup wizard will be initiated when trying to access the site.



Figure 1.17: The debug output appears at the bottom of the page

Chapter 2

Concepts and basics

The purpose of this chapter is to introduce and describe the most important concepts of eZ Publish. A rookie developer should definitively read through this chapter in order to understand the basic terms, models, structures and building blocks of the system.

This chapter is more generic than technical, it is meant to teach the concepts rather than explaining details. People previously unfamiliar with eZ Publish should be able to collect enough information in order to understand the following issues:

- The way eZ Publish is built up
- The main directory structure
- The concept and necessity of separating content and design
- How eZ Publish stores and manages content
- How eZ Publish handles issues related to design
- How eZ Publish manages different sites
- The concept of modules and views
- The way eZ Publish works with URLs
- The configuration system
- The structure of the work-flow system
- How the access/permission system works
- How the web shop works
- A typical page request cycle

2.1 The internal structure of eZ Publish

This section describes the internal structure of eZ Publish by presenting an brief overview of the different software-layers of the system. eZ Publish is a complex, object oriented application written in the PHP language. The system consists of three major parts:

- Libraries
- Kernel
- Modules

The following illustration shows how the different parts of the system are connected.

(see figure 2.1)

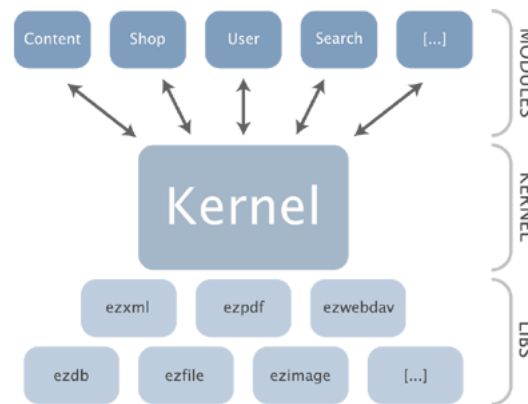


Figure 2.1: Libraries, kernel and modules.

The libraries

The libraries are the main building blocks of the system. These are re-useable general purpose PHP classes. The libraries are in no way dependent on the eZ Publish kernel. However, some of them are strongly interconnected and thus inseparable. People looking for general PHP libraries should take a look in the "lib" folder within the root directory of an eZ Publish installation. The reference chapter contains a complete list and a short description of the currently available libraries.

The kernel

The eZ Publish kernel can be described as the system core. It takes care of all the low level functionality like content handling, content versioning, access control, work-flows, etc. The kernel consists of various engines that build upon and make use of the general purpose libraries.

The modules

An eZ Publish module offers an HTTP interface which can be used for web based interaction with the system. While some modules offer an interface to kernel functionality, others are more or less independent of the kernel. eZ Publish comes with a collection of modules that cover the needs of typical everyday tasks. For example, the content module provides an interface that makes it possible to use a web browser to manage content. The reference chapter contains a complete list and a short description of all the currently available modules. A module can be broken down into the following components:

- Views
- Fetch functions

A view provides an actual web interface. For example, the "search" view of the "content" module provides a web interface to the built-in search engine. Every eZ Publish module provides at least one view. A fetch function makes it possible to extract data through a module from within a template. For example, the "current_user" fetch function of the "user" module makes it possible to access information related to the user who is currently logged in. Some modules provide fetch functions, some don't.

2.1.1 Directory structure

The eZ Publish root directory contains multiple sub-directories. Each sub-directory is dedicated to a specific part of the system and contains a collection of logically related files. The following table gives an overview of the main eZ Publish directories.

Directory	Description
bin	The "bin" directory contains various PHP, Perl and shell scripts. For example, it contains the "ezcache.php" script which can be used to clear all eZ Publish caches from within a system shell. The scripts are mainly used for manual maintenance.
cronjobs	The "cronjobs" directory contains miscellaneous scripts for automated periodical maintenance.
design	The "design" directory contains all design related files such as templates, images, stylesheets, etc.
doc	The "doc" directory contains documentation and change logs.
extension	The "extension" directory contains eZ Publish plug-ins. The extension system of eZ Publish allows external code to plug in and co-exist with the rest of the system. By using extensions it is possible to create new modules, datatypes, template operators, workflow events and so on.
kernel	The "kernel" directory contains all the kernel files such as the core kernel classes, modules, views, datatypes, etc. This is where the core of the system resides. Only experts should tamper with this part.
lib	The "lib" directory contains the general purpose libraries. These libraries are collections of classes that perform various low level tasks. The kernel makes use of these libraries.
packages	The "packages" directory contains the bundled packages (themes, classes, templates, etc.) that can be installed using either the setup wizard or the administration interface.
settings	The "settings" directory contains dynamic, site specific configuration files.
share	The "share" directory contains static configuration files such as code pages, locale descrip-

	tions, translations, icons, etc.
support	The "support" directory contains the source code for additional applications that can be used to do various advanced tasks. For example, it contains the "lupdate" program that can be used to create and maintain eZ the translation files.
update	The "update" directory contains various scripts that should be used when an eZ Publish installation is being upgraded.
var	The "var" directory contains cache files and logs. It also contains actual content that doesn't go into the database (images and files). The size of this directory will most likely increase as the system is being used.

2.2 Content and design

This section explains the fundamental concepts of content and design. It is important to understand what content and design actually are, how they interconnect and how the system handles these fundamental elements.

Content

In the world of eZ Publish, content and design are separated. By content we mean information that is to be organized and stored using some structure. For example, it may be the actual contents of a news article (title, intro, body, images), the properties of a car (make, model, year, color) and so on. In other words, all custom information that is stored for the purpose of later retrieval is referred to as content.

Design

The information stored in a content structure must be presented somehow, preferably in a way that is easily understood by humans. While content means actual data, design is all about the way the data is marked up and visually presented. When talking about design, we're talking about the things that make up a web interface: HTML, style sheets, images that are not a part of the content, etc.

Templates

eZ Publish uses templates as the fundamental unit of site design. For example, a template might dictate that a page should appear with the site's title bar on the top, and then main content in the middle. When the page is accessed, it then becomes the content management system's job to "flow" the content into the appropriate places in the template. An eZ Publish template is basically a custom HTML file that describes how some particular type of content should be visualized. In addition to standard HTML syntax, it is possible to use eZ Publish specific code to for example extract content from the system. The HTML syntax in the built-in/default templates follow the XHTML 1.0 Transitional specification.

The separation of content and design

While content is all about storing and structuring custom/raw data, the purpose of the design is to dictate how the content should be visualized. The result of a combination of these elements is a complete interface, as illustrated in the following diagram.

(see figure 2.2)

This distinction, and the system's ability to handle it is one of the key features of eZ Publish. The separation of content and design opens up an entire range of possibilities that simply cannot

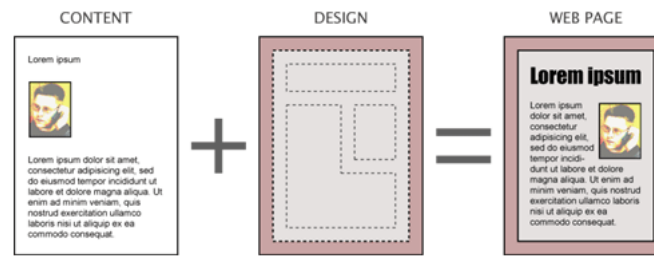


Figure 2.2: *Content + Design = Web page*

be achieved otherwise. The following list outlines some of the most important benefits of this technique:

- Content authors 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/changes can be applied by simple modifications

2.2.1 Storage

This section explains where eZ Publish stores information that belongs to a site (not the system itself). A typical eZ Publish site consists of the following elements:

- Actual content
- Design related files
- Configuration files

Actual content is structured and stored inside a database. This is true for all content except for images and files, which are stored on the file system. The main reason for this is because the file system is much faster than the database when it comes to the storage and retrieval of large data chunks. Having the files on the file system allows the web-server to serve them directly without the need of going through the database. In addition, this technique makes it easier to use external tools to manipulate/scan/index the contents of the uploaded files. For example, the built in search engine is capable of using external utilities to index the contents of miscellaneous files (PDF, Word documents, Excel sheets, etc.). Having the files on the file system dramatically decreases the size of the database and thus makes it easier to copy and handle. Everything that is related to design (template files, CSS files, non content specific images, etc.) and configuration settings are also stored on the file system. A backup of an eZ Publish site must therefore contain both a dump of the database and a copy of the necessary files. The following illustration shows an overview of how the system makes use of the database and the file system to store the different elements of a site.

(see figure 2.3)

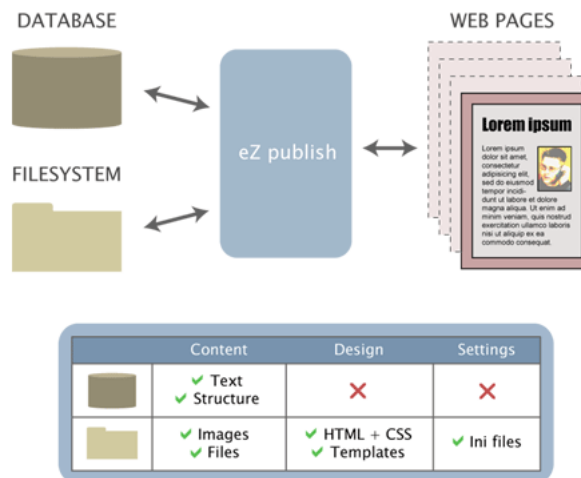


Figure 2.3: Storage overview

2.3 Content management

The role of a content management system is to organize and store content regardless of type and complexity. The main goal of such a system is to provide a well structured, automated yet flexible solution allowing information to be freely distributed and instantly updated across various communication channels (such as the world wide web, intranets and miscellaneous front and back-end systems). This section describes how eZ Publish actually handles content.

A typical example

Let's consider a scenario at a university with a need of storing information about students. Most off-the-shelf content management systems will offer a selection of built in content types. There might for example exist a "Person" type, consisting of fields like "name", "birthdate", "phone number" and so on. However, the custom student data will probably not fit perfectly into this predefined model since it might consist of information that is specific for the university (for example student ID, department, etc.). Even though some systems allow the creation of custom structures, the solution is often a complicated and time consuming process that requires both programming and manipulation of the database. In addition, once the solution is in use, future alternation of the structure itself will most likely become a problem.

Content management in eZ Publish

Unlike other content management systems, eZ Publish does not make use of a predefined "one-size-fits-all" approach. Instead of desperately trying to fit data into predefined and rigid structures, the system allows the creation of custom structures by the way of a unique object oriented approach. For example, the site developer can build custom structures that perfectly satisfies the storage needs of the university. This is one of the key features that make eZ Publish a flexible and successful system. In addition to offering the freedom of custom structures, it also allows the modification of the content structures at run-time. In other words, if the custom student structure used in the example above needs to be modified, then eZ Publish will automatically alter it based on the administrator's commands.

Although the possibility to create and modify content structures is a wonderful feature, there isn't always need for using it. This is why an eZ Publish distribution comes with a selection of predefined content structures and thus allows the developer to choose between the following scenarios:

- Use the standard/built-in structures
- Use modified versions of the standard/built-in structures
- Use only custom structures
- Use a combination of standard, modified and custom structures

An object-oriented content structure

The eZ Publish content structure is based on ideas borrowed from the object oriented world of popular programming languages like Smalltalk, C++, JAVA, etc. Superficially, object-oriented means nothing more than looking at the world in terms of objects. In real life, people are surrounded by several objects: furniture, cars, pets, humans, etc. Each of these objects have traits that we use to identify them. This is also the way eZ Publish defines and manages content.

The system offers a selection of fundamental building blocks and mechanisms that together provide a flexible content management solution. An actual data structure is described using something called a *content class*. A content class is built up of attributes. An *attribute* can be thought of as a field, for example the "birthdate" field in a structure designed to store information about students. The description of the entire structure would be referred to as the "student class". The characteristics of an attribute inside the class are determined by the *datatype* that was chosen to represent that attribute.

It is important to understand that a content class is just a definition of an arbitrary structure. In other words, the class itself describes the structure but it does not store any actual data. Once a content class has been defined, it is possible to create instances of that class. An instance of a content class is called a *content object*. Actual content is stored inside objects of different types. A content object consists of one or more *versions*. The versioning layer makes it possible to have different versions of the same content. Each version consists of one or more *translations*. The translation layer makes it possible to represent the same version of the same content in multiple languages. A translation consists of *attributes*. The attributes are the final elements in the content structure chain, this is where actual data is stored.

The content objects are wrapped and organized by the way of *nodes* that are placed inside a tree-like structure. This tree is often referred to as the *node tree*. The following sections contain comprehensive explanations related to the elements that were introduced above.

2.3.1 Datatypes

A datatype is the smallest possible entity of storage. It determines how a specific type of information should be validated, stored, retrieved, formatted and so on. eZ Publish comes with a collection of fundamental datatypes that can be used to build powerful and complex content structures. In addition, it is possible to extend the system by creating custom datatypes for special needs. Custom datatypes have to be programmed in PHP. However, the built in datatypes are usually sufficient enough for typical scenarios. The following table gives an overview of the most basic datatypes that come with eZ Publish.

Datatype	Description
Text line	Stores a single line of unformatted text
Text block	Stores multiple lines of unformatted text
XML block	Validates and stores multiple lines of formatted text
Integer	Validates and stores a numerical integer value
Float	Validates and stores a numerical floating point value

Please refer to the "Datatypes" section of the reference chapter for a comprehensive list of all the built-in datatypes. Additional datatypes can be downloaded from <http://ez.no/community/contribs/datatypes>; they are created by the members of the eZ Publish community.

Input validation

As the list above indicates, some datatypes take care of more than just storing data. For example, the "XML block" datatype apparently supports validation. This means that the inputted XML will be validated before it is actually stored in the database. In other words, the system will only accept and store the data if it is a valid XML structure. Input validation is supported by most (but not all) of the built in datatypes. The validation feature of a datatype can not be turned on or off. In other words, if a datatype happens to support validation, it will always try to validate the incoming data and thus the system will never allow the storage of incorrectly formatted input.

2.3.2 The content class

A content class is a definition of an arbitrary data structure. It does not store any actual data. A content class is made up of attributes. The characteristics of an attribute are determined by the datatype that is chosen for that specific attribute. By combining different datatypes, it is possible to represent complex data structures. The following illustration shows the anatomy of a content class called "Article", which defines a data structure for storing news articles. It consists of attributes dedicated for storing the title, an introduction text and the actual body of an article.

(see figure 2.4)

ARTICLE CLASS

ATTRIBUTES	Name	Datatype
	Title	Text line
	Intro	Text line
	Body	XML field

Figure 2.4: Example of a content class.

An eZ Publish distribution comes with a set of general purpose classes that are designed for typical web scenarios. For example, the default image class defines a structure for storing image files. It consists of attributes for storing the name of the image, the actual image file, the caption and an alternative image text. The built-in classes can be modified in order to become more suitable for a specific case. In addition, it is possible to create completely new and custom classes. Content classes can be created, modified and removed easily using the administration interface. When a content class is removed, all instances of that class (containing actual data) will also be removed from the system. The following screenshot shows the class edit interface.

(see figure 2.5)

Class structure

A content class consists of the following elements:

- Name
- Identifier
- Object name pattern
- URL alias name pattern
- Container flag
- Default sorting of children

OK Apply Cancel Text line Add attribute

Edit <Article> (5)

Last modified: 15/09/2010 12:36 pm, Administrator User English (United Kingdom)

Name:
Article

Identifier:
article

Description:

Object name pattern:
<short_title|title>

URL alias name pattern:

Container:

Default sorting of children:
Path String Ascending

Default object availability:

Class attributes:

1. Title [Text line] (id:183) 10

Name:
Title

Identifier:
title

Description:

Figure 2.5:

- Default object availability flag
- Attributes

Name

The name is for storing a user friendly name which describes the data structure that the class defines. A class name can consist of letters, digits, spaces and special characters. The maximum length is 255 characters. For example, if a class defines a data structure for storing information about graduate students, the name of the class would most likely be "Graduate student". This name will appear in various class lists throughout the administration interface, but it will not be used internally by the system. If a blank name is provided, eZ Publish will automatically generate a unique name when the class definition is stored.

Identifier

The identifier is for internal use. In particular, class identifiers are used in configuration files, templates and in PHP code. A class identifier can only consist of lowercase letters, digits and underscores. The maximum length is 50 characters. For example, if a class defines a data structure for storing information about graduate students, the identifier of the class would probably be "graduate_student". If a blank identifier is provided, eZ Publish will automatically generate a unique identifier when the class definition is stored.

Object name pattern

The object name pattern controls how the name of an actual object (an instance of a class) will be generated. A pattern usually consists of attribute identifiers (described later) that tell eZ Publish about which attributes it should use when generating the name of an object. Each attribute identifier has to be encapsulated by angle brackets. Text outside the angle brackets will be included directly. If a blank pattern is provided, eZ Publish will automatically use the identifier of the first attribute.

URL alias name pattern

The URL alias name pattern controls how the virtual URLs (page [153](#)) of the nodes will be generated when the objects (instances of a class) are created. Note that only the last part of the virtual URL is affected. The pattern works in the same way as the object name pattern. Text outside the angle brackets will be converted using the selected method of URL transformation. If a blank pattern is provided, eZ Publish will automatically use the name of the object itself.

Container flag

The container flag controls whether an instance of the class should be allowed to have sub items (often called child nodes, children) or not. This setting only affects the administration interface, it was added in order to provide a more convenient environment for administrators and content authors. In other words, it doesn't control any actual low level logic, it simply controls the way the graphical user interface behaves.

Default sorting of children

From 3.9, it is possible to set the "default sorting of children" while editing the classes. When new objects are created and their corresponding nodes appear in the tree, they will use the sorting settings that were specified at the class level. In other words, if you set the "default sorting of children" to priority/ascending for the "Folder" class, the sub items of newly created folders will be sorted by their priorities, starting with the lowest priority.

Note that sorting parameters can always be changed for each individual node by using the sorting controls located in the "Sub items" window. Modifying the default sorting parameters at the class

level will not affect the nodes that encapsulate existing objects of the class (only the nodes of newly created objects will be affected). Refer to "sort method" (page 123) and "sort order" (page 115) for more information about sorting parameters.

Default object availability flag

This flag is related to the multi-language features that were added in eZ publish 3.8. It simply dictates the default value of the "object availability" flag for new instances (objects) of the class. This flag can be further controlled (on the object level) by a check-box labeled "Use the main language if there is no prioritized translation" in the "Languages" window of the administration interface. In other words, the object availability can be modified individually for each object. When the flag is set, an object that does not exist in one of the site/prioritized languages will be shown using its initial/main language. If the flag isn't set, the object will not be shown as long as it does not exist in one of the prioritized languages.

Attributes

As pointed out earlier, it is the structure and type of the attributes that make up the actual data structure that the class defines. A content class must at least have one attribute. On the other hand, a class can contain virtually an unlimited number of attributes. Class attributes can be added, removed and rearranged at any time using the administration interface. If an attribute is added to a class, it will be added to all current and upcoming instances of that class. If an attribute is removed, it will also be removed from all instances.

Although it is possible to remove and add attributes using the administration interface, in some cases these operations may corrupt the database. This usually happens when there are too many instances that need to be updated. If the required processing time exceeds the maximum execution time for PHP scripts, the sequence will be interrupted and thus the database will most likely be left in an inconsistent state. At the time of writing, this problem can only be solved by increasing the maximum execution time, which is defined in "php.ini" as "max_execution_time". The default value is 30 seconds, it should be increased to a couple of minutes. A more reliable solution (a PHP script that takes care of adding/removing attributes and run it from within a shell) will probably be added in the future.

2.3.3 Class attributes

A content class is made up of one or more attributes where each attribute is represented by a datatype. The characteristics of an attribute are determined by the datatype that is chosen for that specific attribute. An attribute is made up of the following elements:

- Name
- Identifier
- Generic controls
- Datatype specific controls

Name

The name is for storing a user friendly name for the attribute. For example, if the attribute is supposed to store birth dates, the name of the attribute would most likely be "Date of birth". This string will appear in various parts of the administration interface, but it will not be used internally by the system. The name of an attribute can consist of letters, digits, spaces and special characters. The maximum length is 255 characters. If a blank name is provided, eZ Publish will automatically generate a unique name for the attribute when the class definition is stored.

Identifier

The identifier of an attribute is for internal use. In particular, attribute identifiers are used in configuration files, templates and in PHP code. An attribute identifier can only consist of lowercase letters, digits and underscores. The maximum length is 50 characters. For example, if the attribute is supposed to store birth dates, the identifier of the attribute would probably be "date_of_birth". If a blank identifier is provided, eZ Publish will automatically generate a unique identifier when the class definition is stored.

Generic controls

Each attribute has a set of generic controls. These controls are the same for each attribute, regardless (but not independent) of the datatype that represents the attribute. The generic controls are a set of switches that can be turned on or off:

- Required
- Searchable
- Information collector
- Translatable

Required

The required switch controls the behavior of the storage procedure for content objects (instances of a content class). It can be used regardless of the datatype that represents the attribute. When the required flag of an attribute is set, the system will keep rejecting the inputted data until all required information is provided. If the required flag is unset, eZ publish will not care whether any actual data was provided or not. When an attribute is added, the required switch is off. Please note that inputted data will be validated according to the chosen datatype's validation rules regardless of the state of the attribute's required switch. Input validation is supported by most (but not all) of the built in datatypes. The following example demonstrates how these features actually work.

Let's say that we have created a content class that defines a data structure for storing information about prisoners. The class would typically consist of various attributes for storing different kinds of data: name, identification number, date of birth, cell, block, etc. Having at least the name and the birth date attributes required will eliminate the possibility of storing convicts without names and/or birth dates. If the birth date attribute is represented by the built-in "date" datatype, the system will only accept the input if the birth date is provided using a correct date format.

Searchable

The searchable switch can be used to control whether the actual data stored using the attribute should be indexed by the search engine or if it should be left unindexed. Search indexing is supported by the majority of the built-in datatypes. Please refer to the "Datatypes" section of the reference chapter to see which datatypes that support search indexing.

Information collector

The information collector switch can be used to control the attribute's behavior in view mode. The default view mode behavior results in the display of the information that was provided in edit mode. For example, when viewing a news article, the contents of the article are displayed but can not be edited. However, if an attribute is marked as a collector, it will allow information to be input in view mode. At first, this feature might seem a bit odd. However, it is actually quite handy. For example, it can be used to quickly create simple feedback forms. The contents of a form created using this technique will be e-mailed to the site administrator (or to a specified address) once the form is submitted. Information collection is only supported by a small set of the built in datatypes. The following example demonstrates how this feature could be used to create a basic feedback form.

Let's say that we have created a content class called "Feedback form" using the following attributes: name, subject and message. The subject and the message attributes would be marked as information collectors. When an instance of this class is viewed, the subject and the message attributes will be displayed as input fields along with a "Send" button.

Translatable

The translatable switch controls whether actual data stored using the attribute should exist in only one language (the default language) or if it should be possible to translate it using the additional languages. The translation mechanism is completely independent of the datatype layer. In other words, this switch can be used regardless of the datatype that was chosen to represent the attribute.

When an attribute is added, the translation switch is "on". Turning it off is typically useful when the attribute is supposed to store non-translatable input. For example, translating dates, numerical values, prices, email addresses, etc. doesn't make much sense.

Datatype specific controls

An attribute can have a set of additional controls that are specific for the datatype that was chosen to represent that attribute. Some datatypes allow fine grained customization, some not. For example, the built-in "Text line" datatype provides two settings: default value and maximum length.

2.3.4 The content object

A content object is an instance of a content class. While the class only defines the data structure, it is the content objects themselves that contain actual data. Once a content class is defined, several content objects / instances of that class can be created. For example, if a class for storing news articles is created, several article objects (each containing a different story) can then be instantiated. The following illustration summarizes and shows the relation between datatypes, attributes, a content class and content objects.

(see figure 2.6)

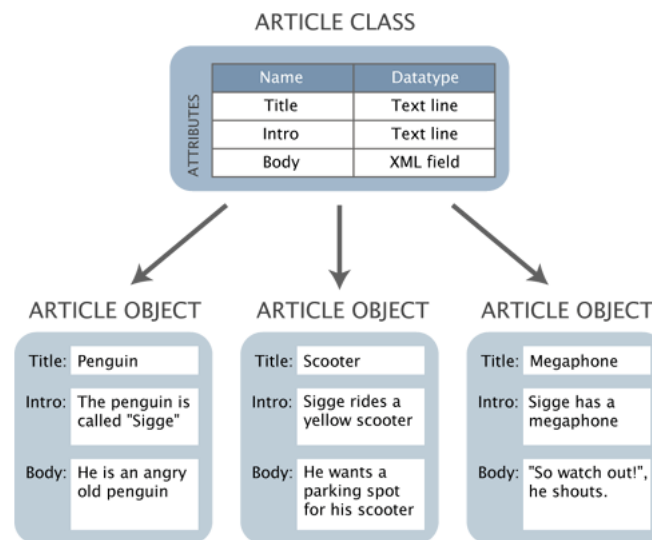


Figure 2.6: Datatypes, attributes, a content class and objects.

Please note that the illustration above is a simplified version of the reality. It doesn't show the exact structure of the objects since the versioning and the translation layers have been left out. The following text gives a more in-depth explanation of the object structure. The versioning and the translation layers will be explained in the upcoming sections.

Note: A non-technical understanding of the "Content Object": one piece of content like an article, a picture or a movie, managed by eZ Publish. Describing attributes like author, title, subtitle etc. are belonging to the content object. A single content object can have different translations and versions. One and the same object can be used and displayed in several channels and contexts € i.e. one picture, used in different publications is still regarded as one content object.

Object structure

A content object consists of the following elements:

- Object ID

- Name
- Type
- Owner
- Creation time
- Modification time
- Status
- Section ID
- Versions
- Current version

Object ID

Every object has a unique identification number. The ID numbers are used by the system to organize and keep track of different objects. These ID numbers are not recycled. In other words, if an object is deleted, the ID number of that object will not be reused when a new object is created.

Name

The name of an object is nothing more than a friendly name that appears in various lists throughout the administration interface. It helps the user to identify different objects by their names instead of having to deal with identification numbers. An object's name is generated automatically by the system when the object is published. It is the object name pattern definition of a class that dictates how objects of that class should be named. This mechanism makes it possible to automatically generate names based on the object's attributes. Since the object name is not used by the system, different objects can have the exact same name.

For example, when dealing with news articles, the title of the article would most likely be used to generate the object names. When an article object is published, its name will be a copy of the object's title attribute. The name of the object will be updated every time the object is published. In other words, if the title is changed, the object's name will automatically also be changed.

Type

The type information indicates which class that was used to create the object.

Owner

The object's owner contains a reference to the user who initially created the object. At any time, an object can only be owned by one user. This reference is set by the system the first time the object is published. The ownership of an object can not be manipulated and will not change even if the owner the object is removed from the system.

Creation time

The published field contains a time-stamp pinpointing the exact date and time when the object was published for the first time. This information is set by the system and it can not be modified. The published time-stamp will remain the same regardless of what happens to the object.

Modification time

The modified field contains a time-stamp revealing the exact date and time when the object was modified. This information is set by the system and it can not be modified. The modified time-stamp will change every time the object is published.

Status

The status indicates the current state of the object. There are three possibilities:

- (0) Draft
- (1) Published
- (2) Archived

When initially created, the object's status is set to *draft*. This status will remain until the object is published and thus the status will be set to *published*. Once published, the object can not become a draft. When a published object is moved to the trash, the status will be set to *archived*. If a published object is removed from the trash (or removed without being put in the trash first), it will be permanently deleted.

Section

The section ID of an object denotes which section that object belongs to. Each object can belong to one section. By assigning different sections to objects, it is possible to have different groups of objects. The section mechanism is explained under "Sections" (page [136](#)).

Versions

The actual contents of an object is stored inside different versions. A version can be thought of as a timestamped collection of data (the object's attributes) that belongs to a specific user. Every time the contents of an object is edited, a new version is created. It is always the new version that will be edited. The current / published version along with earlier versions will remain untouched. This makes it possible to revert unwanted or accidental changes. An object always has at least one version of its content. Each version is identified by a number which is automatically increased for every new version that is created. The structure and logic of the versioning mechanism is explained in the next section.

Current version

The current version is a number that pinpoints the currently published version of the object. As described above, the contents of an object may exist in several versions. However, only one of them can be the current version (also referred to as the *published* version). The current/published version is the version that will be displayed when the object is viewed.

2.3.5 Multiple languages

In addition to the versioning system, the content model of eZ Publish also provides a built-in multi-language framework. This feature allows an object's contents to exist in several languages. The system is able to support up to 30 different languages at the same time.

The multi-language feature provides a generic one-to-one translation mechanism that can be used to translate any kind of content. A one-to-one translation solution makes it possible to represent the exact same content in multiple languages. For example, when a news article is available in English, Norwegian and Hungarian (same content in all three cases), we say that we have one-to-one translation of the content. The translation mechanism is completely independent of the datatypes. In other words, any kind of content can be translated regardless of the datatypes that are used to realize the content's structure. It is possible to start with only one language and when time comes, add translations and thus extend the spectrum of the target audience.

The following illustration shows a simplified example of an object with two versions where each version exists in several languages. A language in this case is often referred to as a *translation*.

(see figure 2.7)

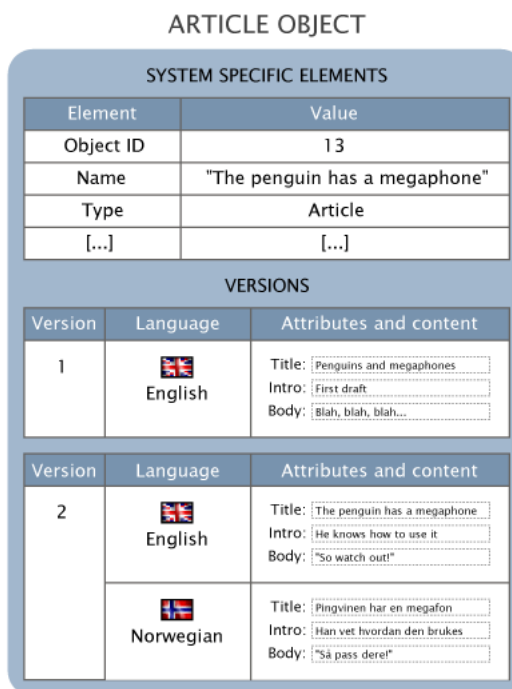


Figure 2.7: Content object structure (with versions and translations).

As the illustration indicates, each version can have a different set of translations. At minimum, a version always has one translation which by default is the *initial/main* translation. The initial/main translation can not be removed. However, if the object exists in several languages, it is possible to select which of the translations that should be initial/main and thus the previous initial/main translation can be removed.



It is important to note that from 3.8, when a user edits an object, it is no longer the entire version that gets edited. Instead, a combination of a version and a translation that is edited. This approach avoids the locking of entire versions (along with all the translations) and thus it allows multiple translators to work with the same content in several languages at the same time.

The global translation list

An object can only be edited/translated using languages that exist in the *global translation list*. Initially, this list contains the languages that were selected during step six of the setup wizard. Additional languages can be added at any time while the site is up and running. The following screenshot shows the global translation list as it appears in the administration interface (under "Setup" and "Languages").

(see figure 2.8)

Available languages for translation of content (2)

<input type="checkbox"/>	Language	Country/region	Locale	Translations	Classes translations
<input type="checkbox"/>	 English (United Kingdom)	United Kingdom	eng-GB	57	39
<input type="checkbox"/>	 French (France)	France	fre-FR	0	0

Remove selected Add language

Figure 2.8:

The global translation list simply keeps track of the languages that users are allowed to use when editing/translating content. A translation added to the list will immediately become available for use. Note that from 3.8, it is no longer possible to remove languages from the global translation list unless they are not used by any objects. The global translation list is capable of handling up to 30 languages.

Differences between 3.8 and earlier versions

In eZ Publish 3.7 and earlier versions, objects had to be created in the primary language before they could be translated to additional languages. Multiple translators could not work simultaneously because the edit process locked the entire version which also contained the translations.

In eZ Publish 3.8, the primary language concept is gone and thus objects can be created using different languages. This means that you can for example have an article available only in English and another article available only in Norwegian. Multiple translators can work on the same object because when editing, they actually edit the translation itself instead of the entire version. This means that if you have written an article in English, different translators can go ahead and add translations (for example Hungarian, Norwegian and Russian) to the object simultaneously. They no longer have to wait for each other because they can work with different translations at the same time on the same object. However, this also means that a user can no longer work with multiple translations at the same time. The problem is that the user must leave the edit interface in order to be able to add (and then edit) new translations for an object. There are some other

drawbacks as well. For example, unless a user is editing the very first version of an object, it is no longer possible to change the object's locations from the edit interface. However, the locations can still be changed using the "Locations" window when the object isn't being edited.

Whenever an object is published, the system automatically collects all the latest translations that the previous version(s) of the object contains and puts them into the version being published. The result is a version that contains all the up-to-date translations. The contents of an object can be translated to a maximum number of 30 languages.

Please refer to the "Updating INI settings for multi-language" part of the "Upgrading from 3.6.x (3.7.x) to 3.8.0" page for information about multi-language related INI settings.

Multilingual classes

From 3.9, it is possible to translate the class names and the attribute names. In other words, you can for example have "Car" and "Bil" as class names in English and Norwegian along with "Top speed" and "Toppfartshastighet" as attribute names. Refer to the "Translatable class attributes" documentation page for more information.

Non-translatable attributes

The data structure defined by a class is built up of attributes where each attribute is represented by a datatype. Among other things, an attribute of a class can be made translatable or not. If an attribute is translatable, the system will allow the translation of its contents when an object of that class is being edited. This is typically convenient when the attribute contains actual text. For example, the written part of a news article can be translated into different languages. However, some attributes are non-translatable by nature. This is typical for images without text, numbers, dates, e-mail addresses and so on. Such attributes can be made non-translatable and thus their contents will simply be copied from the initial/main translation. The copied values can not be edited.

For example, let's say that we need to store information about furniture in multiple languages. We could build a furniture class using the following attributes: name, photo, description, height, width, depth and weight. Allowing the translation of anything else then the description attribute would be pointless since the values stored by the other attributes are the same regardless of the language used to describe the furniture. In other words, the name, photo, height, width, depth and weight would be the same in for example both English and Norwegian. Conversion between different measuring units would have to be done within the template that is used to display the information.

Access control

It is possible to control whether a user (or a group of users) should be able to translate content or not. This policy can be controlled on a class, section, language and owner basis. In particular, the language limitation makes it possible to control which user (or user groups) should be able to edit and/or translate different parts of the content using different languages. In addition, it is

also possible to control access to the global translation list. This makes it possible to allow users other than the site administrator to add and remove translations on a global basis.

Please refer to the multi-language part of the features section for further details.

Share your information

2.3.6 The content node

When the system is in use, new content objects are created on the fly. For example, when a news article is composed, a new article object is created. Obviously, the content objects can't just hover around in space, they have to be organized in some way. This is where the *nodes* and the *content node tree* comes in. A content node is nothing more than an encapsulation of a content object. In eZ Publish, every object is usually represented by one or more nodes. The following illustration shows a simplified example of a node and a corresponding object (which is referenced by the node) as it would have been represented inside the system.

(see figure 2.9)



Figure 2.9: Object - node relation

The content node tree is built up of nodes. A node is simply a location of an object within the tree structure. The tree is the actual mechanism used to hierarchically organize the objects that are present on the system. The content node tree is explained in the next section.

Node structure

A content node consists of the following elements:

- Node ID
- Parent node ID
- Object ID
- Sort method
- Sort order
- Priority

Node ID

Every node has a unique identification number. The ID numbers are used by the system to organize and keep track of the different nodes. These ID numbers are not recycled. In other words, if a node is deleted, the ID number of that node will not be reused when a new node is created.

Parent node ID

The parent node ID of a node reveals the node's superior node in the tree.

Object ID

Every object that exists in the system has a unique identification number. The object ID of a node pinpoints the actual object that the node encapsulates.

Sort method

The sorting method of a node determines how the children of the node should be sorted. The following sorting methods are possible:

Method	ID	Description
Class identifier	6	The nodes are sorted by the class identifiers of the objects.
Class name	7	The nodes are sorted by the class names of the objects.
Depth	5	The nodes are sorted by their depth in the tree. A node further down in the tree has a higher level of depth. The root node has a depth of 1.
Modified	3	The nodes are sorted by the modification time of the objects.
Modified subnode	10	The nodes are sorted based on the modification time of their children.
Name	9	The nodes are sorted by the names of the objects.
Path	1	The nodes are sorted by their path strings.
Priority	8	The nodes are sorted by their priority. Every node has a priority field that can be set by the user. This solution allows the nodes to be sorted in a custom order. The priority field is described below.
Published	2	The nodes are sorted by the creation time of the objects'

		current/published versions.
Section	4	The nodes are sorted by the section IDs of the objects.

Please note that it is possible to combine the available sort methods in order to sort nodes in a more complex way. However, since a node is incapable of "remembering" a combination (you can only set one method and one order for each node), this has to be done in the templates.

Sort order

The sorting order determines the order in which the children of the node should be sorted. There are two possibilities:

- Descending (0 / FALSE)
- Ascending (1 / TRUE)

For example, if the sorting method is set to "Name" and the sort method is set to "Ascending", the underlying nodes will be alphabetically sorted from A to Z. If the sort method is set to "Descending", the underlying nodes will be sorted from Z to A.

Priority

The priority field allows a user to assign both positive and negative integer values to a node (zero is also allowed). This field makes it possible to sort nodes in a custom way. If the sorting method of a node is set to "Priority", the children of that node will be sorted by their priorities.

2.3.7 The content node tree

The content node tree is a hierarchical organization of the objects. Each leaf in the tree is a node (also known as a *location*). Each node refers to one object. The usual case is that an object is referenced by only one node. Because of the node-encapsulation of objects, any type of content object can be placed anywhere in the tree. At the minimum, the tree consists of one node, called the *root node*. The identification number of the root node is 1. The root node is a virtual node, it does not encapsulate an actual object. A node that is directly below the root node is called a *top level node* (the top level nodes are described in the next section). The depth and width of the tree is virtually unlimited. The following illustration shows a simplified example of how objects are referenced by nodes which together make up the content node tree.

(see figure 2.10)

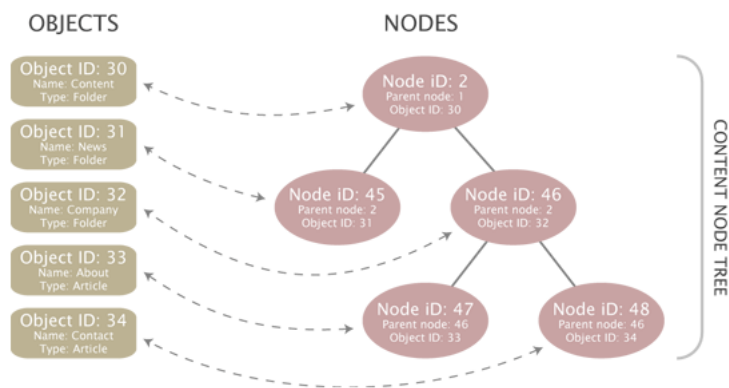


Figure 2.10: Objects, nodes and the content node tree

The following illustration shows the same node structure seen from the outside world.

(see figure 2.11)

Multiple locations

An object may be referenced by several nodes, which means that the same object can appear at different locations within the tree. This feature can for example be used to place a specific news article at two locations: the front page and the archive. In the case of multiple nodes/locations, only one node can be considered as the *main node* of an object. The main node usually represents the object's original location in the tree. The other nodes can be thought of as additional nodes / locations. If an object is referenced by a single node then of course that node would be the main node. Among other things, the main node is used to avoid multiple search hits, infinite recursive loops, smart filtering, etc. The following illustration shows an example of a structure where an object has multiple locations in the tree. It will simply be empty and will have the possibility to contain a different set of sub items.

(see figure 2.12)

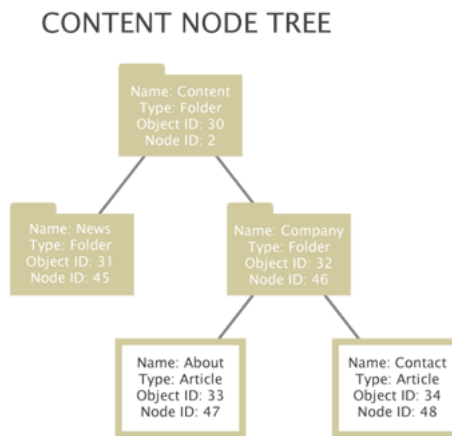


Figure 2.11: Content node tree

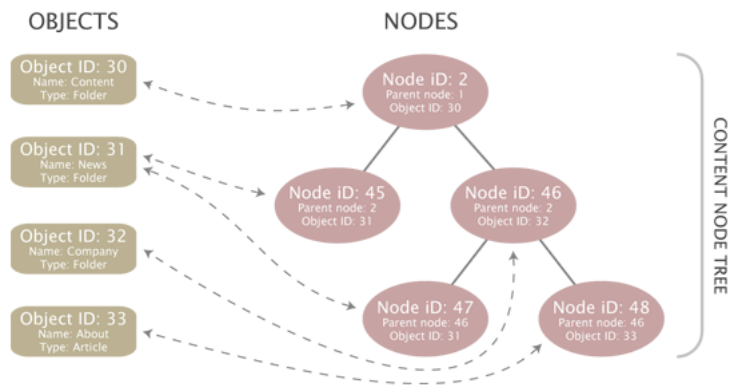


Figure 2.12: Objects, node and the content node tree - multiple locations

The following illustration shows the same node structure seen from the outside world.

(see figure 2.13)

Pitfall

A very common mistake when planning the structure of a site is thinking of multiple locations as shortcuts/links on a file system. Unfortunately, this is not how the node tree works. When a new location is added to an object, eZ Publish will not go through and create replica of the node structure below the object's original location. For example, if a folder containing several sub-folders with articles, images, etc. is assigned a secondary location, the sub-folders with articles, images, etc. will not be automatically available below the new location of the folder.

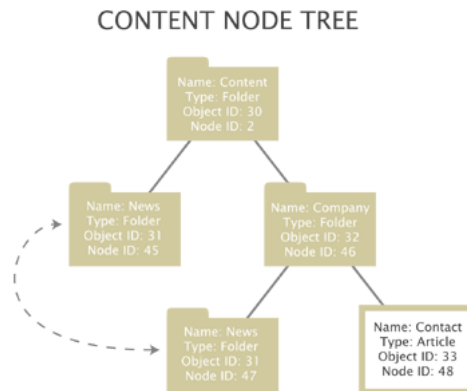


Figure 2.13: *Content node tree with multiple locations*

Additional notes

Only published objects appear in the tree. A newly created object (status set to draft) does not get a node assignment until the object is published for the first time. An object is considered to be deleted (status set to archived) when all nodes referencing that object are removed from the tree. A deleted object will appear in the system trash. It is important to understand that the trash in eZ Publish is a flat structure. This is different from what people are used to from the trash implementation in modern operating systems. Objects in the trash can be recovered to their original locations. However, this is only possible if their original parent nodes have not been deleted. Otherwise, the user must specify a new/alternate location for the objects during recovery. Note that specifying an alternate/new location can be done regardless if the system is able to restore a deleted object at its original location or not.

Furthermore, if a folder containing some news articles is deleted, both the folder and the articles will appear on the same level within the trash. Recovering the folder itself will not bring back the articles since the links between the folder and the articles got lost when the nodes were deleted. In this case, the folder needs to be recovered first. After that, each article has to be manually recovered and given a location.

2.3.8 Top level nodes

A typical eZ Publish installation comes with the following set of top level nodes:

- Content
- Media
- Users
- Setup
- Design

The top level nodes can not be deleted. However, they can be swapped with other nodes. The swap function can be used to change the type of a top level node. For example, the "Content" node references a folder object. By swapping it with another node which refers to a different kind of object, it is possible to change the type of the top level node itself. The following illustration shows the virtual root node and the standard top level nodes:

(see figure 2.14)

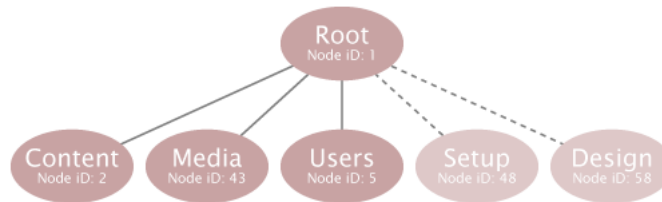


Figure 2.14: Top level nodes

Content

The actual contents of a site is placed under the "Content" node. This node is typically used for organizing folders, articles, information pages, etc. and thus defines the actual content structure of the site. A sitemap can be easily created by traversing the contents of this top level node. The default identification number of the "Content" node is 2. The contents of this node can be viewed by selecting the "Content structure" tab in the administration interface. By default, this node references a "Folder" object.

Media

The "Media" node is typically used for storing and organizing information that is frequently used by the nodes located below the "Content" node. It usually contains images, animations, documents and other files. For example, it can be used to create an image gallery containing

images that are used in different news articles. The default identification number of the "Media" node is 43. The contents of this node can be viewed by selecting the "Media library" tab in the administration interface. By default, this node references a "Folder" object.

Users

The built-in multi-user solution makes use of the native content structure of eZ publish. An actual user is just an instance of a class that contains the "User account" datatype. The user nodes are organized within "User group" nodes below the "Users" top level node. In other words, this node contains the actual users and user groups. The default identification number of the "Users" node is 5. The contents of this node can be viewed by selecting the "User accounts" tab in the administration interface. By default, this node references a "User group" object.

Setup

The "Setup" node contains miscellaneous nodes related to configuration and is used internally. The default identification number of the "Setup" node is 48. By default, this node references a "Folder" object.

Design

The "Design" node contains miscellaneous nodes related to design issues and is used internally. The default identification number of the "Design" node is 58. By default, this node references a "Folder" object.

2.3.9 Node visibility

Since publishing means adding an object (by the way of a node) to the content tree, un-publishing would imply the removal of the object from the tree. Once an object is published, it can not be unpublished because eZ Publish does not provide such a feature. Instead, the system provides a hiding mechanism which can be used to change the visibility of nodes. The hide feature makes it possible to prevent the system from displaying the contents of published objects. This is achieved by denying access to the nodes. A single node or a subtree of nodes can be hidden either by a user or by the system. A node can have one of the following visibility statuses:

- Visible
- Hidden
- Hidden by superior

All nodes are visible by default and thus the objects they reference can be accessed. A user can hide or un-hide a node using the administration interface. Once a node is hidden, all its descendants will automatically be marked "Hidden by superior" and thus the descendants will also become hidden. A node can not become visible if its parent is hidden.

A hidden node will not be available unless the "ShowHiddenNodes" directive within the "[SiteAccessSettings]" block of a configuration override for "site.ini" is set to true. The most common way to use this setting is to disallow all but the administration interface to show hidden nodes.

Implementation

Each node has two flags: "H" and "X". While "H" means "hidden", "X" means "invisible". The hidden flag reveals whether the node has been hidden by a user or not. A raised invisibility flag means that the node is invisible either because it was hidden by a user or by the system. Together, the flags represent the three visibility statuses that were described above:

H	X	Status
-	-	The node is visible.
1	1	The node is invisible. It was hidden by a user.
-	1	The node is invisible. It was hidden by the system because its ancestor is hidden/invisible.

If a user tries to hide an already invisible node then the node's hidden flag will be set in addition to the invisible flag. If a node is hidden and its parent becomes visible, the node will remain hidden while the descendants will remain invisible. The following illustrations show how the node hiding algorithm works.

Case 1: Hiding a visible node

The following illustration shows what happens when a visible node is hidden by a user. The node will be marked hidden. Underlying nodes will be marked invisible (hidden by superior). The visibility status of underlying nodes already marked hidden or invisible will not be changed.

(see figure 2.15)

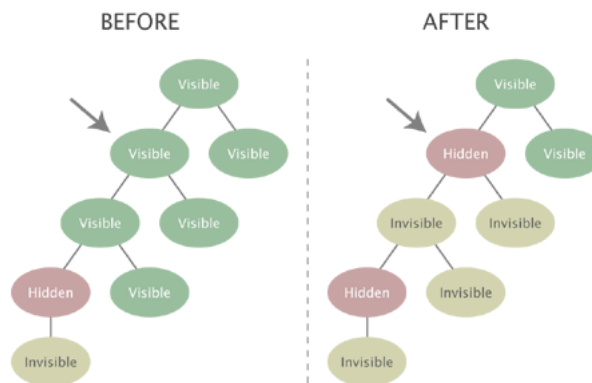


Figure 2.15: *Hiding a visible node*

Case 2: Hiding an invisible node

The following illustration shows what happens when an invisible node (hidden by superior) is explicitly hidden by a user. The node will be marked as hidden. Since the underlying nodes are already either hidden or invisible, their visibility status will not be changed.

(see figure 2.16)

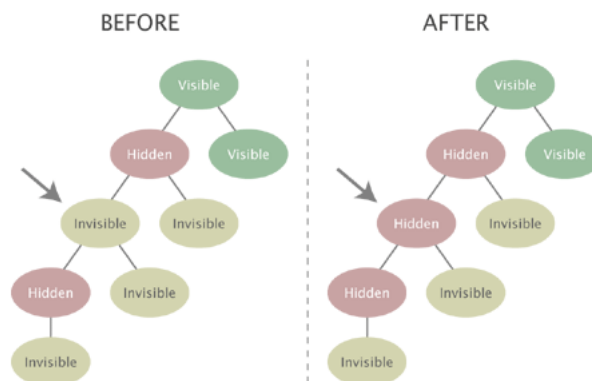


Figure 2.16: *Hiding an invisible node*

Case 3: Un-hiding a node with a visible ancestor

The following illustration shows what happens when a user un-hides a node that has a visible ancestor. Underlying invisible nodes will become visible. An underlying node that was explicitly hidden by a user will remain hidden (and its children will remain invisible).

(see figure 2.17)

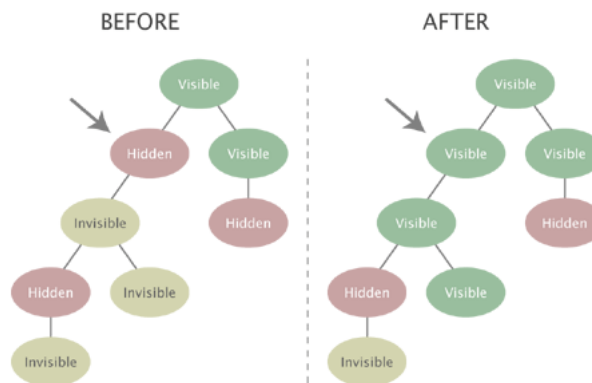


Figure 2.17: *Unhiding a node with a visible ancestor*

Case 4: Un-hiding a node with an invisible ancestor

The following illustration shows what happens when a user un-hides a node that has an invisible ancestor. Since the target node is un-hidden in a subtree that is currently invisible (because a node further up in the hierarchy has been explicitly hidden), the node will not become visible. Instead, it will be marked as invisible and will become visible when the hidden superior node is un-hidden.

(see figure 2.18)

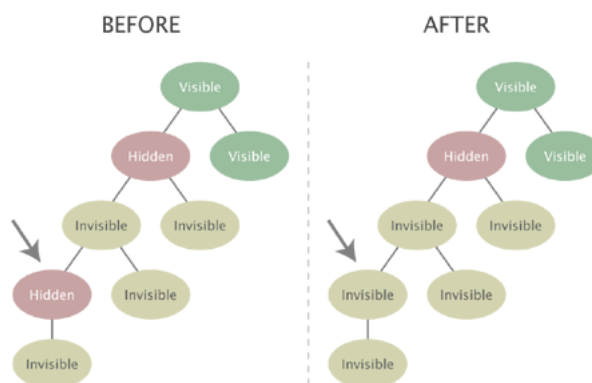


Figure 2.18: *Unhiding a node with an invisible ancestor*

2.3.10 Object relations

The content model of eZ Publish makes it possible to create relations between different objects. Any type of object can be connected to any other type of object. This feature is typically useful in situations when there is a need to bind and/or reuse information that is scattered around in the system.

For example, the concept of related objects makes it possible to add images to news articles. Instead of using a fixed set of image attributes, the images are stored as separate objects outside the article. These objects can then be related to the article and used directly in attributes represented by the "XML block" datatype. This approach is quite flexible because it does not enforce any limitations when it comes to the amount and the type of information that is to be included.

Relation types

A relation between two objects can be created either at the object level or at the object attribute level. The system stores the different types of relations using the same database table. An object can not have a relation to itself.

Relations at the object level

In eZ Publish 3.8 and earlier versions, the relations at the object level were generic and could not be grouped in any way. From 3.9, there are three types of relations at the object level:

- Common
- XML linked
- XML embedded

Common

A relation of the "common" type is created when a user manually adds a content object to the related object list of another object. (In most cases, this is done by using the "Related objects" window in the object edit interface.) This method is always available for use.

XML linked

Whenever an internal link (a link to other node or object) is inserted into an attribute represented by the "XML block" datatype, the system will automatically create a relation of the "XML linked" type. Note that a relation of this type is automatically removed from the system when the corresponding "link" tag is removed from the content.

XML embedded

Whenever an "embed" tag is inserted into an attribute of the "XML block" datatype, the system will automatically create a relation of the "XML embedded" type, i.e. relate the embedded object to the one that is being edited. Note that a relation of this type is automatically removed from the system when the corresponding "embed" tag is removed.

Relations at the attribute level

Relations of this type will be automatically generated whenever the "Object relation" or the "Object relations" datatypes are used. While the first one allows only a single relation, the second allows multiple relations. There is no grouping of the relations. However, by making use of several attributes that are represented by one of these datatypes, it is possible to create a custom structure with grouped relations.

2.3.11 Sections

A *section* is a number that can be assigned to an object. The section ID of an object denotes which section the object belongs to. Each object can belong to one section. By assigning different sections to objects, it is possible to have different groups of objects. Although the sectioning mechanism is implemented at the object level, it is more likely to be used in conjunction with the content node tree. This is why the administration interface makes it possible to manage sections on the node level. Using sections makes it possible to:

- Segment the node tree into different subtrees
- Set up custom template override rules
- Limit and control access to content
- Assign discount rules to a group of products

A default eZ Publish installation comes with the following sections:

ID	Name	Description
1	Standard	The "Standard" section is the default section. The "Content" top level node makes use of this section.
2	Users	The "Users" section is dedicated for user accounts and user groups that exist on the system. The "Users" top level node makes use of this section.
3	Media	The "Media" section is used by the "Media" top level node.
4	Setup	The "Setup" section is used by the "Setup" top level node.

Section definitions can be added, modified and removed using the administration interface. The following illustration shows an example of how the section feature can be used to segment the content node tree.

(see figure 2.19)

Behavior

When a new object is created, its section ID will be set to the default section (which is usually the standard section). When the object is published, it will automatically inherit the section that is assigned to the object encapsulated by the parent node. For example, if an object is created in

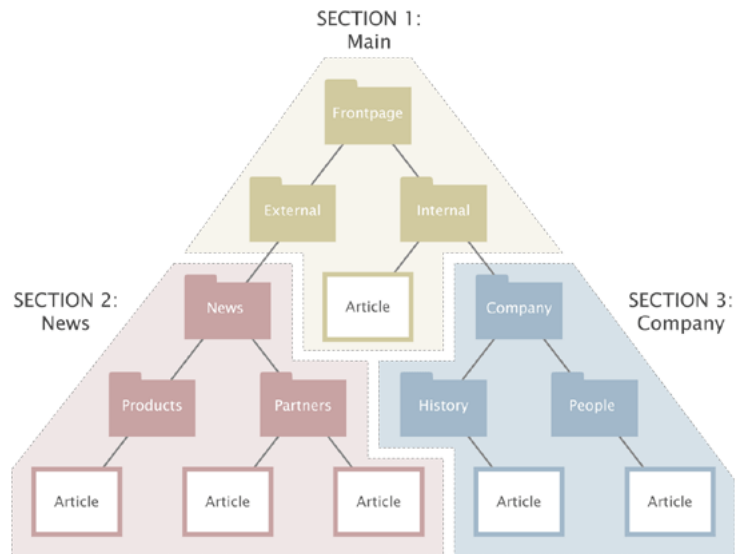


Figure 2.19: Example of sections.

a folder that belongs to section 13, the section ID of the newly created object will be set to 13. If an object has multiple node assignments then it is always the section ID of the object referenced by the parent of the main node that will be used. In addition, if the main node of an object with multiple node assignments is changed then the section ID of that object will be updated.

The administration interface makes it possible to assign sections to objects using the node tree. When a section is assigned to a node, the section ID of the object referenced by that node will be updated. In addition, the section assignment of all subsequent children of that node will also be changed. For example, if the section ID of a folder containing news articles is changed, then the section ID of the articles in that folder will also be changed.

The removal of sections may corrupt permission settings, template output and other things in the system. In other words, a section should only be removed if it is completely unused. When a section is removed, it is only the section definition itself that will be removed. Other references to the section will remain and thus the system will most likely be in an inconsistent state. The section ID numbers are not recycled. If a section is removed, the ID number of that section will not be reused when a new section is created.

2.3.12 URL storage

Every address that is input as a link into an attribute using the "XML block" or the "URL" datatype is stored in a separate part of the database. Actual data stored using these datatypes only contain references to entries in the separate URL table. This feature makes it possible to inspect and edit the published URLs without having to interact with the content objects. The addresses in the URL table can be checked by running the "linkcheck.php" script (which is also executed by the cronjob script) that comes with eZ Publish. This script will simply check if the links in the table actually work by accessing them one by one. If the target server of an URL returns an invalid response (404 Page not found, 500 Internal Server Error, 403 Access Denied, etc.) or if there is simply no response, the URL will be marked invalid.

Keep in mind that if an URL is marked as invalid by this cronjob, the `has_content` attribute for the matching attribute will return `FALSE`. The `has_content` attribute normally only returns `FALSE` if the attribute has no content.

Invalid URLs and the objects that are using them can be easily filtered out and edited using the "URL management" part of the administration interface. An entry in the URL table consists of the following data:

- ID
- Address
- Creation time
- Modification time
- Last checked
- Status

Every URL has a unique identification number. The address contains the actual link. The creation time is the exact date/time when the object containing that URL was published. The modification time is updated every time the URL is changed using the URL management part of the administration interface (and not when the object containing that URL is edited). Whenever a URL is checked by the script, the last checked field will be updated. The status of a URL can be either valid or invalid. By default, all URLs are valid. When the cronjob script is running, it will automatically update the status of the URLs. If a broken link is found, its status will be set to "invalid". Whenever an already existing URL is stored, the system will simply reuse the existing entry in the table.

Please note that the link check script must be able to contact the outside world through port 80. In other words, the firewall must be opened for outgoing HTTP traffic from the web server that is running eZ Publish.

2.3.13 Information collection

The information collection feature makes it possible to gather user input when a node referencing an information collector object is viewed. It is typically useful when it comes to the creation of feedback forms, polls, etc.

An object can collect information if at least one of the class attributes is marked as an information collector. When the object is viewed, each collector attribute will be displayed using the chosen datatype's data collector template. Instead of just outputting the attributes' contents, the collector templates provide interfaces for data input. The generated input interface depends on the datatype that represents the attribute. The following table reveals the datatypes that are capable of collecting information.

Datatype	Input interface	Input validation
Checkbox	Check-box.	No.
E-Mail	Single line of text.	Yes.
Option	Radio buttons or a drop-down menu.	No.
Text block	Multiple lines of unformatted text.	No.
Text line	Single line of unformatted text.	No.

The input interfaces must be encapsulated by an HTML form that posts the data using a submit button named "ActionCollectInformation" to "/content/action" (the "action" view of the "content" module). The submitted data will be stored in a dedicated part of the database, separated from but related to the object itself. In addition, whenever the object collects any data, the information can be sent to a specified E-mail address. The "Collected information" section within the "Setup" part of the administration interface can be used to view and delete information that was collected through content objects.

2.4 Configuration

This section explains the configuration model of eZ Publish. The default configuration files end with a ".ini" extension and are located in the "/settings" directory. Each file controls the behavior of a specific part of the system. For example, the "content.ini" file controls the behavior of the content engine, the "webdav.ini" file controls the behavior of the WebDAV subsystem, and so on. The main and most important configuration file is called "site.ini". Among other things, it tells eZ Publish which database, design, etc. that should be used. The default configuration files contain all the possible directives (with default settings) along with brief explanations. These files should only be used for reference. In other words, they should never be modified. The "Configuration files" section of the reference chapter contains a comprehensive explanation of the different configuration files and their settings.

File structure

An eZ Publish configuration file is divided into blocks, each block contains a collection of settings. The following example shows a part of the main (site.ini) configuration file.

```
...
# This line contains a comment.
[DatabaseSettings]
Server=localhost
User=allman
Password=qwerty
Socket=disabled
SQLOutput=enabled

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

The example above shows two blocks: "DatabaseSettings" and "ExtensionSettings". Each block has several settings which control the behavior of the system. A setting can usually be set to enabled/disabled, a string of text or an array of strings. If the name of a setting ends with a pair of square brackets, it means that the setting accepts an array of values. In the example above, the "ActiveExtensions" setting tells eZ Publish to use two different extensions: "ezdhtml" and "paypal". Lines starting with a hash are treated as comments.

Configuration overrides

As pointed out earlier, the default configuration files should never be modified because they will most likely be overwritten by a new set of files during an upgrade. Making a backup will still not be sufficient because the configuration settings change over time. For example, a previous

version of the files will not contain settings that were recently added. Because of these issues, custom configuration settings must be placed elsewhere. Global configuration overrides can be placed in the `"/settings/override"` directory. The settings of the configuration files located in this directory will override the default settings. The name of the configuration files in the override directory must end with one of the following extensions:

- `.ini.append`
- `.ini.append.php`

If an override configuration file exist with both `"/.ini.append"` and `"/.ini.append.php"` extensions, eZ Publish will process the one which ends with `"/.php"`. Because of possible security issues, the latter (`.ini.append.php`) should be used; specially if eZ Publish is running in a non virtual host environment. The `"/.php"` extension will trick the web server into handling the configuration file as a PHP script. If someone attempts to read it using a browser, the server will not display the contents. Instead, it will attempt to process it as PHP, which again will not produce any output since the configuration settings are commented out (see below). This method makes it more difficult for a hacker to get access to the configuration settings (for example the database password) by attempting to access one of the configuration files from outside. In order for this to work, the contents of the configuration file must be encapsulated by a pair of PHP comment markers: `/*` and `*/`. The following example shows how an override (for example `"/test.ini.append.php"`) should be set up:

```
<?php /* #?ini charset="utf-8"?  
  
# These are my example settings  
[ExampleSettings]  
ExampleSettingOne=enabled  
ExampleSettingTwo=disabled  
...  
  
*/ ?>
```

The `"/charset"` directive reveals the character set that was used to construct the ini file (usually UTF-8).

2.4.1 Site management

A single eZ Publish installation is capable of hosting multiple sites by making use of something called the *siteaccess* system. This system makes it possible to use different configuration settings based on a set of rules. The rules control which group of settings that should be used in a particular case. The siteaccess rules must be specified in the global override for the site.ini configuration file (`"/settings/override/site.ini.append.php"`).

Siteaccess

A collection of configuration settings is called a *siteaccess*. When a siteaccess is in use, the default configuration settings will be overridden by the settings that are defined for the siteaccess. Among other things, a siteaccess dictates which database, design and var directory that should be used (these are sometime referred to as "resources"). By making use of different siteaccesses, it is possible to combine different content and designs. A typical eZ Publish site consists of two siteaccesses: a public interface for visitors and a restricted interface for administrators. Both siteaccesses use the same content (same database and same var directory) but they use different designs. While the administration siteaccess would most likely use the built in administration design, the public siteaccess would use a custom design. The following illustration shows this scenario.

(see figure 2.20)

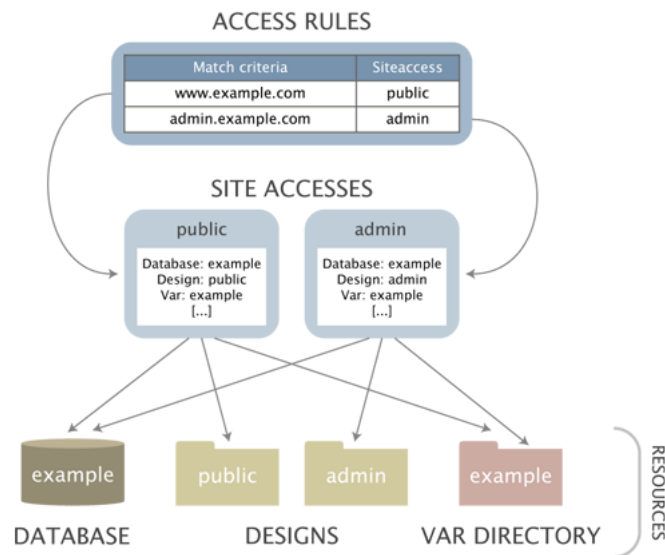


Figure 2.20: Example of a setup with two siteaccesses.

A siteaccess is nothing more than a set of configuration files that override the default settings when the siteaccess is used. A single eZ Publish installation can virtually host an unlimited number of sites by the way of siteaccesses. The configuration settings for a siteaccess are located inside a dedicated subdirectory within the `"/settings/siteaccess"` directory. The name of

the subdirectory is the actual name of the siteaccess. (Please note that siteaccess name should only contain letters, digits and underscores.) The following illustration shows a setup with two siteaccesses: admin and public.

(see figure 2.21)



Figure 2.21: Siteaccess directory example.

When a siteaccess is in use, eZ Publish reads the configuration files using the following sequence:

- 0 Default configuration settings - /settings/*.ini?
- 1 Active extension siteaccesses - /extension/my_extension/settings/siteaccess/my_site/*.ini.append.php
- 2 Siteaccesses - /settings/siteaccess/my_site/*.ini.append.php?
- 3 Active extensions - /extension/my_extension/settings/*.ini[.append.php]?
- 4 Global overrides - /settings/override/*.ini.append.php

In other words, eZ Publish will first read the default configuration settings. Secondly it will read my_site/*.ini.append.php to find the siteaccesses for the active extensions for the installation. Then it will determine which siteaccess to use based on the rules that are defined in the global override for "site.ini" ("/settings/override/site.ini.append.php"). When it knows which siteaccess to use, it will go into the directory of that siteaccess and read the configuration files that belong to that siteaccess. Next it will go into the configuration file for the active extensions and read the configuration files for the active extensions. The settings of the siteaccess will override the default configuration settings.

For example, if the siteaccess uses a database called "Amiga", the system will see this and automatically use the specified database when an incoming request is processed. Finally, eZ Publish reads the configuration files in the global override directory. The settings in the global override directory will override all other settings. In other words, if a database called "CD32" is specified in the global override for "site.ini" then eZ Publish will attempt to use that database regardless of what is specified in the siteaccess settings. If a setting is not overridden by either the siteaccess or from within a global override then the default setting will be used. The default settings are set by the ini files located in the "/settings" directory.

The following figure illustrates how the system reads the configuration files using the "site.ini" file as an example. As already mentioned, settings placed in the override files will be used instead of the default ones.

(see figure 2.22)

Rules for Online Editor and extensions

This applies for Online Editor and other extensions:

Translated versions of a site are not counted as additional site accesses

2.4.2 Extension siteaccess settings

The extension siteaccess settings makes it possible to place siteaccess specific settings in the extensions.

The directory structure must be as follows :

```
extension/<my_extension>/settings/siteaccess/<my_siteaccess>/<file.ini.append.php>
```

Example:

```
extension/ezno/settings/siteaccess/ezno/override.ini.append.php :
```

```
<?php /*  
  
[article_full_ezno]  
Source=node/view/full.tpl  
MatchFile=article/full.tpl  
Match[class_identifier]=article  
Subdir=templates  
  
*/ ?>
```

Note:

All settings except debug settings and including/activating extensions can be set this way.

2.4.3 Access methods

Based on a set of rules, eZ Publish determines which siteaccess it should use every time it processes an incoming request. The rules must be set up in the global override for the site.ini configuration file: `"/settings/override/site.ini.append.php"`. The behavior of the siteaccess system is controlled by the `"MatchOrder"` setting within the `[SiteAccessSettings]` block. This setting controls the way eZ Publish interprets the incoming requests. There are three possible methods:

- URI
- Host
- Port

The following text gives a brief explanation of the different access methods. Please note that the access methods can be combined. The documentation page of the `"MatchOrder"` directive reveals how this can be done.

URI

This is the default setting for the `"MatchOrder"` directive. When the URI access method is used, the name of the target siteaccess will be the first parameter that comes after the `"index.php"` part of the requested URL. For example, the following URL will tell eZ Publish to use the `"admin"` siteaccess: `http://www.example.com/index.php/admin`. If another siteaccess by the name of `"public"` exists, then it would be possible to reach it by pointing the browser to the following address: `http://www.example.com/index.php/public`. If the last part of the URL is omitted then the default siteaccess will be used. The default siteaccess is defined by the `"DefaultAccess"` setting within the `[SiteSettings]` block. The following example shows how to set up `"/settings/override/site.ini.append.php"` in order to make eZ Publish use the URI access method and to use a siteaccess called `"public"` by default:

```
...
[SiteSettings]
DefaultAccess=public

[SiteAccessSettings]
MatchOrder=uri
...
```

The URI access method is typically useful for testing / demonstration purposes. In addition it is quite handy because it doesn't require any configuration of the web server and the DNS server.

Host

The host access method makes it possible to map different host/domain combinations to different siteaccesses. This access method requires configuration outside eZ Publish. First of all, the DNS

server must be configured to resolve the desired host/domain combinations to the IP address of the web server. Secondly, the web server must be configured to trigger a virtual host configuration (unless eZ Publish is located in the main document root). Please refer to the "Virtual Host Setup" (page 26) part of the installation chapter for information about how to set up a virtual host for eZ Publish. Once the DNS and the web server is configured properly, eZ Publish can be set up to use different siteaccesses based on the host/domain combinations of the incoming requests. The following example shows how to set up `"/settings/override/site.ini.append.php"` in order to make eZ Publish use the host access method. In addition, it reveals the basic usage of the host matching mechanism.

```
...
[SiteAccessSettings]
MatchOrder=host
HostMatchType=map
HostMatchMapItems []=www.example.com;public
HostMatchMapItems []=admin.example.com;admin
...
```

The example above tells eZ Publish to use the "public" siteaccess if the requested URL starts with "www.example.com". In other words, the configuration files in `"/settings/siteaccess/public"` will be used. If the requested URL starts with "admin.example.com", then the admin siteaccess will be used. The example above demonstrates only a fragment of the host matching capabilities of eZ Publish. Please refer to the reference documentation for a full explanation of the "HostMatchType" directive.

Port

The port access method makes it possible to map different ports to different siteaccesses. This access method requires configuration outside eZ Publish. The web server must be configured to listen to the desired ports (by default, a web server typically listens for requests on port 80, which is the standard port for HTTP traffic). In addition, the firewall will most likely have to be opened so that the traffic on port 81 actually reaches the web server. The following example shows how to set up `"/settings/override/site.ini.append.php"` in order to make eZ Publish use the port access method. It also shows how to map different ports to different siteaccesses.

```
...
[SiteAccessSettings]
MatchOrder=port

[PortAccessSettings]
80=public
81=admin
...
```

The example above tells eZ Publish to use the "public" siteaccess if the requested URL is sent to the web server using port 80. In other words, the configuration files inside `"/settings/siteaccess/`

public” will be used. If the URL is requested on port 81 (usually by appending a :81 to the URL, like this: <http://www.example.com:81>), then the admin siteaccess will be used.

Source
Your
information

2.5 Modules and views

A *module* offers an HTTP interface which can be used for web based interaction with eZ Publish. While some modules offer an interface to kernel functionality, others are more or less independent of the kernel. The system comes with a collection of modules that cover the needs of typical everyday tasks. For example, the content module provides an interface that makes it possible to use a web browser to manage actual content. It is possible to extend the system by creating custom modules for special needs. Custom modules have to be programmed in PHP. The following table gives an overview of some of the most commonly used modules that come with eZ Publish.

Module	Description
Content	The "Content" module provides an interface to the content engine in the eZ Publish kernel. This module makes it possible to display, edit, search and translate the contents of objects, manage the node tree and so on.
User	The "User" module provides an interface to the user management system in the kernel. This module makes it possible to log users in and out of the system. In addition, it also provides functionality related to user registration, user activation, password changing, etc.
Role	The "Role" module provides an interface to the access control system in the kernel. This module makes it possible to create, modify and delete roles and policies. In addition, it provides functionality for assigning roles to different users and user groups.

Please refer to the "Modules" section of the reference chapter for a comprehensive list of all the built-in modules.

Module execution

Every time eZ Publish is accessed using a web browser, the client application indirectly interacts with one of the modules that are present in the system. The requested URL tells eZ Publish about which module it should execute in order to process the request. In particular, the first part of the URL reveals the name of the module. This is usually the part that comes directly after "index.php" unless the URI access method is used. The following example shows a typical eZ Publish URL:

```
http://www.example.com/index.php/content/edit/13/03
```

A quick glance at this URL reveals that the request is directed at the content module. Another typical example of an eZ Publish URL could be something like this:

```
http://www.example.com/index.php/user/login
```

By looking at the URL, we can immediately tell that eZ Publish will attempt to execute the user module when processing this request. Obviously, some additional information is also specified in the URLs. In the first example, the name of the module is followed by `"/edit/13/03"`. In the second example, the name of module is followed by `"/login"`. These additional strings control the behavior of the requested module and are explained below.

Module views

A module consists of a set *views*. 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. In the first example above, eZ Publish is instructed to access the `"edit"` view within the content module. In the second example, eZ Publish is instructed to access the `"login"` view within the user module.

When a view is called, eZ Publish starts up the program code that is associated with that view. Upon completion, the view returns a result to the module, which in turn returns it to the rest of the system. The result is put inside a template variable called `$module_result.content` which is available from the main template, the pagelayout. Please refer to the `"Template generation"` (page 173) section of the `"Templates"` chapter for more information about this part of the system.

View parameters

Some views support on one or more parameters. A *view parameter* makes it possible to pass information to the view itself and thus allows the view to be controlled from within the requested URL. The view parameters are appended after the name of the view in the URL. In the first example above, the following parameters are passed to the view: `"13"` and `"03"`. These parameters will instruct the edit view of the content module to provide an interface for editing the third version of the thirteenth content object in the system. The URL given in the second example does not make use of any view parameters. The view mechanism supports two types of parameters:

- Ordered parameters
- Unordered parameters

The ordered parameters have to be separated by slashes and they must come after the name of the view. In addition, they have to be provided in the same order as it is specified in the module's definition. For example, if the view parameters in the first example get mixed up, eZ Publish will attempt to edit the thirteenth version of object number three (instead of version number three of object number thirteen).

As the name suggests, the unordered parameters can be provided in an arbitrary order. If the view supports ordered parameters, the unordered parameters must come after the ordered parameters

If the view doesn't support ordered parameters, the unordered parameters will come directly after the name of the view in the URL. The unordered parameters must be provided in pairs. A pair consists of the parameter's name and value separated by a slash. The following example shows an imaginary eZ Publish URL with unordered parameters passed to the requested view:

```
http://www.example.com/index.php/video/dvd/button/play
```

The address in the example above tells eZ Publish to run the imaginary "video" module and execute the "dvd" view. A variable called "button" will be created and made available for the view code. The value of the variable will be set to "play". It is up to the PHP code of the view to discover this variable and to carry out a necessary sequence of actions.

POST variables

Some views make use of parameters that are submitted by the way of forms through the HTTP POST method. For example, the action view of the content module makes an extensive use of POST variables.

GET variables

Views can also make use of parameters that are submitted through the HTTP GET method. For example, parameters of the treemenu view within the content module are transferred using GET variables.

The default request

In order to be able to produce proper output, eZ Publish must know which module it should run and which view that should be executed. In other words, every URL has to contain at least the name of an existing module and a view. If an incomplete or mistyped URL is provided, eZ Publish will display an error page revealing what's wrong (missing/mistyped module or view). If the requested URL doesn't contain anything after "index.php" (except maybe a slash), the default module/view combination will be executed. The default module/view combination can be configured using the "IndexPage" setting under "[SiteSettings]" in an override for "site.ini". The default setting is "/content/view/full/2". It instructs eZ Publish to show a full view of node 2, the content top level node. In other words, if the following request is made:

```
http://www.example.com/index.php
```

...eZ Publish will behave as if the following URL was requested:

```
http://www.example.com/index.php/content/view/full/2
```

No redirection or page reload will be made, which means that the address field of the browser will remain unchanged.

2.6 URL translation

This section explains the different URL types that can be used with eZ Publish and how the URL translator works. By default, eZ Publish is capable of handling two types of URLs:

- System URLs
- Virtual URLs

System URLs

A *system URL* tells eZ Publish about which module that should be run and which view that should be executed. It may contain additional parameters/values that are passed to the view itself. Every system URL follows the same structure and can be broken down into the following components:

- Module name
- View name
- View parameters

The view parameters are optional and may consist of ordered and/or unordered values. A comprehensive description of the view parameters can be found in the "Modules and views" (page 150) section. The following model shows the required sequence of the different URL components:

```
http://www.example.com/index.php/<module>/<view>/[<ordered_view_parameters>]/
[<unordered_view_parameters>]
```

URL component	Description
Module	The name of the module that should be run.
View	The name of the view that should be executed.
Ordered view parameters	Some views make use of ordered parameters.
Unordered view parameters	Some views make use of unordered parameters.

The following example shows a typical system URL:

```
http://www.example.com/index.php/content/edit/13/3
```

By looking at the URL, we can tell that it will instruct eZ Publish to run the "content" module and execute the "edit" view. The values "13" and "3" are parameters that will be passed to the view itself. Please note that the exact style of the URLs depend on the access method (page 161) that is used and the way the web server is configured. For example, the web server can be configured to hide away the "index.php" part of the address.

Virtual URLs

A *virtual URL* (also known as *URL alias* or *nice URL*) is nothing more than an alias for an existing system URL. Virtual URLs are nicer, easier to remember and sometimes shorter than system URLs. While system URLs reveal a great deal about what eZ Publish is instructed to do, virtual URLs do not reveal any system specific information at all. A virtual URL can not be broken down to components in the same way as a system URL. The following example shows a typical virtual URL:

```
http://www.example.com/company/about
```

There are actually two types of virtual URLs, ones that are automatically generated and maintained by eZ Publish and ones that are created and maintained by the site administrator. However, all virtual URLs are treated equally and thus they are handled in the same way.

From 3.10, multilingual virtual URLs (page 319) are supported. The system keeps track of the URLs in a table which basically consists of three columns:

Virtual address	Action	Language mask
company/about	eznode:46	2

An actual URL using the virtual address in the table above could be the following:

```
http://www.example.com/company/about
```

According to the table above, the virtual URL will be translated internally to the following system URL:

```
http://www.example.com/content/view/full/46
```

Both URLs are perfectly valid and will produce the exact same output, in this case a full view of node number 46. When the virtual URL is used, the redirection/mapping will be done internally and thus the user will reach the target node without any glitches in form of re-directions, page reloads, etc. The language mask field is used internally by the system to identify which languages the alias is associated with (based on the same bit-field algorithm as for content objects).

If the site administrator creates a virtual URL for accessing the "content/search" interface, the system will add a new entry to the table:

Virtual address	Action	Language mask
findme	module:content/search	4

An actual URL using the virtual address in the table above could be the following:

```
http://www.example.com/findme
```

According to the table above, the virtual URL will be translated internally to the following system URL:

`http://www.example.com/content/search`

Automated virtual URL generation and maintenance

Every time an object is published, the system will automatically generate a virtual URL for each of the object's node assignments. If an object exists in several languages, the system will generate virtual URLs for all translations. The generated URL for a node is based on the node's location in the tree and the name of the object that the node encapsulates. The virtual URLs generated for the nodes are handled completely by the system and can not be changed using the administration interface. The following illustration shows an example of objects, nodes and corresponding URLs.

(see figure 2.23)

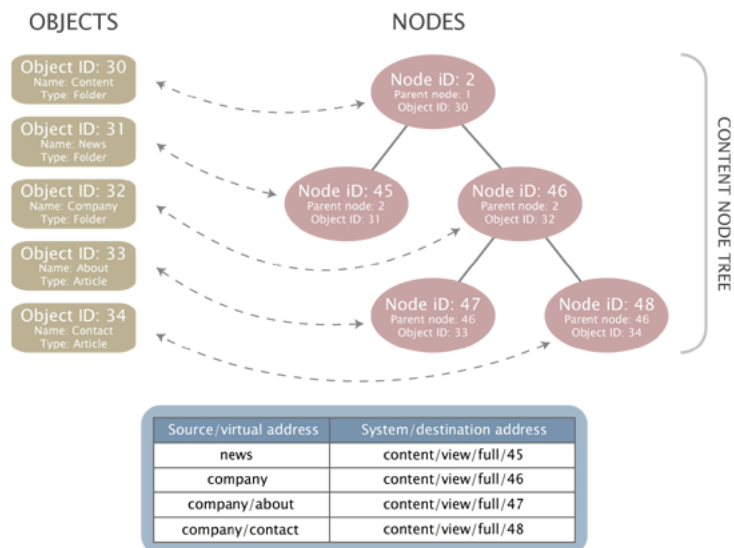


Figure 2.23: Objects, nodes and nice URLs.

The example above clearly demonstrates how the virtual URLs are generated. For each node, the system generates a path of strings separated by slashes. The strings in the path are the names of the objects that are referenced by the nodes up to and including the target node.

From 3.10, it is possible to enable Unicode support for the URLs and thus no transliteration needs to be performed since most characters are allowed. If there are two nodes with identical or almost identical names within the same location, the system will generate unique URL aliases for newly introduced conflicting nodes by attaching numbers to their URL aliases (for example, "Company", "Company2", "Company3" and so on).

When the name of an object is changed, the system will take care of changing the virtual URLs for the involved nodes. In addition, an internal redirection will be created, which will make sure that the old URL still works. The old virtual URL will keep working until the exact same URL needs to be generated for a node. In this case, the old virtual URL will be deleted.

Manual virtual URLs

It is possible to manually add, edit and remove virtual URLs using the administration interface (both global aliases and node URL aliases). Refer to "Managing URL aliases (page 326)" for more information. In addition, wildcard based URL forwarding (page 319) is supported. (This feature was removed when implementing the multilingual URLs functionality for eZ Publish 3.10.0 and then re-added in later versions.)

2.7 Designs

This section explains the concept of designs and how eZ Publish handles different designs. As mentioned in the beginning of this chapter, design is all about the way actual content is marked up and visually presented. When talking about a design, we're talking about the things that make up a web interface: HTML, style sheets, images that are not a part of the content, etc. All files that are related to appearance reside in the "design" directory. An eZ Publish installation is capable of handling a virtually unlimited number of designs. Each design has its own dedicated sub-directory within the main design directory. The name of a sub-directory also functions as the actual name of a design. A typical eZ Publish design consists of the following components:

- CSS files
- Image files
- Font files
- Template files

Among other things, a siteaccess dictates which design that should be used. By making use of different siteaccesses, it is possible to combine different content and designs. A typical eZ Publish site consists of two siteaccesses: a public interface for visitors and a restricted interface for administrators. Both siteaccesses use the same content (database and var directory) but they use different designs. In particular, the administration siteaccess would most likely use the built in administration design. The public siteaccess would use a custom design.

Default designs

An eZ Publish distribution comes with at least two default designs:

- admin
- standard

The "admin" directory contains all design related files that make up the built in administration interface. The "standard" directory contains a set of standard/default design related files such as the default/standard templates, images, etc. The contents of these directories should not be tampered with. Instead, custom designs should be used (if/when necessary). A custom design can be added by creating a new sub-directory within the main "/design" directory.

Design directory structure

All files that belong to a specific design are located inside the directory of that design. The name of the directory also functions as the name for the design itself. An eZ Publish design directory typically contains the following sub-directories:

Subdirectory	Description
fonts	Font files used by the "texttoimage" template operator which is capable of visualizing text using truetype fonts.
images	Non-content specific images (banners, logos, graphical layout elements, etc.).
override	Custom templates that will be used by instead of the default/standard templates. These files will be triggered by template override rules that are specified in a configuration override for "override.ini". Please refer to "The template override system" (page 229) section of the "Templates" chapter for more information about this feature.
stylesheets	CSS files.
templates	Main template(s) (for example the pagelayout, header, footer, etc.) and custom templates that will be used instead of the standard/default templates.

2.7.1 Design combinations

A siteaccess may make use of several designs. This means that the final result generated by eZ Publish (the actual HTML) can be a combination of files originating from various designs. A siteaccess is capable of using a combination of the following:

- One main design
- None or several additional designs
- One standard design

A siteaccess should always have at least a main design and a standard design. While the main design can be set to anything, the standard design should not be modified. The default configuration is to use the built-in standard design. It ensures that eZ Publish always finds the necessary templates and thus any kind of content can be rendered without problems. A more in-depth explanation is presented below.

Automatic fallback

If eZ Publish is unable to find a design specific file (a stylesheet, a template, an image, etc.) within the main design, it will automatically attempt to locate the file elsewhere. The system will sequentially go through all the additional designs (if specified), looking for the requested file. At last, if the requested file still hasn't been found, eZ Publish will attempt to locate the missing file within the standard design. The following diagram illustrates this functionality.

(see figure 2.24)

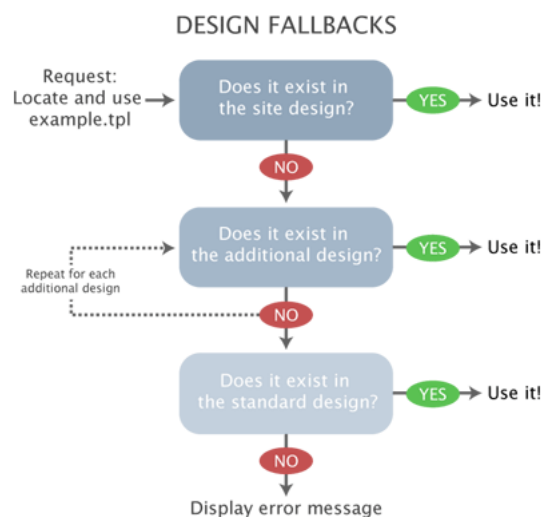


Figure 2.24: *The design fallback mechanism.*

Configuration

The different designs to be used by must be defined in the "[DesignSettings]" block within an override for the "site.ini" configuration file. The following directives can be used:

- SiteDesign
- AdditionalSiteDesignList
- StandardDesign

The "SiteDesign" directive specifies the main design. The "AdditionalSiteDesignList" directive specifies an array of additional site designs. The "StandardDesign" directive specifies the standard design. Even though it is possible to change the standard fallback design, it is not a good idea to do so. The "StandardDesign" directive should always be set to the built-in standard design. This is already defined in the default "site.ini" file and thus there is no need to set the standard design from within an override. If there is a need for a custom fallback design, it should be specified using the "AdditionalSiteDesignList" setting. The automatic fallback mechanism opens up for a lot of possibilities and flexibility. For example, it makes the reuse and combination of designs an easy matter.

Example

The following example shows how to configure the following design settings in an override for the "site.ini" configuration file:

- "my_design" should be the main design
- "fallback_one" should be the first additional design
- "fallback_two" should be the second additional design
- "standard" should be the standard fallback design

```
...  
[DesignSettings]  
SiteDesign=my_design  
AdditionalSiteDesignList []=fallback_one  
AdditionalSiteDesignList []=fallback_two  
StandardDesign=standard  
...
```

In this particular case, if eZ Publish is unable to find the requested file within the main design "my_design", it will automatically fallback to the additional designs. At first, the system will look for the requested file within the "fallback_one" design directory. If the requested file is not found, the system will look in the "fallback_two" design directory. If the file still hasn't been found, the system will attempt to locate it within the "standard" design directory. The standard directory will most likely contain the requested file (unless a custom template/override is requested).

2.8 Access control

This section explains how eZ Publish manages user accounts and access permissions. The system comes with a built-in access control mechanism that can be used to limit access to content or to certain functions. The access control system is based on the following elements:

- User
- User group
- Policy
- Role

The following illustration shows the relations between the elements in the list above.

(see figure 2.25)

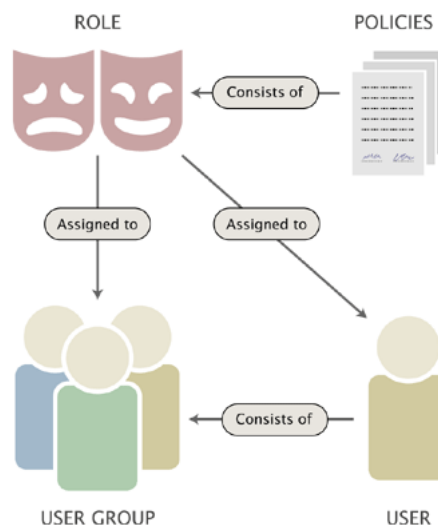


Figure 2.25: Users, groups, policies and roles.

A *user* defines a valid user account on the system. A *user group* consists of users and other user groups. A *policy* is a rule that grants access to content or a certain system function. For example, a policy may grant read access to a collection of nodes. A *role* is a named collection of policies. A role can be assigned to users and user groups. The following text gives a more in-depth explanation of the user/group/policy/role elements.

User

An actual user account is represented by a content object (with at least one node assignment) that contains information about a specific user. The default "User" class allows the storage of

the following elements: first name, last name, E-mail, username and password. The last three elements (E-mail, username and password) are provided by the "User account" datatype. This is a special datatype which plugs more deeply into the system. Instances of any content class containing the "User account" datatype will function as valid users on the system. In other words, if there is a need to store additional information about users, it is possible to either modify the default user class or to create a custom class that contains the datatype.

Enabled and disabled user accounts

The user accounts can be enabled or disabled from within the administration interface. When disabled, an account will continue to exist, but the user will not be able to log in until the account is enabled. Newly created accounts are enabled by default.

Locked and unlocked user accounts

In addition to being enabled and disabled, user accounts can be locked and unlocked. An account will be automatically locked by the system if the maximum number of failed login attempts is exceeded. A failed login attempt is a combination of a valid user name and an invalid password. Once an account is locked, its owner will not be allowed to log in until the account is either unlocked by another user with administrator privileges or if the login request is coming from a trusted IP address / range.

The number of failed log-in attempts are stored in a database table called "ezuservisit". An account's failed login counter is automatically reset upon a successful log-in. In other words, as long as you log in with a valid username/password combination, the failed log-in attempt counter associated with your account will be zero.

E-mail

Note that the default configuration does not allow different users to be registered with the exact same E-mail address. This is just a built-in precaution mechanism which can be easily turned off by setting the "RequireUniqueEmail" directive within the [UserSettings] block of a configuration override for "site.ini" to "false".

User ID

Every user has a unique identification number which is the same as the ID number of the actual object that represents the user account. Among other things, the user IDs are used by other objects on the system. In particular, an object contains references (by the way of user IDs) to the initial creator and to all users who have created versions within that object. Removing a user account might lead to an inconsistent state where objects have owner/modifier references to non-existing user accounts. Because of this, it is not recommended to remove users from the system, the accounts to be removed should be disabled instead.

User group

A user group is a content object (with at least one node assignment) that contains user accounts and other user groups. In other words, a user group is just a collection of users (similar to a directory containing files and sub-directories on a file system).

Policy

A policy is a rule that grants access to a specific function or all functions of a module. A policy consists of the following elements:

- Module name
- Function name
- Function limitation

The module name reveals the actual module that the policy grants access to. The function name specifies which function the policy should be limited to. A policy can either be restricted to a single function or grant access to all functions of a module. A module can have none or several functions. The functions are assigned to the module's views and thus the access requirements for a view are controlled by the functions that are assigned to that view. The function-view assignments can not be tampered with from within the administration interface. A policy granting access to a module's function can be further restricted by the way of function limitations. This can only be done if the function itself supports limitations. A function may support none, one or several limitations. The following table shows an overview of the available function limitations.

Limitation	Description
Class	The "Class" limitation makes it possible to limit a policy to objects of certain types.
Group	The "Group" limitation makes it possible to limit a policy to objects that are owned by a group.
Language	The "Language" limitation makes it possible to limit a policy to object versions in specific languages.
Node	The "Node" limitation makes it possible to limit a policy to a specific node.
Owner	The "Owner" limitation makes it possible to limit a policy to objects that are owned by the user who is logged in.
Parent class	The "Parent class" limitation makes it possible to limit a policy based on the type of the object referenced by the parent node.
Section	The "Section" limitation makes it possible to limit a policy to objects that are assigned to

	certain sections.
Siteaccess	The "Siteaccess" limitation makes it possible to limit a policy to a certain siteaccess.
Status	The "Status" limitation makes it possible to limit a policy to a certain version status (published, archived, etc.).
Subtree	The "Subtree" limitation makes it possible to limit a policy to a certain part of the content node tree.

Role

A role is a named collection of policies. A role can be assigned to users and user groups. It is possible to assign a role with additional limitations. The role limitation feature is typically useful in a case where multiple users with similar permissions have to manipulate different parts of the content node tree. Instead of creating a role for each user, the site administrator can create a generic role and assign it with different limitations to the different users. The role limitations will override the limitations of the role's policies. The following table shows an overview of the available role limitations.

Limitation	Description
Section	The "Section" limitation makes it possible to limit a role to objects that are assigned to certain sections.
Subtree	The "Subtree" limitation makes it possible to limit a role to a certain part of the content node tree.

2.9 Workflows

This section explains the workflow capabilities of eZ Publish. The system comes with an integrated workflow mechanism that makes it possible to perform different tasks with or without user interaction. The workflow implementation is based on the following components:

- Events
- Workflows
- Workflow groups
- Triggers

The following illustration shows the relations between the elements in the list above.

(see figure 2.26)

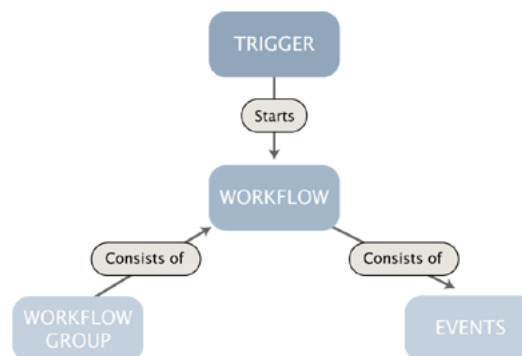


Figure 2.26: *The workflow system.*

An *event* is the smallest entity of the workflow system, it carries out a specific task. eZ Publish comes with a collection of events that cover the needs of typical everyday tasks. For example, the built-in approve event makes it possible to have the contents of an object approved by an editor (a user) before it is published. The built-in events are documented in the "Workflow events" section of the "Reference" chapter. It is possible to extend the system by creating custom events for special needs. Custom workflow events have to be programmed in PHP.

A *workflow* is a collection of events. In other words, it defines an ordered sequence of actions that will be executed when the workflow is running. The workflows can be placed in different groups. A *workflow group* is nothing more than a collection of workflows. A workflow is initiated by a *trigger*. Although a trigger is only capable of initiating a single workflow, several other workflows can be started through the built-in multiplexer event (from within the workflow that was originally initiated by the trigger). A trigger is associated with a function of a module. It will start the specified workflow either before or after that the function has completed. The following table gives an overview of the standard/built-in triggers.

ID	Module	Function	Connection type
1	content	publish	before
2	content	publish	after
3	shop	confirmorder	before
4	shop	checkout	before
5	shop	checkout	after

2.10 Webshop

This section explains the e-commerce capabilities of eZ Publish. The system comes with an integrated shop mechanism that plugs directly into the object / node tree model.

Note: From 4.3 onwards the eZ Webshop tab is disabled by default in the eZ Publish installation. Follow this [link](#) to see how you enable the eZ Webshop functionality in eZ Publish.

Note 2: As of version 4.5 the webshop functionality has been deprecated and will be moved out of the kernel into an extension in a future version.

The webshop functionality is built around the following components:

- Products
- Value Added Taxes (VATs)
- Discount rules
- Wishlist
- Basket
- Orders

The following illustration shows how the different components interconnect and work together.

(see [figure 2.27](#))

An actual product is represented by a content object (with at least one node assignment) that contains information about the product itself along with a price. The price must be represented by an attribute that makes use of the built-in price or multi-price datatype. These are special datatypes which plug more deeply into the system. The main difference is that the price datatype allows to specify only one price value for each object (simple price product) whereas the multi-price datatype makes it possible to specify several price values in different currencies for each object (multi-price product). A content class can only contain one price attribute or one multi-price attribute. There is no way to have a simple price product and a multi-price one in the shopping basket at the same time and it is not recommended to use both price and multi-price datatype on your site.

The price can be affected by a value added tax and/or a discount rule. A discount rule can be configured to reduce the price of certain products by a percentage. The products can be put into a user's wishlist and/or shopping basket. A user's wishlist and basket can be modified at any time. The contents of the shopping basket can be purchased by initiating the checkout process. Once the checkout process is completed, an order will be created. The system will automatically notify the site administrator and the user who placed the order by sending out E-mails. A list of placed orders and sales statistics can be viewed using the administration interface. An order is assigned a status which may be changed by a user with sufficient permissions. A status log is kept for each order.



Figure 2.27: *The integrated e-commerce solution.*

Value added taxes

The system allows the site administrator to set up different kinds of value added taxes (VAT types). A VAT type consists of a name and a percentage. The administration interface makes it possible to add, remove and modify VAT types. The VAT types are used by the price and multi-price datatype. There is an additional possibility to create VAT charging rules which instruct the system to charge VATs according to the product category and the buyer's country. (Please refer to the "VAT charging system" section of the "Features" chapter for more information.)

The price datatype

As pointed out above, a product is nothing more than a content object with a price. The price can be represented by an attribute that makes use of the built-in price datatype. Instances of any class containing the price datatype will automatically be treated as simple price products. A class attribute represented by the price datatype makes use of one of the predefined VATs. There are two ways in which the selected VAT can be used. This configuration depends on how the product prices are entered when the objects are created. The first alternative (Price inc. VAT) is to be used if the prices that are entered already include the value added tax. The second alternative (Price ex. VAT) should be used if the prices that are entered do not contain the value added tax. When the first alternative is used and the product is viewed, the price that was entered will be

shown. When the second alternative is used and the product is viewed, the price will be the price that was entered plus the VAT. When the object is in the basket and the basket is viewed, it is possible to see the price of the products with and without the VATs (regardless of which approach that was used).

Please note that the price datatype allows to set only one price value for each product (the system will use your locale currency for when displaying this price). This datatype does not work with multiple currencies.

The multi-price datatype

The price can be represented by an attribute that makes use of the built-in multi-price datatype. This datatype allows you to set several prices in different currencies for each product independently of your locale currency. Instances of any class containing the multi-price price datatype are automatically treated as multi-price products. (Please refer to the "Multi-currency" section of the "Features" chapter for more information.) This datatype interacts with VATs in the same way as the price datatype.

Discount rules

The final price of a product can be affected by a discount rule. A discount rule can be configured to reduce the price of certain products by a percentage. The discount rules can be placed in different discount rule groups and are always active (there is no way to turn them on/off). The discount rule groups make it possible to choose which group(s) of customers will be affected. This can be done by assigning a discount rule group to the target user group(s).

By default, a newly created discount rule affects all the products that are in the system. However, a discount rule can be easily limited to a group of products. A discount rule can be limited in two ways, which are mutually exclusive. The first alternative is to use a combination of the "Product type" and the "Section" limitations, which are described in the table below.

Limitation	Description
Product type	The "Product type" limitation makes it possible to limit a policy to products/objects of certain types (only classes that make use of the price datatype will be shown). The default setting is "Any", which means that it will affect all kinds of product objects.
Section	The "Section" limitation makes it possible to limit a policy to products/objects that are assigned to certain sections. The default setting is "Any", which means that the discount rule will affect product objects in all sections.

The second alternative is to add individual products to the discount rule's product list. When the individual product list is used, the "Product type" and "Section" limitations will be omitted and thus only the products that are in the list will be affected.

Shop related datatypes

The following table shows the datatypes that plug in to the e-commerce subsystem of eZ publish.

Datatype	Description
Price	When used as an attribute in a content class, the "Price" datatype connects the instances (objects) of that class to the webshop system. As soon as an object has a price attribute, users can put the object in their baskets and/or wishlists. This datatype allows to set only one price value for each product (the system will use your locale currency for this price).
Multi-price	When used as an attribute in a content class, the "Multi-price" datatype also connects the instances (objects) of that class to the webshop system. As soon as an object has a multi-price attribute, users can view its price in different currencies, put the object in their baskets and/or wishlists. Objects without a price or multi-price attribute can not be put into a user's basket and/or wishlist and thus they are not connected to the e-commerce subsystem. The multi-price datatype allows you to set several prices in different currencies for each product.
Option	The "Option" datatype makes it possible to create a single group of options for each content object. Each option can be assigned a short text and an additional price. For example, it can be used to sell T-shirts in different colors where the price is different for some (or all) colors.
Multi-option	Note that this datatype should no longer be used. It was deprecated in eZ Publish 3.10 and is replaced by the "Multi-option2" datatype (see below). The "Multi-option" datatype makes it possible to create multiple groups of options for each content object. Each option can be assigned a short text and an additional price. This datatype works in the same way as the "Option" datatype. The only difference is that instead of supporting only one group of options, it allows the creation of multiple groups

	of options for each content object.
Multi-option2	The "Multi-option2" datatype makes it possible to create multiple and distinctive groups of multi-options for each content object. The multi-options can be nested. For each option, you can specify an additional price, an image, whether the option should be the default selection and if it should be possible to select it (sometimes you wish to force the selection of a set of options without providing a default selection for the user; in that case, the first option can be set to something like "Make a choice"). In addition, this datatype makes it is possible to set up rules for allowing/disallowing certain combinations of options.
Range-option	The "Range-option" datatype makes it possible to create a single group of enumerated options for each content object. For example, it can be used in a scenario where the goal is to sell shoes of different sizes and the size does not affect the price. For each content object, the administrator needs to set up the available range (if any).

Chapter 3

Templates

The purpose of this chapter is to reveal and teach everything there is to know about the template system. It describes both the template language and the way the system handles the template files. People previously unfamiliar with eZ Publish templates should be able to collect enough information in order to understand the following issues:

- What a template is and what it is not
- Template types (page layout, node and system templates)
- Template structure
- The template language
- The main template (the page layout)
- Template variables available in the page layout
- How basic template tasks can be done
- How information can be retrieved from the Content Management System
- The template override system

3.1 Template basics

This section explains the concepts behind templates and the template system. eZ Publish uses templates as the fundamental unit of site design. A *template* is basically a custom HTML file that describes how some particular type of content should be visualized. A template file always ends with a ".tpl" extension. Actual HTML code in the built-in/default templates follow the [XHTML 1.0 Transitional](#) specification. In addition to standard HTML syntax, a template consists of eZ Publish specific code. The eZ publish specific code makes it possible to extract information from the system and to solve common programmatic issues like for example conditional branching, looping, etc. All eZ Publish specific code must be placed inside a set of curly brackets, "{" and "}". The following example shows a part of a template that prints out the current time:

```
...
<h1>Time machine</h1>
<p>
    The current time is: {currentdate()|l10n( time )}
</p>
...
```

The example above demonstrates how standard HTML is mixed with eZ Publish specific code. It shows the usage of the "currentdate" and the "l10n" template operators. Since "currentdate" returns a UNIX timestamp, it must be formatted using the "l10n" localization operator (or else the output would not make any sense to humans). This is done by piping the output from the "currentdate" operator into the "l10n" operator, which will output the requested information according to the current locale settings. The "time" parameter tells the operator to output only the time (it could have been "date", "shortdate", "datetime" and so on).

Template generation

The template system is component based. In other words, an actual HTML page is usually made up of several templates. At the minimum, eZ Publish always renders the main template, which is called *pagelayout*. The page layout contains the HTML, HEAD and BODY tags; it dictates the overall look of a site. Among other things, it describes the visual structure (main layout, logo, main menu, footer, etc.) that will be presented for each HTML page that the system generates.

Every incoming request tells eZ Publish to run a specific module and to execute one of the module's views. When finished, the requested module/view combination will generate a result. The result can be accessed through the `$module_result` array which is available in the page layout template. The following illustration shows a simplified 3-step explanation of how eZ Publish responds to an HTTP request.

(see [figure 3.1](#))

Every view generates a chunk of HTML code by making use of a template. Templates that are used by views are often referred to as *view templates*. Whenever a view has finished running, it will issue an internal template request. The requested template will be interpreted, processed and thus converted to HTML. After processing, the system will put the resulting HTML in the

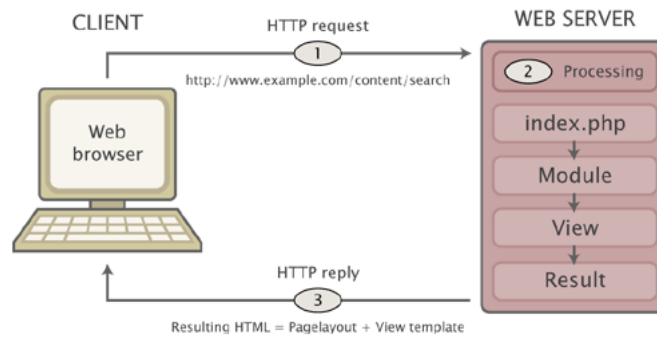


Figure 3.1: Client - server cycle.

module's result array. The module/view's result can be accessed through the ".content" extension: `{ $module_result.content }`. By printing out the contents of this variable, it is possible to include the HTML code that was generated by the view in the page layout. The following illustration shows how the module/view result (generated by different modules/views - depending on the request) is included in the page layout:

(see figure 3.2)



Figure 3.2: The module result as a part of the pagelayout.

View templates

A template used by a view can either be a *node template* or a *system template*. A node template will only be used when a node is being viewed, for example when a system URL containing `"/content/view"` or the virtual URL of a node is requested. A system template typically provides an HTML interface to a specific eZ Publish feature. For example, the template used by the "search" view of the "content" module provides an interface to the built-in search engine.

The difference between the template types mentioned above is the available variables and the combination of override rules that can be used. A node template (page 176) gives access to a vari-

able (`$node`) which contains information about the actual node that is being viewed. Depending on the view that was called, a system template (page 178) typically gives access to several variables. A template override rule makes it possible to display custom templates in specific cases. The override rules for node templates are much more flexible than the override rules for system templates. For example, it is possible to set up complex rule combinations that depend on the type of the node being viewed, the depth of the node in the tree, the section which the node's object is assigned to and so on. Please refer to the "The template override system" (page 229) section for a detailed description of the template override mechanism.

3.1.1 Node templates

Whenever eZ Publish is requested to output information about a node (either by a system URL or a virtual URL), it executes the "view" view of the "content" module. If a system URL is used, both the desired view mode and the target node must be specified in the URL. If a virtual URL is used, eZ Publish will automatically know which node that should be accessed by looking up the corresponding system URL in the internal URL table. When a virtual URL is used, the system will always use the full view mode.

The templates for the different view modes must be placed inside the "/templates/node/view/" directory of a design. If the requested file is not found within the main design of the siteaccess, the system will search for it in the additional designs and the standard design. Please refer to the documentation of the automatic fallback system (page 159) for more information about this feature. The "/templates/node/view" directory of the standard design contains templates for different view modes. A basic custom design typically contains a page layout and a full view template. The following illustration shows the locations of these templates in a custom design called "example".

(see figure 3.3)

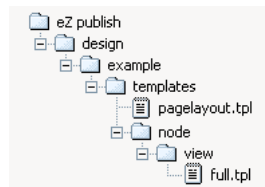


Figure 3.3: Location of pagelayout and full view template in example design.

When a node is requested (and there are no template override rules for node templates), eZ Publish will generate a page that is built up of the following templates:

(see figure 3.4)

Custom node templates

A typical eZ Publish site always makes use of custom node templates. The main reason for this is because there is almost always a need for displaying the various types of nodes in different ways. For example, information pages need to look different than news articles; the welcome page has to be formatted in a special way, and so on. Unlike custom system templates (which are mostly just modified copies of the standard templates placed in a custom design), custom node templates are created as override templates. The override templates are triggered by the template override system. This system offers a flexible mechanism that can be programmed to use different templates based on various conditions. For example, it can be programmed to use a template called "article.tpl" when the system is requested to show the contents of nodes referencing article objects and at the same time show "special_article.tpl" when a specific article is accessed. Note that override templates (in this case "article.tpl" and "special_article.tpl") must



Figure 3.4: *Pagelayout + node view full template.*

be placed in the "override/templates" directory of the main design used by the siteaccess. Please refer to the documentation of the template override system (page 229) for more information about how this mechanism actually works and how it can be used to trigger override templates.

The \$node variable

Whenever the system makes use of a node template (regardless of the view mode, the target node and if the template is an override or not), a variable called `$node` will be available in the template that is used. This variable is automatically set by the system and it contains an `ezcontentobjecttreenode` object that represents the requested node. This variable allows the extraction and display of various information about the node and the object that it encapsulates. Please refer to "Outputting node and object data" (page 226) for information about how to display node/object data.

3.1.2 System templates

Whenever eZ Publish is requested to do something else than displaying a node (in other words the URL does not contain `"/content/view"` or isn't the virtual URL of a node), it will use a system template. There are two main differences between system templates and node templates:

- System templates provide access to various variables (depending on the view that was requested). A node template only provides access to a `$node` variable representing the node that was requested.
- The override rules for node templates are much more flexible than the override rules for system templates.

An eZ Publish distribution provides default templates for all views. These templates are located in the `"templates"` directory of the standard design. A view typically uses a template that is located in a subdirectory that has the same name as the module which the view belongs to. The name of the template is usually the same as the name of the view (with a `".tpl"` extension). For example, the `"login"` view of the `"user"` module looks for a template called `"login.tpl"` inside a directory called `"user"`. Another example would be the `"basket"` view of the `"shop"` module. This view looks for a template called `"basket.tpl"` within the `"shop"` directory.

Custom system templates

Although eZ Publish provides all the necessary system templates (by the way of the standard design), a typical eZ Publish site always makes use of customized system templates. The main reason for this is because the default templates usually need to be tailored in order to fit perfectly in with the style of a custom design. Unlike custom node templates which are mostly provided using the template override system, custom system templates are usually just modified copies of the standard templates located in the custom design. These are not connected with the override system and must be placed in the `"templates"` directory of a custom design (not in the `"override/templates"` directory). For example, a custom template for the `"login"` view of the `"user"` module in a design called `"example"` would be `"/design/example/templates/user/login.tpl"`. A custom template for the `"search"` view of the `"content"` module would be `"/design/example/templates/content/search.tpl"`.

Design combinations

As mentioned in the text above, a custom design typically contains a set of customized system templates. However, creating a custom design that provides templates for all possible scenarios would be too much / unnecessary work. This is why the standard design always should be used as the last fallback resort. The automatic fallback system makes it possible to combine several designs so that the main design (which is usually a custom design) does not have to provide all the necessary templates. Whenever eZ Publish is unable to find a template within the main design of the siteaccess, the system will look for it in the additional designs and the standard design.

Commonly used system templates

The following table shows some of the most commonly used system templates.

Request	URL	Module	View	Template
Search interface	/content/search	content	search	/templates/content/search.tpl
Shopping basket	/shop/basket	shop	basket	/templates/shop/basket.tpl
Login page	/user/login	user	login	/templates/user/login.tpl
User registration	/user/register	user	register	/templates/user/register.tpl

3.2 The pagelayout

The page layout is the main template. Among other things, it dictates the overall look of a site. The file name of the page layout template must be "pagelayout.tpl". It has to be placed inside the "templates" directory of a design. If eZ Publish is unable to find a page layout within the current design (specified by the siteaccess), it will attempt to use the page layout template that is provided by one of the fallback designs (page 159). The following illustration shows the location of the page layout template located in a design called "example".

(see figure 3.5)



Figure 3.5: The location of the pagelayout (main) template.

The page layout contains the HTML, HEAD and BODY tags (the other HTML framework). In addition, it dictates the overall look of a site. Among other things, it is used to describe the visual structure (main layout, logo, main menu, footer, etc.) that will be presented for every page request. The following example shows what is considered to be the most basic page layout:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://
www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">

<head>

<style type="text/css">
    @import url({'stylesheets/core.css'|ezdesign});
    @import url({'stylesheets/debug.css'|ezdesign});
</style>

{include uri='design:page_head.tpl'}

</head>

<body>

{$module_result.content}

</body>
</html>
```

The document type

The very first line in the page layout is used to declare the document type of the pages that are generated by eZ Publish. Per HTML and XHTML standards, a DOCTYPE (short for "document type declaration") informs browsers and syntax validation engines about which version of (X)HTML that is used. This information should be included at the very top of in every web page, this is why it is the first part of the page layout.

The DOCTYPE declaration is one of the key components when it comes to proper rendering and compliant web pages. A DOCTYPE that includes a full URL tells the browser to render the page in standards-compliant mode, treating the (X)HTML, CSS, and DOM structures as they should be treated according to the standards. A missing, incomplete or outdated DOCTYPE throws most browsers into something called "Quirks" mode. In this mode, the browser assumes that the document was written using old-fashioned, invalid markup and code per the chaotic industry norms of the late 1990s. In other words, the page will most likely not be rendered according to the standards and it will certainly not validate.

The HTML tag

The HTML tags encapsulate the marked up contents of an actual web page. In addition to the tag itself, the HTML tag in the example above includes a URL to the XHTML specification. XHTML is a family of current and future document types and modules that reproduce, subset, and extend HTML 4. The XHTML family document types are XML based, which means that they are designed to work in conjunction with XML-based user agents.

In document processing, it is often useful to identify the natural or formal language in which the content is written. The "lang" and "xml:lang" attributes specify the language of the entire HTML element. The value of the xml:lang attribute takes precedence. The language values should be set to the language that is used throughout the site. The values of the attributes are language identifiers as defined by ISO 3166-1 (and the corresponding ISO 3166-1-alpha-2) standards.

The head tag

The head tag contains information about the document itself. The information contained here doesn't show up on the page displayed in a web browser. Only the contents of the title tag will be made visible (as the title of the browser window). The head tag typically contains information about which CSS files that should be used, a description of the document itself, keywords and so on.

Cascading Style Sheets

The page layout in the example above makes use of two CSS files: "core.css" and "debug.css". The code encapsulated by curly brackets is eZ Publish specific code. What happens here is that the text within the quotes is piped into a template operator called "ezdesign". The operator prepends the text with the path to the current design directory (the one which is specified using

the "SiteDesign" configuration directive). This technique assures that the path to the CSS files are always correct, regardless of the access method (page 147) that is being used. For example, if the name of the current design is "my_design" and it includes a CSS file called "example.css", the following output will be produced:

```
@import url("/design/my_design/stylesheets/example.css");
```

The "core.css" and "debug.css" files are a part of the standard design that comes with eZ Publish. It is not necessary to have these CSS files in the "stylesheets" directory of a custom design. If eZ Publish is unable to find the files within the current/custom design, it will automatically use the ones that are in the standard design. Please refer to the description of the automatic fallback system for a detailed description of the fallback mechanism. Because of the fallback system, the style-part of the page layout presented above will most likely result in the following output:

```
...  
<style type="text/css">  
    @import url("/design/standard/stylesheets/core.css");  
    @import url("/design/standard/stylesheets/debug.css");  
</style>  
...
```

The core stylesheet

The "core.css" file defines a standard set of basic styles (font styles, sizes, margins, etc.) for both general HTML elements and some eZ Publish specific classes. The eZ publish specific classes are used by the standard templates. A site that makes an extensive use of the default templates should always have the "core.css" file included in the page layout. Otherwise, the missing styles may cause the unexpected rendering of various elements.

The standard "core.css" file should never be changed. If there are basic styles in core.css that doesn't fit the visual environment of a site, a modified version of "core.css" may be placed in the custom design that the site uses. However, the recommended solution is to create a completely new CSS file that contains both custom classes and overrides for elements defined in "core.css".

The debug stylesheet

The "debug.css" file contains styles that are used to format the debug output which appears at the bottom of the page when debug output is enabled. The usage of the "debug.css" file is only necessary during the development of the site (typically when debug information is needed) and thus it can be removed or commented out before the site is launched.

Document information

The system is capable of automatically generating information about the page itself (title, meta tags, keywords, etc.). This can be done by the inclusion of the page head (page 184) template

("page_head.tpl"), which is located in the templates directory of the standard design. If eZ Publish is unable to find the requested file in current/custom design, it will automatically fallback and use the file located in the standard design.

The body tag

The body tag defines the document's body, which contains the actual contents of the web page (text, images, etc.) marked up in an orderly fashion. At the minimum, an eZ Publish page layout should contain the result from the requested modules.

Module result

Upon every request, eZ Publish automatically generates an array called "module_result". This array is available only in the page layout template. It contains all the necessary information about which module that was run, which view that was called, the output that was produced and so on. The actual output (for example the contents of a news article) can be included in the page layout by accessing the "content" element of the \$module_result array, the syntax is:

```
{ $module_result.content }
```

When the page layout is rendered, the { \$module_result.content } part will be replaced with the actual output that the requested module produced. Please refer to the "Variables in pagelayout" (page 188) page for an overview of the template variables that can be accessed from within the pag elayout.

Debug information

The last part of a typical eZ Publish pagelayout is an HTML comment that looks like this:

```
<!--DEBUG_REPORT-->
```

If the debug information is turned on, eZ Publish will replace this comment with the actual debug report when the page layout is processed. In other words, the debug report will be included as a part of the generated page and thus it will not cause invalid output by breaking the HTML structure. The debug reports that eZ Publish generates follow the XHTML 1.0 Transitional specification and thus the debug information validates.

3.2.1 The page head

The standard design contains a page head template that can be used to automatically generate important tags that should be included in the head section of every HTML response. The output of the standard head template (/design/standard/template/page_head.tpl) can be broken down into the following group of tags:

- Title tag
- Meta tags
- Link tags

The following HTML dump shows an example of the output from the standard page head template.

```
<title>Current / Parent / Top - Site name</title>

<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<meta http-equiv="Content-language" content="eng-GB" />
<meta name="author" content="eZ Systems" />
<meta name="copyright" content="eZ Systems" />
<meta name="description" content="Content Management System" />
<meta name="keywords" content="cms, publish, e-commerce, content management,
development framework" /
>
<meta name="MSSmartTagsPreventParsing" content="TRUE" />
<meta name="generator" content="eZ publish" />

<link rel="Home" href="/" title="Front page" />
<link rel="Index" href="/" />
<link rel="Top" href="/" title="Current / Parent / Top - Site name" />
<link rel="Search" href="/content/advancedsearch" title="Search Site name" />
<link rel="Shortcut icon" href="/design/standard/images/
favicon.ico" type="image/x-icon" />
<link rel="Copyright" href="/ezinfo/copyright" />
<link rel="Author" href="/ezinfo/about" />
<link rel="Alternate" href="/layout/set/print/content/view/full/
64" media="print" title="Printable version" />
```

Title

The contents of the title tag is based on the location being viewed (the location within either the content node tree or the system itself) and the actual name of the site. The path to the element being viewed is reversed and thus the current element becomes the first component of the title. The components of the path are separated by slashes. When a node is viewed, the path elements

will be the actual names of the objects which are encapsulated by the nodes that make up the path up to and including the target node. When a system function is being accessed (for example the log-in view of the user module: `"/user/login"`), the path will most likely be a reversed version of the module/view combination that was used. The name of the site is appended at the end of the path, separated by a dash. The site name can be configured using the `"SiteName"` directive in a configuration override for `"site.ini"`.

The example above demonstrates the output of the page head template when a node is being viewed. The name of the object encapsulating the node is `"Current"`. The name of the other objects (encapsulated by the parent node and so on) are `"Parent"` and `"Top"`. The name of the site is `"Site name"`.

Meta tags

In addition to the actual information contained on a web page, the HTML of the page may also include information about the document itself. This is achieved by making use of so called meta tags. The information given by meta tags is usually not visible when the web page is viewed. However, the meta tags are used by the web browser and miscellaneous search engines that index and rank the contents of web pages. The standard page head template outputs the most commonly used meta tags. It can be broken down into three types of tags:

- HTTP-EQUIV meta tags
- Generic meta tags
- Additional meta tags

HTTP-EQUIV meta tags

Meta tags with an HTTP-EQUIV attribute are equivalent to HTTP headers. These tags usually control the way a browser interprets the document. Tags using this form should have an equivalent effect when specified as an HTTP header. Some web servers automatically translate the contents of these tags to actual HTTP headers. The HTTP-EQUIV meta tags in the page head make sure that the browser (and also search engines) know which character set and language the document uses. The language and character set values are automatically set by eZ publish based on the language and character set that the site uses.

Generic meta tags

The generic meta tags make it possible to reveal meta information about the document itself. Although the specification of meta tags does not define a set of legal meta data properties, it is a common practice to include generic information such as the name of the author, description of the site, copyright notices, keywords, etc. By making use of the `"MetaDataArray[...]"` directive in a configuration override for `"site.ini"`, the site administrator can set up a custom set of generic meta tags. eZ Publish will loop through and display the name and value of the specified tags. The

example above shows the default meta tags that will be used if no custom meta tag configuration is present.

Additional meta tags

The last meta tags set by the standard page head template prevent the usage of smart tags (page 184) and reveal the name of the software that was used to generate the output.

Link tags

Link tags in the HTML head make it possible to relate the document to other documents. This is done by the way of REL and REV attributes. While REL links are used to establish relationships, REV links are used to establish reverse relationships. Some browsers make use of the link tags in order to produce a navigation bar that can be used to quickly navigate the site. The links tags generated by eZ Publish are specified in the "link.tpl" file within the templates directory of the standard design. The standard page head makes use of the "links.tpl" file. The default output of the standard page head template produces a basic set of links that can be used to navigate to different parts of the site. The following list shows the link tags that the page head generates:

Link	Description
Home	The "Home" link points to the root/start of the site. It will always bring the user back to the front page (for example http://www.example.com).
Index	The "Index" link points to the root/start of the site. It will always bring the user back to the front page (for example http://www.example.com).
Top	The "Top" link points to the root/start of the site. It will always bring the user back to the front page (for example http://www.example.com).
Search	The "Search" link points to the "advanced search" view of the "content" module. It will bring the user to the advanced search interface (http://www.example.com/content/advancedsearch).
Shortcut icon	The "Shortcut icon" defines the location of the favorite/shortcut icon. Most browsers will display this icon in front of URLs in the address field and in the bookmark list. The default shortcut icon is the double square white-orange eZ Systems logo. It can be easily replaced by putting a 16x16 pixel icon file (16 color BMP/Windows Icon Format) called "fav-

Copyright	icon.ico" in the images folder of a site design. The "Copyright" link points to the "copyright" view of the "ezinfo" module. The default copyright page of eZ publish will be displayed (http://www.example.com/ezinfo/copyright).
Author	The "Author" link points to the "about" view of the "ezinfo" module. The default about page of eZ publish will be displayed (http://www.example.com/ezinfo/about).
Alternate	The "Alternate" link points to a alternate/printerfriendly version of the page. The printerfriendly version of a page is achieved by making use of the "set" view of the "layout" module. This technique makes it possible to use an alternative pagelayout which is usually stripped for everything (menus, logos, etc.) except the actual content that is being presented.

Link parameters

The links can be completely turned off by passing "enable_link=false()" when including the page head template:

```
{include uri='design:page_head.tpl' enable_link=false()}
```

The link to the alternate/print layout can be turned off by passing "enable_print=false()" when including the page head template:

```
{include uri='design:page_head.tpl' enable_print=false()}
```

3.2.2 Variables in pagelayout

The page layout template contains miscellaneous variables that can be used to display information about the state of the system and/or to control the output. The following table shows the available variables along with a brief description.

Variable	Type	Description
\$access_type	array	The name of the siteaccess (as "name") and the ID number (as "type") of the access method (page 147) that was used (1=URL, 2=Host, 3=Port).
\$anonymous_user_id	integer	The ID number of the content object that represents the anonymous user account (the default/standard value is 10).
\$current_user	object	The ezuser object of the user who is currently logged in. If no user is logged in, the anonymous user account will be used.
\$ezinfo	array	An array of three strings: "version", "version_alias" and "revision". These strings reveal basic information about the eZ publish release that is being used.
\$module_result	array	Contains information about the result (and the result itself) generated by the module/view that was executed.
\$navigation_part	array	A hash containing the name and the identifier (the keys are "name" and "identifier") of the current navigation part; for example: "Content structure" and "ezcontent-navigationpart". The navigation part is used by the administration interface to determine which part the user interacts with.
\$requested_uri_string	string	Contains the site specific part of the requested URL, for ex-

		ample: "content/view/full/44" (system URL) or "company/about" (virtual URL).
\$site	array	Contains miscellaneous information about the siteaccess that is being used (site name, design resource, meta tags, etc.)
\$ui_component	string	The user interface component which eZ publish uses while the current page is being shown. This variable is used by the administration interface.
\$ui_context	string	The user interface context in which eZ publish is in while the current page is being shown. This variable is used by the administration interface to distinguish between different modes (for example "navigation", "edit", "browse", etc.).
\$uri_string	string	The system version of the requested URL (for example "/content/view/full/13").
\$warning_list	array	An array of warnings related to problems that were discovered when the page was rendered.

\$module_result

The \$module_result array contains the result that was generated by the module/view which was executed. If eZ Publish was instructed to display the contents of a node, the variable will contain additional information about the node that was requested. If eZ Publish was instructed to do something else (practically anything that is not an actual node view), the result will not contain additional information. The following tables show the contents of the \$module_result variable in the different scenarios.

The default \$module_result

Element	Type	Description
content	string	The actual content (result of

		templates) that was generated by the requested view.
path	array	<p>An array of hashes containing information about the path which leads to the page that is currently being viewed. Each hash contains the following keys: "text", "url". The "text" element usually contains the name of the module/view (for example "Collected information"). The "url" element contains the address. The "url" key of the last element in the array is usually set to false.</p> <p>The standard page head (page 184) template uses the path array to build the TITLE component of the HEAD section. In addition, the path array can for example be used to build breadcrumbs (a path with names (as hyperlinks) of pages/views that lead to the current page/view).</p>
is_default_navigation_part	boolean	Returns TRUE if the default navigation part is being used (the one which is set in PHP code). Returns FALSE if the navigation part of the current module/view has been reconfigured by the site administrator. This can be done by making use of the "NavigationPart" directive of the "[ModuleSettings]" section within a configuration override for "module.ini".
navigation_part	string	The identifier of the current navigation part (for example "ezcontentnavigation-part"). This variable is used by the administration interface to determine which part

		the user interacts with.
ui_context	string	The user interface context in which eZ publish is in while the current page is being shown. This variable is used by the administration interface to distinguish between different modes (navigation, edit, browse, etc.)
ui_component	string	The user interface component which eZ publish uses while the current page is being shown. This variable is used by the administration interface.
uri	string	Contains the site specific part of the requested URL, for example: "content/view/full/44" (system URL) or "company/about" (virtual URL).

The \$module_result when a node is being viewed

Element	Type	Description
content	string	The actual content (result of templates) that was generated by the requested view.
view_parameters	array	An array of the parameters that were sent to the view (for example "limit", "offset", etc.).
path	array	An array of hashes containing information about the path of nodes which lead to the node that is currently being viewed. Each hash contains the following components: Key: text Description: The name of the object referenced by the node. Key: url Description: The system

		<p>URL of the node (for example <code>"/content/view/full/44"</code>).</p> <p>Key: <code>url_alias</code> Description: The virtual URL of the node (for example <code>"company/about_us"</code>).</p> <p>Key: <code>node_id</code> Description: The ID number of the node.</p> <p>The node being viewed will have its <code>"url"</code> and <code>"url_alias"</code> components set to false. In addition, the <code>"node_id"</code> will not be available. The path array can for example be used to build breadcrumbs (a path with names (as hyperlinks) of the objects referenced by the nodes that lead to the target/current node).</p>
<code>title_path</code>	array	<p>Almost the same as the <code>"path"</code> array (see above). When a node is being viewed, the standard page head (page 184) template uses the <code>"title_path"</code> array to build the TITLE component of the HEAD section.</p>
<code>section_id</code>	string	<p>The ID number of the section which the object referenced by the node being viewed belongs to.</p>
<code>node_id</code>	string	<p>The ID number of the node that is being viewed.</p>
<code>navigation_part</code>	string	<p>Contains the name identifier of the current navigation part (for example <code>"ezcontentnavigationpart"</code>). This variable is used by the administration interface to determine which part the user interacts with.</p>
<code>content_info</code>	array	<p>Contains miscellaneous information about the node</p>

that is being viewed:

Variable: node_id

Type: string

Description: The ID number of the node.

Variable: parent_node_id

Type: string

Description: The ID number of the parent node.

Variable: object_id

Type: string

Description: The ID number of the object referenced by the node.

Variable: class_id

Type: string

Description: The ID number of the class which the object is an instance of.

Variable: class_identifier

Type: string

Description: The identifier of the class which the object is an instance of (for example "forum_message").

Variable: offset

Type: integer

Description: The offset view parameter.

Variable: viewmode

Type: string

Description: The view mode that was used to display the node (for example "full", "line", etc.).

Variable: node_depth

Type: string

Description: The depth of the node in the content tree.

		<p>Variable: url_alias Type: string Description: The virtual URL of the node (for example "company/about_us").</p> <p>Variable: persistent_variable Type: n/a Description: A variable set in one of the templates used by the view that was executed. Regardless of the caching mechanisms used, this variable will be available in the pagelayout. The type of the persistent variable depends on the value it contains. If the variable is not set, it will simply return a boolean FALSE.</p> <p>Variable: class_group Type: array Description: The ID numbers of the class groups that the class (which the object being viewed is an instance of) belongs to. This variable is connected with a feature that makes it possible to create template overrides based on class groups. By default the "class_group" always returns a boolean FALSE value because the class group override feature is turned off. It can be turned on by setting the "EnableClassGroupOverride" directive in the [ContentOverrideSettings] block of a configuration override for "content.ini" to "true".</p>
cache_ttl	integer	The TTL (Time To Live) value

		of the result that was generated by the module's view (as seconds). A TTL of minus one means that the view cache should never expire. A TTL of zero means that the result should never be cached.
is_default_navigation_part	boolean	Returns TRUE if the default navigation part is being used (the one which is set in PHP code). Returns FALSE if the navigation part of the current module/view has been reconfigured by the site administrator. This can be done by making use of the "NavigationPart" directive of the "[ModuleSettings]" section within a configuration override for "module.ini".
ui_context	string	The user interface context in which eZ publish is in while the current page is being shown. This variable is used by the administration interface to distinguish between different modes (navigation, edit, browse, etc.)
ui_component	string	The user interface component used by eZ publish while the current page is being shown. This variable is used by the administration interface.
uri	string	The site specific part of the requested URL, for example: "content/view/full/44" (system URL) or "company/about" (virtual URL).

3.3 The template language

The eZ Publish template language makes it possible to extract information from the system and to solve common programmatic issues like for example conditional branching, looping, etc. All eZ Publish specific code must be placed inside a set of curly brackets, "{" and "}". A template file is a combination of HTML and eZ Publish template code. Everything that is encapsulated by curly brackets will be interpreted by the template parser when the template is processed. Everything outside the curly brackets will be ignored and thus it will be sent to the browser without any changes.

Curly bracket issues

Since curly brackets are reserved for defining blocks of eZ Publish template code, these characters can not be used directly in a template. For example, JavaScript code can not be inserted directly into a template file because it makes an extensive use of curly brackets. All non template specific code/text that uses curly brackets must be put inside a "literal" section. The contents of a literal section will be ignored by the template parser. The following example demonstrates the usage of the literal tags:

```
...
{literal}
<script language="JavaScript" type="text/javascript">
<!--
    window.onload=function()
    {
        document.getElementById( 'sectionName' ).select();
        document.getElementById( 'sectionName' ).focus();
    }
-->
</script>
{/literal}
...
```

Outputting curly brackets

It is possible to output curly brackets using two template functions called "ldelim" and "rdelim" (short for left delimiter and right delimiter). The following example demonstrates the usage of these functions:

```
...
This is the left curly bracket: {ldelim} <br />
This is the right curly bracket: {rdelim} <br />
...
```

The following output will be produced:

This is the left curly bracket: {
This is the right curly bracket: }

3.3.1 Comments

Just like in almost any programming language, comments can be used to add explanations, descriptions, etc. Template comments are ignored by the parser and will not be displayed in the resulting HTML output.

There is only one way to add template comments, and that is by encapsulating a block of code by a matching pair of the "{*" and "*}" sequence of characters (left curly bracket + asterisk and asterisk + right curly bracket). In other words, a template comment is just like any other template code except that the curly brackets are accompanied by adjacent asterisks. It is possible to comment both single and multiple lines of code. However, nesting of comments is not supported (it is not possible to comment a chunk of code that already is a comment). The following examples demonstrate the use of comments.

Single line comment

```
{* This is a single line comment. *}
```

The example above will not produce any output.

Multi-line comment

```
{* This is a long comment that  
   spans across several lines  
   within the template file. *}
```

The example above will not produce any output.

Nested comments (illegal)

```
{* {* Nested comments are not supported! *}  
This text will be displayed. *}
```

The example above will produce the following output:

```
This text will be displayed.
```

3.3.2 Variable types

The eZ Publish template language supports the following variable types:

- Numbers
- Strings
- Booleans
- Arrays
- Objects

While some variable types can be created on the fly, others need to be created using an operator. Types that may be created directly are numbers and strings. Booleans and arrays must be created using operators, objects may be created using miscellaneous functions and operators. In addition to the types listed above, it is also possible to create and use custom variables. Custom variable types must be represented as objects.

Numbers

Numbers are numerical values. A number can be a positive or a negative integer or a floating point value. The following example demonstrates how different numbers can be used directly within template code:

```
{13}
{1986}
{3.1415}
{102.5}
{-1024}
{-273.16}
```

Strings

A string is an arbitrary sequence of characters (text) that is encapsulated by a matching pair of either single or double quotes, ' or ". If the quotes are omitted, the string will most likely be interpreted as a function name. Strings are usually defined in the following way:

```
{'This is a string.'}
{"This is another string."}
```

The output of the example above would be:


```
This is a string.
This is another a string.
```

Using quotes

It is possible to use quotes inside strings. This can be done by either using a different kind of quote or by making use of the escape character (backslash). The following examples demonstrate the use of quotes inside strings:

```
{'The following text is double quoted: "Rock and roll!"  '}
```

```
{"The following text is single quoted: 'Rock and roll!'  "}
```

```
{'Using both single and double quotes: "Rock\n roll!"  '}
```

```
{'Using both single and double quotes: \'Rock\n roll!\'  '}
```

```
{"Using both single and double quotes: 'Rock'n roll!'  "}
```

```
{"Using both single and double quotes: \"Rock'n roll\"  "}
```

The output of the example above will be:

```
The following text is double quoted: "Rock and roll!"
The following text is single quoted: 'Rock and roll!'
Using both single and double quotes: "Rock'n roll!"
Using both single and double quotes: 'Rock'n roll!'
Using both single and double quotes: 'Rock'n roll!'
Using both single and double quotes: "Rock'n roll!"
```

Because of the way template code is defined (encapsulated in a matching pair of curly brackets), the right curly bracket, "}", must also be prepended by the backslash escape character. The following example demonstrates this.

```
{' { This text is inside curly brackets. \} '}
```

The output of the template code above will be:

```
{This text is inside curly brackets.}
```

Template strings do not support inline expansion of variables (as in Perl and PHP). In other words, it is not possible to mix variables into strings. However, the concat operator can be used to append the contents of some variable to a string; which means that this operator can be used to build strings consisting of other strings and/or miscellaneous variables.

Booleans

Booleans are binary, they are either TRUE (1) or FALSE (0). A boolean must be created using either the "true" or the "false" template operator. Example:

```
{true()}  
{false()}
```

For some operators and functions, it is possible to use integers as booleans. However, these are not "real" booleans. Zero means FALSE; all non-zero values mean TRUE. Some operators are able to treat an array as if it were a boolean value. While an empty array means FALSE, a non-empty array means TRUE.

Arrays

Arrays are containers that are capable of holding a collection of any other variable type including other arrays. An array can be a simple vector or a hash map (associative array). An element of a vector can be accessed using an index number. The number denotes the position of the element inside the array (the first element is zero, the second element is one, and so on). An element of an associative array can be accessed using an identifier. Regular arrays can be created with the "array" operator. Associative arrays can be created with the hash operator. The following examples demonstrate the creation of arrays and hashes.

Example 1: Array of numbers

```
{array( 2, 4, 8, 16 )}
```

This example creates an array containing four numbers. The array will consist of the following elements:

Index	Value of element
0	2
1	4
2	8
3	16

Example 2: Array of strings

```
{array( 'This', 'is', 'a', 'test' )}
```

This example creates an array containing four strings. The array will consist of the following elements:

Index	Value
0	'This'
1	'is'
2	'a'
3	'test'

Example 3: Associative array

```
{hash( 'Red', 16, 'Green', 24, 'Blue', 32 )}
```

This example creates an associative array containing three key-value pairs. The array will consist of the following elements:

Key	Value
Red	16
Green	24
Blue	32

Objects

Template objects are created by PHP code or by special template operators. The system uses objects to represent data structures of different kinds and sizes. For example, objects are used to represent information about content nodes, translations, webshop orders, user accounts, roles, policies and so on. Refer to the "Objects" section of the "Reference" chapter for a complete overview of the objects and their contents.

Object attributes

Objects consist of named attributes where each attribute can be a different type. The attributes may represent any type of data (numbers, strings, arrays, etc.) and even other objects. Since the attributes are named (each one has an identifier associated to it), their contents can be easily accessed using the different identifiers. This is done in the same way as when accessing the values of associative arrays using identifiers.

The following illustration shows the structure (with example values) of an object ("ezdate") that contains information about a date.

(see figure 3.6)

Attribute	Type	Value
timestamp	string	567990000
is_valid	boolean	TRUE
year	string	1988
month	string	01
day	string	01

Figure 3.6: The structure of the "ezdate" object.

The illustration above reveals that the "ezdate" object consists of five attributes ("timestamp", "is_valid", "year", "month" and "day"). All attributes are represented as strings except the "is_valid" attribute, which is a boolean. The values are the actual data that the object contains.

Attribute availability

It is worth noting that while some attributes are pre-fetched/calculated when an object itself is fetched, others are not. This means that accessing the contents of attributes may require additional processing (usually in the form of database queries). The "static" column in the reference documentation for objects indicates whether the different attributes provide pre-fetched values or if they need to be computed upon request. This information should be helpful when it comes to optimizing your templates.

3.3.3 Variable usage

Template variables must be referenced using dollar (\$) notation, for example: `$my_variable`, `$object_array`, etc. An eZ Publish template variable is case sensitive. In other words, `$lollipop` is not the same variable as `$LolliPop`. Template variables can be created by the system (from PHP code) or by the author of the template (from within template code). Regardless of where a variable was created, it can be changed using the "set" function. Some templates have preset variables, for example, the main template (pagelayout) provides access to a collection of variables (page 188).

Creating and destroying variables

All variables used in a template must be declared and defined by the "def" function (short for define) before they can be used. A variable exists until the "undef" function (short for undefine) is used in order to destroy it. A previously declared variable will be automatically destroyed at the end of the template file in which it was created. The following example demonstrates the most basic use of the "def" and "undef" functions.

```
{def $temperature=32}

    {$temperature}

{undef}
```

The output of the example will be "32". After the `{undef}` function is called, the `$temperature` variable will not be available. Both the "def" and the "undef" function can be used with multiple variables at the same time. In addition, the "undef" function can be used without any parameters. When called without parameters the "undef" function automatically destroys all variables that were previously created within the template. The following example demonstrates how the "def" and "undef" functions can be used to create and destroy multiple variables at the same time.

```
{def $weather='warm' $celsius=32 $fahrenheit=90}

The weather is {$weather}: {$celsius} C / {$fahrenheit} F <br />

{undef $celsius $fahrenheit}

The weather is still {$weather}. <br />

{undef}
```

The output of this example will be:

```
The weather is warm: 32 C / 90 F
The weather is still warm.
```

In the example above, the "def" function is used to create three new variables: \$temperature, \$celsius and \$fahrenheit. The "undef" function is used twice. The first time, it is used to destroy the \$celsius and \$fahrenheit variables. The second is time it is called without parameters and thus the remaining variables (in this case only \$temperature) will be destroyed. For more information, please refer to the documentation page of the "def" and "undef" functions.

Changing the contents of variables

The contents/value of a variable can be changed at any time using the "set" function. Please note that this function can be used to change the value of any variable, regardless of if it was created by the system or inside a template. No warning will be given if a system variable is changed. The "set" function can be used to change the value of any variable regardless of the variable's current type and the type of the new value. In other words, this function is capable of changing the type of a variable. The "set" function can not be used to change the value of an element/attribute of an array, hash or an object. In fact, the elements/attributes of arrays, hashes and objects can not be changed from within template code. The following example demonstrates the usage of the "set" function.

```
{def $weather='warm'}  
  
The weather is {$weather}. <br />  
  
{set $weather='cold'}  
  
The weather is {$weather}.  
  
{undef}
```

The output of the example will be:

```
The weather is warm.  
The weather is cold.
```

Just like the "def" and "undef" functions, the "set" function can work with multiple variables at the same time. For more information, please refer to the documentation page of the "set" function.

Accessing array elements

The elements of a simple/vector array can only be accessed using numerical indexes. This method is called "index look-up". The elements of an associative array can be accessed by using the key identifiers. This method is called "identifier lookup". The following example demonstrates the different lookup methods.

Index look-up

Index look-up is carried out by appending a period/dot and an index number to the name of a simple/vector or associative array. Index look-up may also be carried out by appending a matching pair of brackets that encapsulate the desired index value. The following example demonstrates how to access the elements of a simple array using index look-up. Please note the different syntax (dot and brackets).

```
{def $sentence=array( 'Life', 'is', 'very', 'good!' )}  
  
The 1st element is: {$sentence.0} <br />  
The 2nd element is: {$sentence.1} <br />  
The 3rd element is: {$sentence[2]} <br />  
The 4th element is: {$sentence[3]} <br />  
  
{undef}
```

The code above will output the following:

```
The 1st element is: Life  
The 2nd element is: is  
The 3rd element is: very  
The 4th element is: good!
```

Identifier look-up

Identifier look-up can be carried out by appending a period/dot and an identifier name to the name of an associative array. Identifier look-up may also be carried out by appending a matching pair of brackets that encapsulate the desired index value. The following example demonstrates how to access the elements of an associative array using the identifier look-up method. Notice the different syntax (use of dot and brackets).

```
{def $sentence=hash( 'first', 'Life',  
                   'second', 'is',  
                   'third', 'very',  
                   'fourth', 'good!' )}  
  
The 1st element is: {$sentence.first} <br />  
The 2nd element is: {$sentence.second} <br />  
The 3rd element is: {$sentence[third]} <br />  
The 4th element is: {$sentence[fourth]} <br />  
  
{undef}
```

The following output will be produced:

```
The 1st element is: Life
The 2nd element is: is
The 3rd element is: very
The 4th element is: good!
```

Accessing object attributes

The attributes of an object can only be accessed using the attributes' identifiers. An identifier is just the name of an attribute (similar to the keys of an associative array). The following example demonstrates how the different attributes of a node object can be accessed from within a template.

```
The ID of the node: {$node.node_id} <br />
The ID of the object encapsulated by the node: {$node.object.id} <br />
The name of the object: {$node.object.name} <br />
First time the object was published: {$node.object.published|l10n( shortdate
)} <br /
>
```

If the `$node` variable contains a node that has ID number 44 and encapsulates object number 13 named "Birthday" published on the first of April in 2003, the following output will be produced:

```
The ID of the node: 44
The ID of the object encapsulated by the node: 13
The name of the object: Birthday
First time the object was published: 01/04/2003
```


3.3.4 Array and object inspection

By using the "attribute" template operator, it is possible to quickly inspect the contents of arrays and template objects. The operator creates an overview of available keys, attribute names and/or methods in an object or an array. By default, only the array keys and object attribute names (also called identifiers) are shown. By passing "show" as the first parameter, the operator will also display the values.

The second parameter can be used to control the number of levels/children that will be explored (the default setting is 2). The following example demonstrates how the operator can be used to inspect the contents of an "ezcontentobjecttreenode" object.

```
{ $node | attribute( show, 1 ) }
```

The following output will be produced:

Attribute	Type	Value
node_id	string	2
parent_node_id	string	1
main_node_id	string	2
contentobject_id	string	1
contentobject_version	string	10
contentobject_is_published	string	1
depth	string	1
sort_field	string	8
sort_order	string	1
priority	string	0
modified_subnode	string	1108118324
path_string	string	'/1/2/'
path_identification_string	string	"
is_hidden	string	0
is_invisible	string	0
name	string	'eZ publish'
data_map	array	Array(6)
object	object[ezcontentobject]	Object
subtree	array	Array(114)
children	array	Array(44)
children_count	string	44
contentobject_version_object	object[ezcontentobjectversion]	Object
sort_array	array	Array(1)
can_read	boolean	true
can_create	boolean	false
can_edit	boolean	false
can_hide	boolean	false
can_remove	boolean	false
can_move	boolean	false
creator	object[ezcontentobject]	Object
path	array	Array(0)
path_array	array	Array(2)
parent	object[ezcontentobjecttreenode]	Object
url	string	"
url_alias	string	"
class_identifier	string	'folder'
class_name	string	'Folder'
hidden_invisible_string	string	'-/-'
hidden_status_string	string	'Visible'

As the output shows, there is a lot of information that can be extracted from a node object. In addition to strings and numbers the object also consists of other objects. For example, the creator of the node is a "ezcontentobject" object. The creator object can be further inspected by doing

the following:

```
{ $node.creator | attribute( show, 1 ) }
```

The following output will be produced:

Attribute	Type	Value
id	string	14
section_id	string	2
owner_id	string	14
contentclass_id	string	4
name	string	'Administrator User'
is_published	string	0
published	string	1033920830
modified	string	1033920830
current_version	string	1
status	string	1
current	object[ezcontentobjectversion]	Object
versions	array	Array(1)
author_array	array	Array(1)
class_name	string	'User'
content_class	object[ezcontentclass]	Object
contentobject_attributes	array	Array(5)
owner	object[ezcontentobject]	Object
related_contentobject_array	array	Array(0)
related_contentobject_count	string	0
reverse_related_contentobject_array	array	Array(0)
reverse_related_contentobject_count	string	0

contentobject_count		
can_read	boolean	false
can_create	boolean	false
can_create_class_list	array	Array(0)
can_edit	boolean	false
can_translate	boolean	false
can_remove	boolean	false
can_move	boolean	false
data_map	array	Array(5)
main_parent_node_id	string	13
assigned_nodes	array	Array(1)
parent_nodes	array	Array(1)
main_node_id	string	15
main_node	object[ezcontentobjecttreenode]	Object
default_language	string	'eng-GB'
content_action_list	boolean	false
class_identifier	string	'user'
class_group_id_list	array	Array(1)
name	string	'Administrator User'
matchingroup_id_list	boolean	false

Again, this object consists of a lot of information. As mentioned above, the "attribute" operator can be used on both objects and arrays. The following example demonstrates how to inspect the "data_map" array (which reveals the object's attributes) of the node's creator object.

```
{ $node.creator.data_map|attribute( show, 1 ) }
```

The following output will be produced:

Attribute	Type	Value
first_name	object[ezcontentobjectattribute]	Object
last_name	object[ezcontentobjectattribute]	Object
user_account	object[ezcontentobjectattribute]	Object
signature	object[ezcontentobjectattribute]	Object
image	object[ezcontentobjectattribute]	Object

3.3.5 Control structures

The eZ Publish template language offers a selection of mechanisms that can be used to solve common programmatic issues like for example condition control, looping, etc. The following list shows an overview of the available mechanisms:

- IF-THEN-ELSE
- SWITCH
- WHILE
- DO...WHILE
- FOR
- FOREACH

IF-THEN-ELSE

The IF construct allows for conditional execution of code fragments. It is one of the most important features of many programming languages. The eZ Publish implementation makes it possible to do conditional branching by the way of the following elements: IF, ELSE and ELSEIF. The ELSE and ELSEIF elements are optional. The following examples demonstrate the use of this construct.

Example 1

```
{if eq( $var, 128 )}
    Hello world <br />
{else}
    No world here, move along. <br />
{/if}
```

Example 2

```
{if eq( $fruit, 'apples' )}
    Apples <br />
{elseif eq( $fruit, 'oranges' )}
    Oranges <br />
{else}
    Bananas <br />
{/if}
```

SWITCH

The SWITCH mechanism is similar to a series of IF statements used on the same expression. This construct is typically useful when the same variable needs to be compared to different values. It

executes a piece of code depending on which value that matched a given criteria. The following example demonstrates basic use of this construct.

```
{switch match=$fruits}

  {case match='apples'}
    Apples <br />
  {/case}

  {case match='oranges'}
    Oranges <br />
  {/case}

  {case}
    Unidentified fruit! <br />
  {/case}

{/switch}
```

If the value of the \$fruits variable is "oranges", the following output will be produced:

```
Oranges
```

WHILE

The WHILE construct is the simplest loop mechanism that the template language offers. It tells eZ Publish to execute the nested statement(s) repeatedly, as long as a given expression evaluates to TRUE. The value of the expression is checked for every loop iteration (at the beginning of the iteration). If the given expression evaluates to FALSE from the very beginning, the nested statement(s) will not be executed. The following example demonstrates basic use of this construct.

```
{while ne( $counter, 8 )}

  Print this line eight times ({$counter}) <br />
  {set $counter=inc( $counter )}

{/while}
```

If the initial value of \$counter is zero, the following output will be produced:

```
Print this line eight times (0)
Print this line eight times (1)
Print this line eight times (2)
Print this line eight times (3)
Print this line eight times (4)
```

```
Print this line eight times (5)
Print this line eight times (6)
Print this line eight times (7)
```

DO...WHILE

A DO...WHILE loop is very similar to WHILE loops, except that the expression is checked at the end of each iteration instead of in the beginning. The main difference is that this construct will always execute the first iteration (regardless of how the test expression evaluates). The following example demonstrates basic use of this construct.

```
{do}

    Keep printing this line ({$counter}) <br />
    {set $counter=inc( $counter )}

{/do while ne( $counter, 8 )}
```

If the initial value of \$counter is 0, the following output will be produced:

```
Keep printing this line (0)
Keep printing this line (1)
Keep printing this line (2)
Keep printing this line (3)
Keep printing this line (4)
Keep printing this line (5)
Keep printing this line (6)
Keep printing this line (7)
Keep printing this line (8)
```

FOR

Generic looping may be achieved using FOR loops. This construct supports looping over numerical ranges in both directions. In addition it also supports breaking, continual and skipping. The following example demonstrates basic use of this construct.

```
{for 0 to 7 as $counter}

    Value of counter: {$counter} <br />

{/for}
```

The following output will be produced:

```
Value of counter: 0  
Value of counter: 1  
Value of counter: 2  
Value of counter: 3  
Value of counter: 4  
Value of counter: 5  
Value of counter: 6  
Value of counter: 7
```

FOREACH

The FOREACH construct can be used to iterate over arrays in different ways. The loop can be tweaked using miscellaneous techniques. The following example demonstrates basic use of this construct.

```
{foreach $objects as $object}  
    {$object.name} <br />  
{/foreach}
```

The example above will print out the names of the objects that are stored in the \$objects array. If this array stores 4 objects with the following names: "Emmett Brown", "Marty McFly", "Lorraine Baines" and "Biff Tannen", the following output will be produced:

```
Emmett Brown  
Marty McFly  
Lorraine Baines  
Biff Tannen
```


3.3.6 Functions and operators

The eZ publish template language offers a collection of various functions and operators that can be used to carry out different tasks. In addition, it is possible to extend the system by creating custom operators for special needs. Custom operators have to be programmed in PHP.

Template functions

A function takes a set of named parameters, carries out a specific task and returns a result. It can be called anywhere in a template using the following syntax:

```
{function_name parameter1=value1 parameter2=value2 ...}
```

A function may take none, one or several parameters. The parameters must be specified after the function name, separated by spaces. Since each parameter is specified using the parameter's name, the parameters can be provided in any order. Each parameter must be assigned a value using the equal sign. The following illustration shows the typical usage of a commonly used function.

(see figure 3.7)

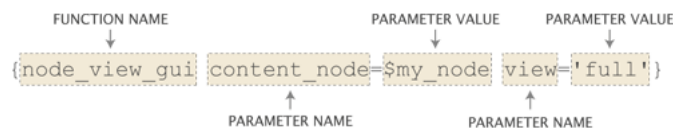


Figure 3.7: Typical components of a function call.

The example above calls the "node_view_gui" function. This function displays a node by including the template that is associated with the view mode. The node is specified using the "content_node" parameter. The desired view mode is specified using the "view" parameter.

Template operators

An operator takes unnamed parameters, carries out a specific task and returns a result. In addition, an operator is capable of handling a parameter which is passed to it using a pipe. It can be called anywhere in a template using the following syntax:

```
{$input_parameter|operator_name( parameter1, parameter2 ... )}
```

Because the operator only takes unnamed parameters, the parameters must be specified in the order dictated by the operator's documentation page. In addition, the parameters must be separated by commas. The following illustration shows the typical usage of a commonly used operator.



Figure 3.8: Typical components of a template operator call.

(see figure 3.8)

The example above demonstrates the usage of the "datetime" operator. This operator can be used to convert a UNIX timestamp to a human readable format. The timestamp is provided by the `$yesterday_evening` variable as the input parameter. The first parameter tells the operator that the output should be formatted using a custom schema. The schema is defined by the second parameter (hours : minutes).

Piping

An operator takes input on the left hand side and produces output on the right hand side. A collection of operators can be glued together using pipes. A pipe makes sure that the output from one operator is presented as the input parameter to another operator. The following example demonstrates how pipes and operators can be used to create a string.

```
{concat( 'To ', 'The ' )|prepend( 'Back ' )|append( 'Future' )}
```

The following output will be produced:

```
Back To The Future
```

3.4 Basic template tasks

This section sheds light on some common issues related to template development.

Template inclusion

A template file can be included using the "include" function. Since this function makes it possible to include any file from any location within the eZ Publish directory, it must be told that it should look for the file within the design directory. This can be done by prefixing the path/filename with "design:". The following example demonstrates how the include function can be used to include a template file called "footer.tpl", which is located in the templates directory of a design.

```
{include uri='design:footer.tpl'}
```

If the requested file is not found within the main design of the siteaccess, the system will search for it in the additional designs and the standard design. Please refer to the documentation of the automatic fallback system (page 159) for more information about this feature.

Output washing

Variables that may contain bogus strings should always be washed using the "wash" operator. This operator makes sure that the output does not contain any elements that may mess up the HTML generated by eZ Publish. The following example demonstrates how the wash operator works.

```
{def $bogus_string='hello < world'}  
{ $bogus_string|wash() }
```

The following output will be produced:

```
hello < world
```

E-mail address obfuscation

In addition to securing proper output, the wash operator can also be used to obfuscate E-mail addresses on a web page. An obfuscated E-mail address has a less chance of getting picked up by a robot searching for E-mail addresses to put on a spammer's list. The following example demonstrates how the wash operator can be used with an E-mail address.

```
{def $email_address='allman@example.com'}  
{ $email_address|wash( 'email' ) }
```

The following output will be produced:

```
allman[at]example[dot]com
```

String concatenation

The "concat" operator makes it possible to glue several strings together in order to produce a single string. The following example demonstrates how this operator works.

```
{def $my_string='sausage'}
{concat( 'Liver ', $my_string, ' sandwich' )}
```

The following output will be produced:

```
Liver sausage sandwich
```

Custom view parameters

The URL of a node view request may contain custom parameters. The custom view parameters must be specified at the very end of the URL using a special notation. For each parameter, a name and a value must be specified. The name must be encapsulated by parenthesis. Each element must be separated by slashes. The following example demonstrates how custom parameters can be used (in addition to the view parameters (page 150)) in a system URL that requests a node.

```
http://www.example.com/content/view/full/13/(color)/green/(amount)/34
```

The same parameters can be appended to the virtual URL of the node:

```
http://www.example.com/company/about_us/(color)/green/(amount)/34
```

When custom view parameters are used, the system will create an associative array using the name of the provided parameters as the keys. All parameter values will be treated as strings. The array will be represented by the `$view_parameters` variable in the template. The parameters given in the examples above will produce an associative array with the following contents:

Key	Type	Value
color	string	green
amount	string	34

The following example demonstrates how the custom view parameters can be accessed in the template that is used to display the node.

```
The color is: {$view_parameters.color} <br />
The amount is: {$view_parameters.amount} <br />
```

The following output will be produced:

The color is: green
The amount is: 34

Custom view parameters in "edit.tpl" templates

In eZ Publish versions prior to 3.9, you cannot pass custom view parameters to the "edit" view of the "content" module. From 3.9, it is possible and thus you can use custom view parameters in the "edit.tpl" templates. The following example demonstrates a typical system URL in this case:

```
http://www.example.com/content/edit/13/03/eng-GB/(color)/green/(amount)/34
```

This will instruct eZ Publish to use custom view parameters, specified in the link above, when editing the "eng-GB" translation of the third version of the thirteenth content object in the system.

3.4.1 URL handling

Whenever a link, a non-content specific image, a stylesheet, etc. is to be included, a suitable template operator must be used in order to ensure that the path to the included file is correct. At any time, one of the following operators should be used:

- ezurl
- ezimage
- ezdesign

ezurl

The "ezurl" operator makes sure that a URL works regardless of the location of the eZ Publish folder, the access method (page 147) and the environment that eZ Publish is running in (non virtual host, virtual host (page 26), etc.). It is only the eZ Publish specific part of the URL that needs to be provided. The rest (http://, host, domain, directory, siteaccess, port, etc.) will be generated by the operator. The final output will be a valid address. This approach makes it possible to use generic URLs in template without the risk of having to modify every address when the site is moved and/or when the access method is changed. By default, the "ezurl" operator outputs an address that is already encapsulated by two double quotes. In other words, the output can be fed directly to an hyperlink reference in the HTML code. The following examples demonstrate the usage of this operator.

Link to a module/view (using a system URL)

```
<a href='{'/user/login'|ezurl()}'>Login</a>
```

The example above demonstrates how to create a link to the *login* view of the *user* module. The `"/user/login"` is just an example, another example would be a link to a node: `"/content/view/full/34"`. If eZ Publish is running in a directory called `"ezpublish"` on `www.example.com` using the URL access method (page 147) and the name of the siteaccess is `"my_company"`, the operator will produce the following output:

```
"http://www.example.com/ezpublish/index.php/my_company/user/login"
```

If eZ Publish is running in a virtual host mode (page 26) and uses the host access method (page 147), the following URL will be produced:

```
"http://www.example.com/user/login"
```

Link to a node (using the node's virtual URL)

When a link to a node (using the node's virtual URL, also known as URL alias) is created, the address must be piped through the "ezurl" operator. The reason for this is that the internal URL table only contains the eZ Publish specific part of the URLs. The following example demonstrates how to use the "ezurl" operator to create a valid virtual URL for a node.

```
<a href={$node.url_alias|ezurl()}>Link to a node</a>
```

If the URL alias of the node is "company/about_us" and eZ Publish is running in a virtual host environment using the host access method, the following URL will be produced:

```
"http://www.example.com/company/about_us"
```

For information about how eZ Publish treats URLs, please refer to the "URL translation" (page 153) section of the "Concepts and basics" chapter.

ezimage

The "ezimage" operator works in the same way as the "ezurl" operator (described above), except that it does not include the "index.php" part. This operator must be used every time a non content specific image is included in a template. The image must be placed in the "images" directory of one of the designs that are used by the siteaccess. The operator produces a valid link to the image regardless of the directory, access method and/or the environment that eZ Publish is running in. The following example demonstrates how the "ezimage" operator should be used.

```
<img src={'women.jpg'|ezimage()} alt="This is my image." ... />
```

If eZ Publish is using the host access method and the siteaccess is using a design called "my_design", the operator will produce the following output:

```
"http://www.example.com/design/my_design/images/women.jpg"
```

If the image is placed inside a sub-directory within the "images" directory, the name of the sub-directory must be specified in the template. If the requested file is not found within the main design of the siteaccess, the system will search for it in the additional designs and the standard design. Please refer to the documentation of the automatic fallback system (page 159) for more information about this feature.

ezdesign

The "ezdesign" operator works in the same way as the "ezurl" operator (described above), except that it does not include the "index.php" part. This operator must be used every time a design

element (style sheets, JavaScript, etc.) is included in a template. The operator takes care of producing a valid link for the given design component by providing the root to the design directory which contains the target file. The following example demonstrates the proper way of including a CSS file using this operator.

```
...
<style type="text/css">
  @import url({'stylesheets/my_stuff.css'|ezdesign()});
</style>
...
```

If eZ Publish is using the host access method and the siteaccess is using a design called "my_design", the operator will produce the following output:

```
"http://www.example.com/design/my_design/stylesheets/my_stuff.css"
```

If the requested file is not found within the main design of the siteaccess, the system will search for it in the additional designs and the standard design. Please refer to the documentation of the automatic fallback system (page [159](#)) for more information about this feature.

3.5 Information extraction

Information that is stored by eZ Publish can be extracted using the "fetch" template operator. This operator gives access to the fetch functions that a module provides. It is typically used to extract nodes, objects, etc. using the content module. The fetch operator can only be used with modules that provide support for data fetching. Please refer to the "Fetch functions" section of the reference chapter for a complete overview of the fetch functions. The following model and table shows the usage and the parameters of the fetch operator.

```
fetch( <module>, <function>, <parameters> )
```

Parameter	Description
module	The name of the target module.
function	The name of the fetch function within the target module.
parameters	An associative array containing the function parameters.

A module's fetch functions and parameters are defined in the "function_definition.php" file within the directory of the module.

Fetching a single node

The following example demonstrates how the fetch operator can be used to extract a single node from the database.

```
{def $my_node=fetch( content, node, hash( node_id, 13 ) )}
...
{undef}
```

The example above instructs eZ Publish to fetch a single *node* from the *content* module. Only one parameter is given, which is the ID number of the node that should be fetched. The operator will return an "ezcontentobjecttreenode" object which will be stored in the `$my_node` variable. This variable can then be used to extract information about the node and the object that it encapsulates. For example, it is possible to extract the name, attributes and the time when the object was published. If the node is unavailable / non-existing or the currently logged in user doesn't have read access to it, the operator will return a FALSE boolean value.

Fetching multiple nodes

It is possible to fetch all the nodes that are directly below a specific node. This can be done by using *list* instead of *node* as the second parameter to the "fetch" operator. The following example

demonstrates how the fetch operator can be used to extract all the nodes that are directly below node number 13.

```
{def $my_node=fetch( content, list, hash( parent_node_id, 13 ) )}
...
{undef}
```

The operator will return an array of "ezcontentobjecttreenode" objects. The list fetch function of the content module can take several parameters. These parameters are optional and can be used to fine-tune the fetch for example by filtering out specific nodes. The following table gives an overview of the most commonly used parameters.

Parameter	Description
sort_by	The method and direction that should be used when the nodes are sorted (must be specified as an array).
limit	The number of nodes that should be fetched.
offset	The offset at which the fetch should start.
class_filter_type	The type of filter that should be used, either "include" or "exclude".
class_filter_array	The type of nodes that should be included or excluded by the filter (must be specified as an array).

The following example demonstrates how to fetch an alphabetically sorted array of the ten latest articles that are directly below node number 13.

```
{def $my_node=fetch( content,
                    list,
                    hash( parent_node_id, 13,
                        limit, 10,
                        class_filter_type, include,
                        class_filter_array, array( 'article' ) ) )}
...
{undef}
```

Please refer to the documentation page of the "list" fetch function for a complete overview of the available parameters and examples of usage.

3.5.1 Outputting node and object data

Once an "ezcontentobjecttreenode" object representing a node is available in a template variable, it can be used to output information about the node and the contents of the object that the node encapsulates. The following text demonstrates the extraction of the most common elements.

General information

The name of the object

```
{ $node.name | wash }
```

The name of the object is directly available through the node (in other words it is possible to reach it by `$node.name` instead of `$node.object.name`). The "wash" operator is used for making sure that the output doesn't contain any bogus characters and/or sequences that may mess up the HTML.

The date/time when the object was first published

```
{ $node.object.published | l10n( 'shortdatetime' ) }
```

Since the publishing value is stored as a UNIX time-stamp, it must be properly formatted for output. This can be done by using the "l10n" operator, which makes it possible to format different types of values according to the current locale settings.

The date/time when the object was last modified

```
{ $node.object.modified | l10n( 'shortdatetime' ) }
```

Since the modification value is stored as a UNIX time-stamp, it must be properly formatted for output. This can be done by using the "l10n" operator, which makes it possible to format different types of values according to the current locale settings.

The name of the user who initially created the object

```
{ $node.object.owner.name | wash }
```

The name of the user who last modified the object

```
{ $node.object.current.creator.name | wash() }
```

The name of the class which the object is an instance of

```
{ $node.object.class_name | wash() }
```

Object attributes

The attributes of the object can be reached by the way of the "data_map" method. This method returns an associative array of "ezcontentobjectattribute" objects where each object represents one of the attributes. The keys of the array are the class attribute identifiers. The following example demonstrates how an attribute called "first_name" can be reached using the object's data map.

```
{ $node.object.data_map.first_name }
```

The example above will not produce any valuable output because the requested data needs to be formatted. There are two ways of outputting the contents of attributes:

- Raw output (the ".output" extension)
- Formatted output (the "attribute_view_gui" function)

The main difference between raw and formatted output is that formatted output makes use of a template which in turn outputs the requested data. Raw output simply outputs the data within the same template where the request for output was issued. Output should always be presented through the "attribute_view_gui" function. The raw output method should only be used when/if necessary (for example when checking the value of an attribute using an IF statement).

Raw output

Raw output is exactly what the definition indicates: a raw dump of the contents that are stored by the attribute. The actual syntax depends on the datatype that represents the attribute. In most cases, it is possible to generate the output by appending ".output" to the identifier.

Generic solution

The following example demonstrates how to output the contents of an attribute called "my_attribute".

```
{ $node.object.data_map.my_attribute.content }
```

XML block

The following example demonstrates how to output the contents of an XML block called "my_xml".

```
{$node.object.data_map.my_xml.content.output.output_text}
```

Image

The following example demonstrates how to output an image stored by an attribute called "my_image".

```

```

Formatted output

Each datatype has a set of templates which are used to display the contents in different contexts. There are at least two templates for each datatype: a view template and an edit template. While the view template is used to display information, the edit template is used when the data is being edited. The default templates for the datatypes are located within the standard design: "/design/standard/templates/content/datatype".

The "attribute_view_gui" function makes it possible to display the contents of an attribute by inserting the view template of the datatype that the attribute uses. The following example demonstrates how this function can be used.

```
{attribute_view_gui attribute=$node.object.data_map.name_of_any_attribute}
```

The example above will generate proper output for any attribute (regardless of the datatype).

3.6 The template override system

The template override system makes it possible to use other templates than the default ones (specified in the code for the different views and templates). This mechanism allows the creation of template overrides for virtually any template that is used by eZ Publish (including templates that are requested by the "include" template function using the "design:" prefix). In particular, template overrides are typically useful for displaying different types of nodes in different ways.

An override for a view template is usually activated by a set of conditions. If the conditions match, the alternate template will be used. Different views provide different conditions, some views do not provide any conditions at all. Please refer to the "Template override conditions" section of the "Reference" chapter for a complete overview of the available match rules. The most flexible set of conditions are provided by the "view" view of the "content" module (used when a node is displayed). The following illustration shows how the override mechanism plugs into the rest of the system.

(see figure 3.9)

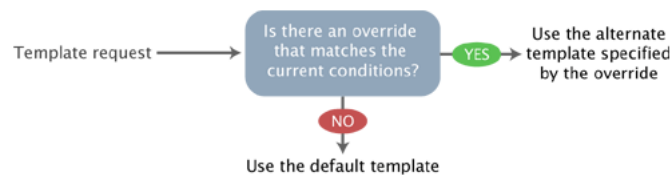


Figure 3.9: *The override system.*

The template overrides must be defined in the "override.ini.append.php" file of a siteaccess. This file consists of override blocks. A block is a named set of rules that tells eZ Publish to use an alternate template in a specific situation. For each block, the following information must be specified:

- A unique name for the override.
- The template that should be overridden.
- The template that should be used instead of the one being overridden.
- The name of the directory in which the override template resides (usually "templates").
- A set of conditions/rules that control when the override should be activated.

Please note that the rules/conditions are optional. If no rules are specified, the override will always be active. The following illustration shows a typical example of a template override with additional explanations.

(see figure 3.10)

The example above defines an override called "special_folders". This override will be used when the system is requested to display a node using the full view mode. The override will only be

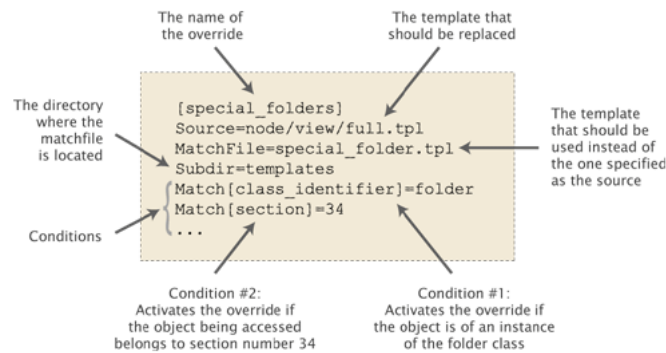


Figure 3.10: *Template override example.*

activated if the object referenced by the node is an instance of the folder class and if it belongs to section number 34. When the override is activated, the system will attempt to use the alternate template ("`/override/templates/special_folder.tpl`", located in the main design). If eZ Publish is unable to find the alternate template, it will look for it in the additional designs and the standard design. Please refer to the documentation page of the automatic fallback system (page [159](#)) for more information about this feature.

Multiple / conflicting overrides

The priorities of the overrides are determined by their positions in the file. If there are several overrides with similar/equal rules, eZ Publish will use the first override that matches and thus the rest of the overrides will be omitted. Because of this, overrides that are for example activated on a node ID or an object ID basis should always be placed first; otherwise they might never be triggered because of the presence of a more generic override with a higher priority.

3.6.1 Template override example

The following example demonstrates how the template override system can be used to display alternate templates in different situations.

Let's say that we have a simple content tree made up of two folders: "News" and "Products". The "News" folder contains news articles and the "Products" folder contains products. The following illustration shows an example of such a tree.

(see figure 3.11)

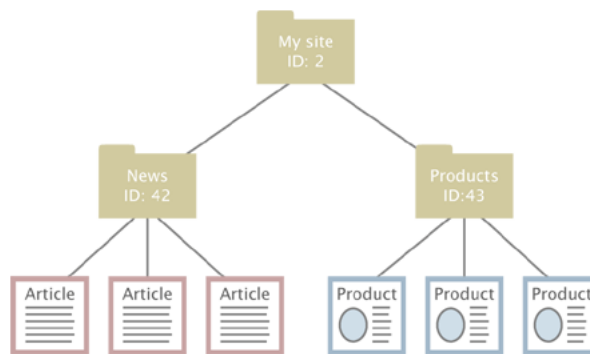


Figure 3.11: Example content node tree.

Without any overrides, eZ publish will most likely display all the nodes using the same template. This would probably be the default full view template located in the standard design. However, what if we wish to display custom/alternate templates for the different nodes? We would perhaps like the system to behave in the following way:

- Display a special "welcome" template when the "My site" node is accessed.
- Display a custom folder template when a folder is accessed.
- Display a custom article template when a news article is accessed.
- Display a custom product template when a product is accessed.

The requests in the list above can be easily achieved by creating a couple of overrides. The welcome page should be solved using an override that is triggered by the identification number of the "My site" node. The rest of the requests can be solved using the class identifier key, which allows an override to be triggered when an object of a certain class is accessed. The following example shows the contents of an "override.ini.append.php" file that makes this possible:

```
# Override for welcome page
[welcome_page]
Source=node/view/full.tpl
MatchFile=my_welcome.tpl
```



```

Subdir=templates
Match[node]=2

# Override for folders
[my_folder]
Source=node/view/full.tpl
MatchFile=my_folder.tpl
Subdir=templates
Match[class_identifier]=folder

# Override for articles
[news_articles]
Source=node/view/full.tpl
MatchFile=my_article.tpl
Subdir=templates
Match[class_identifier]=article

# Override for products
[products]
Source=node/view/full.tpl
MatchFile=my_product.tpl
Subdir=templates
Match[class_identifier]=product

```

The alternate templates must be placed in the "override/templates" sub-directory of the main design used by the siteaccess. The following illustration shows where the templates would be located in a design called "example".

(see figure 3.12)

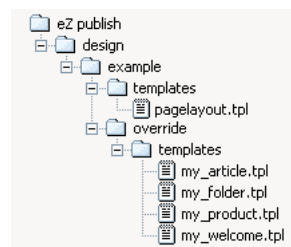


Figure 3.12: Pagelayout + override templates in example design.

When the system is in use, the different overrides would be activated based on the given conditions. The following illustration shows where/when the different alternate templates would be used.

(see figure 3.13)

Every time a node referencing a folder object is viewed, the system will use the "my_folder.tpl"

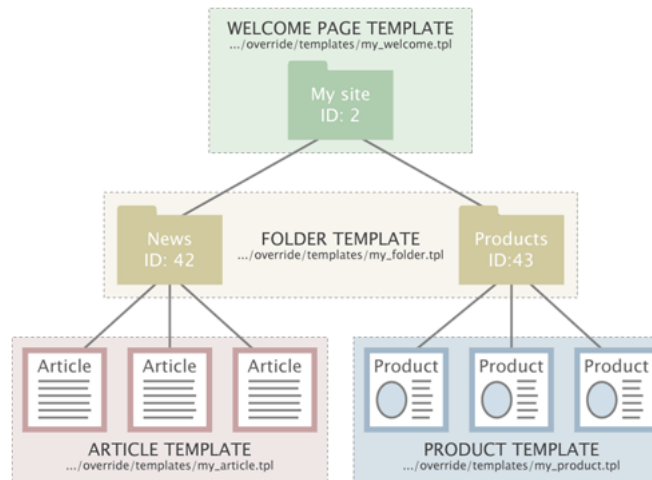


Figure 3.13: *Template override example.*

template. When an article is viewed, the "my_article.tpl" template will be used. When a product is viewed, the "my_product.tpl" template will be used. When node number 2 (the "My site" node) is viewed, the "my_welcome.tpl" will be used.

Chapter 4

Features

This chapter contains information about miscellaneous eZ Publish features along with instructions revealing how to configure and use them.

4.1 Rest API

eZ Publish is introducing a RESTful web service in order to support the development of mobile applications, or even desktop applications. The REST API is built on top of [MVCTools](#) from eZ/ Zeta Components

The REST API for eZ Publish seeks to make the content and the features of an eZ Publish installation available to external clients. The first phase of development centers around providing read-only access to the data within the installation, and for ways to easily extend the REST interface with custom functionality.

Implemented features:

- Basic content retrieval
- REST API extension mechanism
- REST API versioning
- OAuth authentication

This new service is described on these pages:

- Installation (page [236](#))
- REST API configuration settings (page [237](#))
- Getting started with the eZ Publish REST API (page [240](#))
- Authentication (page [241](#))
- Resources (page [250](#))
- Extension functionality (page [266](#))

4.1.1 Installation

The REST interface is bundled with eZ Publish 4.5.0. If you wish to use it, you need to make the the REST endpoint, *index_rest.php* available via rewrite rules. Conversely if you prefer not expose this functionality, you need to make sure that the this file is not exposed in your rewrite rules, either via *.htaccess* files, or virtualhost configuration.

Click this link to find the rewrite rules (page [26](#)).

Now you can continue the tutorial here:

- Step 1: Understanding the REST URI pattern (page [268](#))
- Step 2: Understanding the REST API versioning (page [269](#))
- Step 3: Extending the REST API (page [270](#))
- Conclusion (page [274](#))

4.1.2 REST API configuration settings

Please read on below to find an overview of the INI settings currently existing in the REST API layer:

System settings

```
[System]
PrefixFilterClass=ezpRestDefaultRegexPrefixFilter
ApiPrefix=/api
```

API provider

```
[ApiProvider]
ProviderClass []
```

Debug settings

```
[DebugSettings]
Debug=disabled
```

Output settings

```
[OutputSettings]
RendererClass [xhtml]=ezpContentXHTMLRenderer

[ezpRestContentController_viewContent_OutputSettings]
Template=rest_pagelayout.tpl
```

Cache settings

```
[CacheSettings]
# Global switch to enable/disable REST application cache
ApplicationCache=enabled
```

The ApplicationCache is set to "Enabled" as the default value if no specific value has been defined for your controller/action. If the Application cache is set to enabled the result of each service call will be cached. You can refine this with setting specific to your controller/action. The system will look for a [`<controllerClass>_<action>_CacheSettings`] block to check if cache can be used, and if so, which TTL to use. If this block cannot be found, the system will search at the controller level, and so look for a [`<controllerClass>_CacheSettings`] block.

See example block below for more information

```
# Basically this setting allows you to activate the cache to your controllers/  
actions individually  
ApplicationCacheDefault=enabled  
  
# Set default TTL to 10min, in seconds  
DefaultCacheTTL=600  
  
# Example for action "viewContent", in "ezpRestContentController" controller  
class  
#[ezpRestContentController_viewContent_CacheSettings]  
#ApplicationCache=enabled  
#CacheTTL=3600  
  
# Below an example for every action contained in "ezpRestContentController"  
controller class  
#[ezpRestContentController_CacheSettings]  
#ApplicationCache=enabled  
#CacheTTL=1200  
  
# Switch to enable/disable Routes cache with APC  
RouteApcCache=enabled  
# TTL for Route APC cache, in seconds  
RouteApcCacheTTL=3600
```

Authentication

```
[Authentication]  
RequireAuthentication=enabled  
AuthenticationStyle=ezpRestOAuthAuthenticationStyle  
#AuthenticationStyle=ezpRestBasicAuthStyle  
RequireHTTPS=disabled
```

Route settings

```
[RouteSettings]  
RouteSettingImpl=ezpRestIniRouteFilter  
# Pattern for the skip filter  
# SkipFilter[]=controller_action;version  
# If the version component is not listed version "1" is assumed.  
SkipFilter[]  
SkipFilter[]=ezpRestErrorController_show  
SkipFilter[]=ezpRestAuthController_basicAuth  
SkipFilter[]=ezpRestAuthController_oauthRequired  
SkipFilter[]=ezpRestOAuthTokenController_handleRequest
```

```
# Skip (auth) filter for every action in 'myController' which is of API
version 2
SkipFilter[]=myController_*;2
```


4.1.3 Getting started with the eZ Publish REST API

URI template

As with any RESTful API, it is accessed by doing HTTP requests against URIs. The pattern of the of the API is as follows:

```
GET|POST|PUT|DELETE <HOST:PORT>/<prefix>/<provider>/<version>/<call>/<params>/
```

4.1.4 Authentication

eZ Systems has decided to use OAUTH 2.0 as the authentication service for the REST API of eZ Publish. This is configurable and other authentication mechanisms may be used.

Note that for existing eZ Publish users you can use the HTTP Basic AUTH authentication process. It is also possible to authenticate with **BasicAuth** => **[Authentication].AuthenticationStyle=ezpRestBasicAuthStyle**

This chapter serves as a short recipe on how to deal with OAUTH 2.0. What to do in the development phase of your REST API application, how to register the developed REST API application, and then how the end-user relates to OAUTH 2.0 to get access to and communicate with the new REST API application. You will find the information in the separate, linked chapters below:

The development phase: In the development phase of the REST API application you will not need to activate the OAUTH 2.0 service at all.

Deployment phase: Now eZ Publish administrator should register the REST API application for the OAUTH 2.0 service in the eZ Publish administration interface. You do this to provide yourself and your application end users a secure environment to access and use the application. Click the link "oAuth admin - register the new application" below for more information.

The activation and usage phase: This is the phase where the end users want to access, communicate and use your REST API application. Find more information by clicking the link "The authentication process below.

Maintenance phase: The eZ Publish administrator should read the chapters on how to override the OAUTH log-in page layout template and how to configure the token expiry time for how to further configure the OAUTH 2.0 module.

- oAuth admin - register the new application (page [243](#))
- The authentication process (page [244](#))
- Overriding the OAUTH log-in page layout template (page [248](#))
- Configuring the token expiry time (page [249](#))

OAuth 2.0

- OAuth administration - register the new application
- The authentication process
- Overriding the OAuth log-in page layout template
- Configuring the token expiry time

OAuth administration - register the new application

The OAuth 2.0 client administration is done in the "Administration interface" - under the "Setup" tab. In the left menu you will find the "oAUTH admin". In the screen below you register the new application that you wish end users to access. This serves as the endpoint for authentication and identification in the OAuth authorization process.

(see figure 4.1)

The screenshot displays the eZ Publish administration interface. At the top, there is a navigation bar with tabs for 'Dashboard', 'Content structure', 'Media library', 'User accounts', 'Setup', and 'Service portal'. The 'Setup' tab is active. Below the navigation bar, a breadcrumb trail reads 'You are here: oAuth admin / Edit REST application'. On the left, a sidebar menu titled 'Setup' lists various administrative options, with 'oAuth admin' selected. The main content area shows the 'Edit application <New REST application>' form. The form includes a 'Name' field with the value 'New REST application', a 'Description' field, and an 'Endpoint URI' field. Below the fields are 'OK' and 'Cancel' buttons. At the bottom of the page, a footer contains the text: 'eZ Publish Copyright © 1999-2011 eZ Systems AS and others. For more information see [ezinfo/about](#).'

Figure 4.1:

You (an eZ Publish administrator) enter the name of the new REST application, a description of it and the Endpoint URI of the new REST API application.

The Endpoint URI must contain the complete url of the application since this is an important part of the identification of the application in the automated authorization process. The Endpoint URI is also essential in redirecting from the eZ Publish OAuth module to internal HTTP requests or an alternative application scheme.

The authentication process

The authentication process seen from a user's viewpoint

OAuth 2.0 provides a method for clients to access a protected resource/eZ Publish content on behalf of an eZ Publish user. Before an end user can access a protected resource/eZ Publish content, he/she must first obtain authorization from the eZ Publish OAuth 2.0 module, then exchange the access grant for an access token (representing the grant's scope, duration, and other attributes) and a refresh token that is used when the access token expires. The client accesses the protected resource the first time by presenting the access token to the resource server.

The use cases below have a starting-point with a user who is:

- **Use case 1:** Starting up to get authorization to the client application you have developed
- **Use case 2:** Authorized with an access token and needs to exchange it with a refresh token
- **Use case 3:** Authorized with both an access token and a refresh token and is good to go!

Getting authorization to your client application

The flow illustrated in the figure below includes what goes on when a user (a registered eZ Publish user) wishes to access your newly developed REST API application:

(see figure 4.2)

- The end user requests access to your client application.
- The client application requests user authorization via the eZ Publish OAuth 2.0 module that acts as the authentication filter.
- A consistency check is carried out towards the eZ Publish user accounts.
- Then the eZ Publish back-end displays an authentication form to the user.
- The user logs in and authorizes the application.
- An HTTP 302 message is sent to the client application with a "code" and "expires in" as GET parameters
- The client application requests an access token via an HTTP POST to the eZ Publish back-end
- The eZ Publish back-end now returns a temporary access token (1 hour), a refresh token and an expiry limit in JSON format.
- The client application sends a REST request with an OAuth header to the eZ Publish OAuth 2.0 module, which in turn sends a request to the eZ Publish server to check that the access token is valid
- The eZ Publish back-end returns the requested REST service data to the client application which in turn gives the temporary access token to the user.

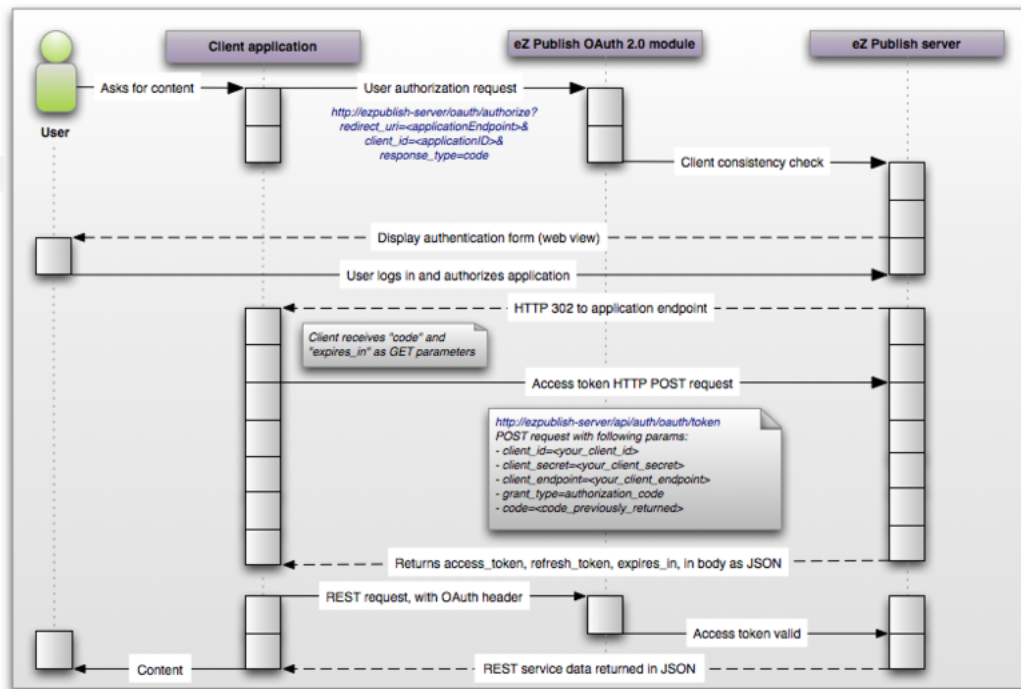


Figure 4.2:

Refreshing the expired access token with a refresh token

The access token is for accessing the back-end, and the refresh token is for refreshing the access token when it is expired. You request a new access token with your refresh token. Then eZ Publish OAuth 2.0 module will return a new access token, with a new refresh token. Follow the flow in the diagram below:

(see figure 4.3)

- The end user requests the client application to refresh his temporary access token with a new access token and a refresh token.
- The client application sends a REST request with the access token in the header to eZ Publish OAuth 2.0 module, which in turn returns an "expired token" with a `www-Authenticate` header in an HTTP 401 format to the client application.
- The client application requests a refresh token via an HTTP POST to the eZ Publish back-end
- The eZ Publish back-end now returns a refresh token and an expiry limit in JSON format.
- The client application sends a REST request with an OAUTH header to the eZ Publish OAuth 2.0 module, which in turn sends a request to the eZ Publish server to check that the access token is valid

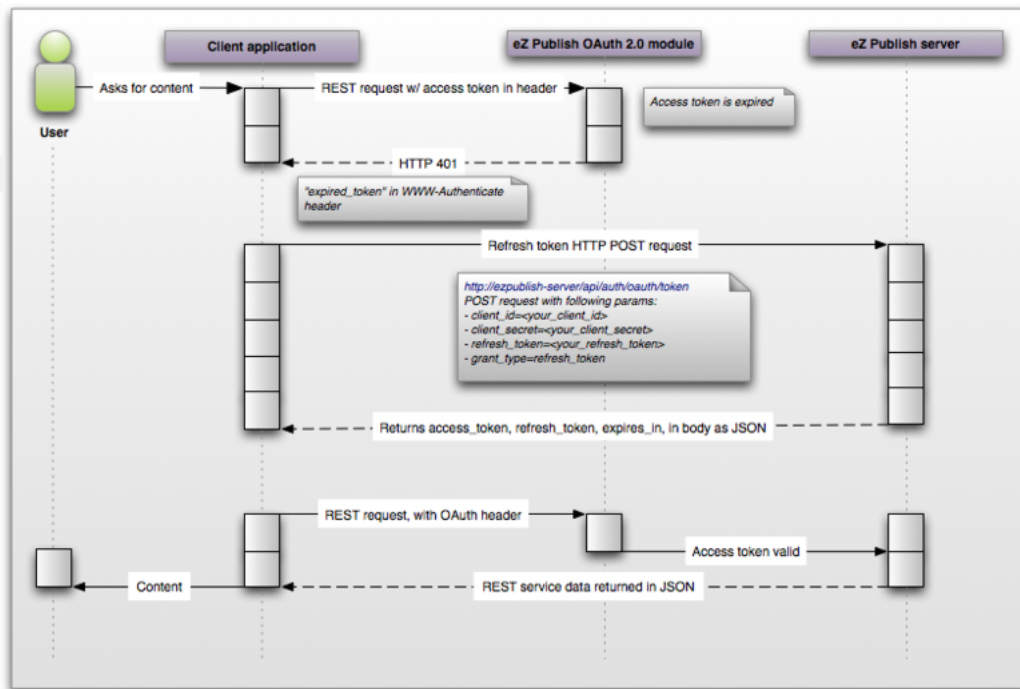


Figure 4.3:

- The eZ Publish back-end returns the requested REST service data to the client application which in turn gives the refresh token to the user.

Authorization is ok and the user is ready to go

Follow the flow in the diagram below:

(see figure 4.4)

- The end user wishes to access the new REST API application
- The client application sends a REST request with the refresh token in the header to eZ Publish OAUTH 2.0 module, which in turn sends a request to the eZ Publish server to check that the access token is valid.
- The eZ Publish back-end returns the requested REST service data to the client application which in turn gives access to the client application to the user.

Note that all use cases above are either fully or partially automated, so in each case the whole process only takes a second or two.

Share information

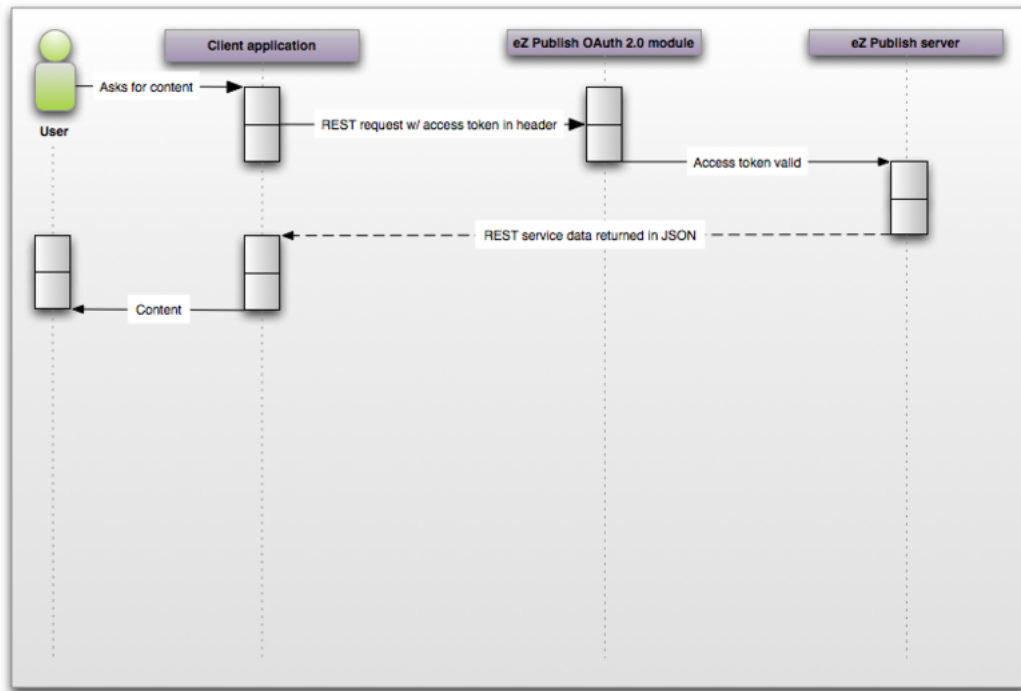


Figure 4.4:

Overriding the OAUTH log-in page layout template

When you first authorize a client to use REST with OAuth, a user needs to be identified. In order to support this a log-in page is displayed. The client enters its credentials and is then redirected back to his application.

The page layout of this log-in page can be overridden like any regular template. The name of this template is "oauthloginpagelayout.tpl".

Configuring the token expiry time

When you first identify your client (let us say a mobile app) on OAUTH, you're given an authorization code. This code is for a one-use-only purpose. This code has then to be traded against an access token which will be used.

As a default this token is set to 1 hour (3600sec) before expiry. The expiry time is configurable in "rest.ini".

4.1.5 Resources

The initial set of resources exposed by the eZ Publish REST API is listed in the following sections. The output type of the REST calls is JSON-data.

Rendered HTML-output of nodes/objects is a part of the JSON-response. This is to serve a practical need when implementing clients. In many cases the best (or only) tool for the job, is to use a web-view to display data, this resource would cater for this need. This functionality will likely exploit new representation features in NEWAPI, in order to provide a unified and flexible rendering mechanism.

```
Accept: application/xhtml+xml / text/html  
GET /api/ezp/content/node/55
```

```
GET /api/ezp/content/node/55.html
```

The fields to return should be configurable for each request. As it is almost impossible to predict what fields the web-service consumer needs to be returned, we organize them in **Response Groups**. They actually control the kind of information returned by the request. This concept is greatly inspired from [Amazon Product Advertising API](#).

Each operation provides one or more default response group(s) that can be completed by one or several more response group(s) available for the operation.

```
CONTENT-URI?ResponseGroups=OneResponseGroup,AnotherOne
```

Services

- List of services

Share your information

List of services

In the selections to the left you will find all the services we provide for 1.0 of the eZ Publish REST API, with their parameters and response groups.

Content

The content resource exposed your run-of-the-mill representations of the traditional content objects and their data maps inside eZ Publish.

Content can exist in various languages. To fetch a specific translation the following query parameters can be added.

```
CONTENT-URI?translation=xxx-YY
```

Response groups

The fields to return should be configurable for each request. As it is almost impossible to predict what fields the web service consumer needs to be returned, we organize them in `Response Groups`. They actually control the kind of information returned by the request. This concept is greatly inspired from [Amazon Product Advertising API](#).

Each operation provides one or more default response group(s) that can be completed by one or several more response group(s) available for the operation:

```
CONTENT-URI?ResponseGroups=OneResponseGroup,AnotherOne
```

For every request, a `requestedResponseGroups` field will be added to each response, providing every response groups that has been requested, including non-supported ones.

Content/object service

HTTP request using an object ID

```
GET /[api]/ezp/v1/content/object/<objectID> HTTP/1.1
Host: api.example.no
```

Parameters

ID: numeric. Content object ID

Response Groups

Metadata (default)

Metadata for the requested object (type, date, etc):

- classIdentifier
- objectName
- datePublished
- dateModified
- objectRemoteID
- objectID
- Links to the fields resources for the node
 - links/<fieldname>

Locations

Available locations for content object with:

- fullURL
- nodeID
- isMainLocation

Fields

Value for all the fields, indexed by attribute identifier

- type (data_type_string)
- identifier (attribute identifier)
- value (textual field representation)
- id (content attribute ID)
- classattribute_id

Content/node service

HTTP request using a node ID

```
GET /[api]/ezp/v1/content/node/<nodeID> HTTP/1.1
Host: api.example.no
```

Parameters

ID: numeric. NodeID

Response Groups

Metadata (default)

Metadata for the requested object (type, date, etc):

- classIdentifier
- objectName
- datePublished
- dateModified
- objectRemoteID
- nodeRemoteID
- objectID
- fullURL
- Links to the fields resources for the node
 - links/<fieldname>

Fields

Value for all the fields, indexed by attribute identifier

- type (data_type-string)
- identifier (attribute identifier)
- value (textual field representation)
- id (content attribute id)
- classattribute_id

Content/fields service

HTTP request for object fields

```
GET /[api]/ezp/v1/content/object/<objectId>/fields HTTP/1.1
Host: api.example.no
```

or

```
GET /[api]/ezp/v1/content/node/<nodeId>/fields HTTP/1.1
Host: api.example.no
```

Parameters

ID: numeric. Either node or object ID, depending on the resource used

Response Groups

FieldValues (default) Value for all the fields of the given object/node

- fields/<fieldidentifier>
 - type (data_type_string)
 - identifier (attribute identifier)
 - value (string representation of data)
 - id (content attribute ID)
 - classattribute_id

Metadata: Metadata for the requested object/node (type, date, etc):

- classIdentifier
- objectName
- datePublished
- dateModified
- objectRemoteID
- nodeRemoteId (if node requested)
- nodeID (if node requested)
- fullURL (if node requested)
- objectid

Content/field service (specific field)

HTTP request for specific fields

```
GET /[api]/ezp/v1/content/object/<objectId>/field/<fieldIdentifier> HTTP/1.1  
Host: api.example.no
```

or

```
GET /[api]/ezp/v1/content/node/<nodeId>/field/<fieldIdentifier> HTTP/1.1  
Host: api.example.no
```

Parameters

ID: Numeric. Either node or object ID, depending on the resource used

FieldIdentifier: An eZ Publish content field reference. This is the class attribute identifier

Response Groups

The output is similar to that of the full fields call, except that it only references the requested field.

Content/list service

HTTP request for node children

```
GET /[api]/ezp/v1/content/node/<nodeId>/list(/offset/<offset>/limit/<limit>/
sort/<sortKey>/<sortType>) HTTP/1.1
Host: api.example.no
```

Parameters

- **nodeID:** Numeric. Node ID
- **offset:** Numeric (optional). Starting offset. Default is **0**.
- **limit:** Numeric (optional). Max number of child nodes that is returned. Default is **10**.
- **sortKey:** String (optional). Sort attribute. Can be every non-attribute sort key supported by content/list fetch function
- **sortType:** String (optional). Can be **asc** (default) or **desc**.

Note for optional parameters:

- offset and limit are optional, but if both are wanted, they must follow the order:
offset/<offset>/limit/<limit>
- sortKey and sortType are optional, but if both are wanted, they must follow the order:
sort/<sortKey>/<sortType>
- if every parameter is wanted they must follow the order:
offset/<offset>/limit<limit>/sort/<sortKey>/<sortType>

Response Groups

Metadata (default)

- childrenCount (total children count, not the current result set count)
- parentNodeId (reproduction of nodeID parameter)

Metadata for the children objects (type, date, etc):

- classIdentifier
- objectName

- datePublished
- dateModified
- objectRemoteID
- nodeRemoteID
- objectID
- fullURL
- link (Link to content/node service for this node)

Fields

Value for all children fields, like in content/fields service.

Content/childrenCount service

HTTP request for content children count

```
GET /[api]/ezp/v1/content/object/<objectID>[api]/ezp/v1/content/node/<nodeId>/
childrenCount HTTP/1.1
Host: api.example.no
```

Parameters

nodeID: Numeric. Parent Node ID

Response Groups

Metadata (default)

- childrenCount (total children count)
- parentNodeId (reproduction of nodeId paramter)


```
<?php /*
    [OutputSettings]
    RendererClass [ezxcr]=ezpContentCustomRenderer

    */ ?>
```

Where the "RendererClass" key is a type of format or renderer name and value is a PHP class name which implements "ezpRestContentRendererInterface".

3. Once you have registered your custom renderer class, you need to create a new PHP class which has to implement the "ezpRestContentRendererInterface". Create an empty `classes` folder under the `ezxcustomrenderer` extension. Next, under `classes` folder create `custom_renderer.php` file with the following code:

```
<?php

class ezpContentCustomRenderer extends ezpRestContentRendererInterface
{
    /**
     * Creates an instance of a ezpContentCustomRenderer for given content
     *
     * @param ezpContent $content
     */
    public function __construct( ezpContent $content, ezpRestMvcController
    $controller )
    {
        $this->content = $content;
        $this->controller = $controller;
    }

    /**
     * Returns string with rendered content
     *
     * @return string
     */
    public function render()
    {
        return 'Custom Renderer String'; //
        Return string with your representation of $this->content
    }
}

?>
```

4. Enable the `ezxcustomrenderer` by adding the following to the `ActiveExtensions` list in the `settings/override/site.ini.append.php`.

```
[ExtensionSettings]
[]
ActiveExtensions []=oauth
ActiveExtensions []=rest
ActiveExtensions []=ezprestapiprovider
ActiveExtensions []=ezxcustomrenderer
```

5. Update the autoload array with a following command from the eZ Publish root folder:

```
php bin/php/ezpgenerateautoloads.php -e -p
```

6. If you have followed all steps carefully then you should be ready to test your new output renderer provider by calling `www.example.com/api/ezp/v1/content/node/2?OutputFormat=ezxcr`. A simple JSON response should look like:

```
{
  "metadata": {
    "objectName": "Home",
    "classIdentifier": "frontpage",
    "datePublished": 1193906012
  },
  "links": {
    "name": "http://www.example.com/api/ezp/content/node/2/field/name?OutputFormat=ezxcr",
    "billboard": "http://www.example.com/api/ezp/content/node/2/field/billboard?OutputFormat=ezxcr",
    "left_column": "http://www.example.com/api/ezp/content/node/2/field/left_column?OutputFormat=ezxcr",
    "center_column": "http://www.example.com/api/ezp/content/node/2/field/center_column?OutputFormat=ezxcr",
    "right_column": "http://www.example.com/api/ezp/content/node/2/field/right_column?OutputFormat=ezxcr",
    "bottom_column": "http://www.example.com/api/ezp/content/node/2/field/bottom_column?OutputFormat=ezxcr",
    "tags": "http://www.example.com/api/ezp/content/node/2/field/tags?OutputFormat=ezxcr",
    "*": "http://www.example.com/api/ezp/content/node/2/fields?OutputFormat=ezxcr"
  },
  "renderedOutput": "Custom Renderer String",
  "requestedResponseGroups": ["Metadata"]
}
```

Custom templates (built-in XHTML renderer)

The `ezpContentXHTMLRenderer` class which is a default renderer for the XHTML output for the `/content/node/<node_id>` service allows you to select main template (pagelayout type) used for the XHTML rendering. By default the main template is called `rest_pagelayout.tpl` and configured in `rest.ini.append.php` in section `[ezpRestControllerViewContentOutputSettings].Template`

Pagelayout

Pagelayout is a main template used for rendering XHTML output for `/content/node/<node_id>` service. Because system is expecting `rest_pagelayout.tpl` template to be present, you need to place it under the design folder used by REST layer e.g `design/<design_name>/templates`. Consider following example for the `rest_pagelayout.tpl`:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://
www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<title>REST Demo</title>
</head>

<body>
{$module_result.content}
</body>
</html>
```

Node templates

Output for node content and it's related templates is generated for view called `rest`. Consider following example for article template:

```
<h1>
    {$node.name|wash()}
</h1>

<div class="intro">
    {attribute_view_gui attribute=$node.data_map.intro}
</div>

<div class="body">
    {attribute_view_gui attribute=$node.data_map.body}
</div>
```

Override rules

Since the `node/view/rest` view is used for rendering node content, custom override rules needs to define the `Source` parameter accordingly. The following example demonstrates the template override rule for the article class for REST output:

```
[rest_article]
Source=node/view/rest.tpl
MatchFile=rest/article.tpl
Subdir=templates
Match[class_identifier]=article
```

Based on the template override information the eZ Publish template engine will search for "article.tpl" in the "design/<design_name>/override/templates/rest/" folder and use it for the article class instance objects.

4.1.6 Extension functionality

At the end of this tutorial, you will be able to embed your custom RESTful interface into eZ Publish, to be further consumed by any 3rd party-service :

- mobile application
- external business application
- web service

This lets you expose any service reading content (heading for [full CRUD](#) support eventually), as well as other, fully custom features, with direct access to about any eZ Publish feature, from PHP.

Pre-requisites and target population

This tutorial is for developers who are keen on making their eZ Publish a REST engine for their specific needs. This 2nd developer preview follows a first one [<link>](#), with which we recommend you to quickly get acquainted. Developer preview means that what you will learn in a few moments is necessarily going to evolve, change, maybe not only slightly. This is not a crystalized piece of knowledge, rather a intermediate (yet pretty advanced) deliverable in an iterative process. Let us all be agile !

Important preliminary note: We are highly recommending to setup REST layer in Virtual Host environment for this release. We are working on adding support for other environments as well. Stay tuned!

Read more on how to set-up your eZ Publish in Virtual Host mode ([page 26](#))

Steps

- Step 1: Understanding the REST URI pattern
- Step 2: Understanding REST API versioning
- Step 3: Extending the REST API
- Conclusion

Step 1: Understanding the REST URI pattern

After going through the installation process, please continue with the REST API functionality below:

The resource URIs provided by default by the eZ Publish REST extension support the following convention:

```
/<prefix>/ + <provider>/<version> + /<function>/<params>
```

For the built-in REST API, the default provider name is `ezp`. The version token is build as `v` + integer for example `v1` for REST API version one. The global prefix (first URI part) can be defined in the REST configuration as `settings/rest.ini.[System].ApiPrefix`.

The RegExp based URI filtering is handled by a default `ezpRestDefaultRegexpPrefixFilter` class implementation where `<provider>` and `<version>` data are used for routs filtering. Developers can implement custom filters by implementing the `ezpRestPrefixFilterInterface` in first place and then updating the `rest.ini.[System].PrefixFilterClass` accordingly.

Getting implementation ideas out of the `ezpRestDefaultRegexpPrefixFilter` can be a good start.

Step 2: Understanding REST API versioning

The resource URIs contain a version token which is build as `v + integer`, for example `v1` for REST API version one, `v2` for version two and so on. Based on the version information provided in the resource URI the pre-routing filter extracts version information and matches against defined versioned routes.

The `EzpRestVersionedRoute` is a route wrapper around existing instance of `EzcMvcRoute` object providing multiple versions of it. Find an example of a versioned routes definition below:

URI: `/api/provider/v1/foo`

```
new EzpRestVersionedRoute( new EzcMvcRailsRoute( '/  
foo', 'ezxRestController', 'foo' ), 1 )
```

URI: `/api/provider/v2/foo`

```
new EzpRestVersionedRoute( new EzcMvcRailsRoute( '/  
foo', 'ezxRestController', 'fooBar' ), 2 )
```

For the first URI the

```
ezxRestController::doFoo()
```

method will be executed, which is version one. For the second URI which is a version two,

```
ezxRestController::doFooBar()
```

will be called. Both methods can share implementation logic but differ with the output result for instance.

Step 3: Extending the REST API

As of now it is possible to extend the eZ Publish REST interface with custom functionality. One REST extension can support many different data providers as well as different versions. The following steps present how to create a new eZ Publish REST extension.

- Create an empty folder called `ezxrestapidemo` under the `<eZ Publish>/extension` location.
- Inside `ezxrestapidemo` create a setting with the `rest.ini.append.php` file with following content:

```
<?php /*
[ApiProvider]
ProviderClass [ezx]=ezxRestApiProvider
*/ ?>
```

ProviderClass is an array where the index is a provider token used in the resource URI. For example for URI: `/api/ezx/v1/foo` the registered provider with token `ezx` will be called. You can have many providers registered within one extension.

- Next create a folder called `classes` under the `<eZ Publish>/extension/ezxrestapidemo/` location with the `rest_provider.php` file. Edit the newly created PHP file and add the following code:

```
<?php
 
class ezxRestApiProvider implements ezpRestProviderInterface
{
    /**
     * Returns registered versioned routes for provider
     *
     * @return array Associative array. Key is the route name (beware of name
     collision!). Value is the versioned route.
     */
    public function getRoutes()
    {
        return array( 'foo' => new ezpRestVersionedRoute( new
ezcMvcRailsRoute( '/
foo', 'ezxRestController', 'foo' ), 1 ),
                    'foobar' => new ezpRestVersionedRoute( new
ezcMvcRailsRoute( '/foo', 'ezxRestController', 'fooBar' ), 2 ) );
    }
}
```

```

/**
 * Returns associated with provider view controller
 *
 * @return ezpRestController
 */
public function getViewController()
{
    return new ezxRestApiViewController();
}
}
?>

```

Every REST API provider has to implement the `ezpRestProviderInterface` where `getRoutes()` methods returns registered versioned routes for provider and `getViewController()` returns view controller object which has to implement the `ezpRestControllerInterface`.

- In the next step create the file `view_controller.php` under the `<eZ Publish>/extension/ezxrestapidemo/classes` folder with following code:

```

<?php
class ezxRestApiViewController implements ezpRestControllerInterface
{
    /**
     * Creates a view required by controller's result
     *
     * @param ezcMvcRoutingInformation $routeInfo
     * @param ezcMvcRequest $request
     * @param ezcMvcResult $result
     * @return ezcMvcView
     */
    public function loadView( ezcMvcRoutingInformation $routeInfo,
        ezcMvcRequest $request, ezcMvcResult $result )
    {
        return new ezpRestJsonView( $request, $result );
    }
}
?>

```

Every view controller has to implement the `ezpRestControllerInterface` where the `loadView()` method returns an instance of the `ezcMvcView` object. In this example we re-use the

€ezpRestJsonView€ which is a part of the built-in API, if your provider need a custom view, you can return it here.

- Registered for REST provider URIs are associated with controllers. This example calls the methods foo() and fooBar() on €ezxRestController€. Create a file called €rest_controller.php€ under €<eZ Publish>/extension/ezxrestapidemo/classes€ folder and put the following code inside:

```
<?php
class ezxRestController extends ezcMvcController
{
    public function doFoo()
    {
        $res = new ezcMvcResult();
        $res->variables['message'] = "This is FOO!";
        return $res;
    }

    public function doFooBar()
    {
        $res = new ezcMvcResult();
        $res->variables['message'] = "This is FOOBAR!";
        return $res;
    }
}
?>
```

Notice the controller methods naming convention. All methods have to be prefixed with €do€ word. This is a €ezcMvcController€ requirement.

- Enable €ezxrestapidemo€ by adding the following to €ActiveExtensions€ list in the €settings/override/site.ini.append.php€

```
[ExtensionSettings]
[...]
ActiveExtensions[]=oauth
ActiveExtensions[]=rest
ActiveExtensions[]=ezprestapiprovider
ActiveExtensions[]=ezxrestapidemo
```

Update the autoload array with a following command from the eZ Publish root folder:

Conclusion

You should now have the "ezxrestapidemo" extension ready, exposing your custom REST API to any REST-enabled channel.

This tutorial showed the principle of extending the eZ Publish REST framework. Given the €preview€ nature of this framework, some changes might occur in the future on the way REST extensions are done, stay tuned for all details. However, the foundations are now laid, let us build off of them together.

4.2 Audit trailing

It is possible to automatically generate audit logs based on what the users are doing with the system. This feature can be useful for big sites with many administrators and editors where information about various operations should be logged and stored. For example, auditing makes it possible to find out which user that removed content, from which IP address the request came from and so on.

The system provides a set of built-in audit functions that make it possible to generate audit logs for different types of activities. At minimum, for every operation, the system logs the following information:

- When it happened (time-stamp)
- Where the request came from (IP address)
- Which user that did it (user name and ID number)

Note that most audit functions provide additional information. The following example shows how a record in one of the log files look like after a node has been moved.

```
[ May 23 2007 14:47:58 ] [127.0.0.1] [editor:16]
Node ID: 124
Old parent node ID: 2
New parent node ID: 59
Object ID: 114
Content Name: Folder
Comment: Moved the node to the given node: eZContentObjectTreeNode::move()
```

The following table shows the available built-in audit functions along with when they are triggered, what kind of information that is actually logged and the default log file where the information is stored.

Audit function	Activity	Logged information	Default log file
user-login	Successful login attempts	<ul style="list-style-type: none"> • Timestamp • IP address • User (name:ID) 	login.log
user-failed-login	Failed login attempts	<ul style="list-style-type: none"> • Timestamp • IP address 	failed_login.log

		<ul style="list-style-type: none"> • &nbsp;User (name:ID) 	
content-move	Location change of content	<ul style="list-style-type: none"> • &nbsp;Timestamp • &nbsp;IP address • &nbsp;User (name:ID) • &nbsp;Old parent node ID • &nbsp;New parent node ID • &nbsp;Object ID • &nbsp;Object name • &nbsp;Comment 	content_move.log
content-delete	Removal of content	<ul style="list-style-type: none"> • &nbsp;Timestamp • &nbsp;IP address • &nbsp;User (name:ID) • &nbsp;Node ID • &nbsp;Object ID • &nbsp;Object name • &nbsp;Comment 	content_delete.log
content-hide	Hide/unhide a content node	<ul style="list-style-type: none"> • Node ID • Object ID • Object Name • Timestamp • Comment 	content_hide.log
role-change	Role and policy changes	<ul style="list-style-type: none"> • &nbsp;Timestamp • &nbsp;IP address • &nbsp;User 	role_change.log

		<ul style="list-style-type: none"> (name:ID) • &nbsp;&nbsp;&nbsp;Role ID • &nbsp;&nbsp;&nbsp;Role name • &nbsp;&nbsp;&nbsp;Comment 	
role-assign	Role assignment to users and groups	<ul style="list-style-type: none"> • &nbsp;&nbsp;&nbsp;Timestamp • &nbsp;&nbsp;&nbsp;IP address • &nbsp;&nbsp;&nbsp;User (name:ID) • &nbsp;&nbsp;&nbsp;Role ID • &nbsp;&nbsp;&nbsp;Role name • &nbsp;&nbsp;&nbsp;Object name • &nbsp;&nbsp;&nbsp;Comment 	role_assign.log
section-assign	Section assignments	<ul style="list-style-type: none"> • &nbsp;&nbsp;&nbsp;Timestamp • &nbsp;&nbsp;&nbsp;IP address • &nbsp;&nbsp;&nbsp;User (name:ID) • &nbsp;&nbsp;&nbsp;Section ID • &nbsp;&nbsp;&nbsp;Section name • &nbsp;&nbsp;&nbsp;Node ID • &nbsp;&nbsp;&nbsp;Object ID • &nbsp;&nbsp;&nbsp;Object name • &nbsp;&nbsp;&nbsp;Comment 	section_assign.log
order-delete	Removal of webshop orders	<ul style="list-style-type: none"> • &nbsp;&nbsp;&nbsp;Timestamp • &nbsp;&nbsp;&nbsp;IP address • &nbsp;&nbsp;&nbsp;User (name:ID) 	order_delete.log

		<ul style="list-style-type: none"> • Order ID • Comment 	
--	--	---	--

Configuration

By default, the auditing feature is turned off. In order to use audit trailing on your site, enable the "Audit" setting located in the "[AuditSettings]" section of an override for the "audit.ini" configuration file. Using the "AuditFileNames" configuration array located in the same file, you can specify which types of activities that should be logged (which audit functions that should be used) and to which files they should be logged. The audit function names must be the array keys and the log file names should be the values. Note that the default configuration logs everything to a collection of files (refer to the table in the previous section for details).

The "LogDir" setting can be used to specify where the audit log files should be stored. The default directory is "var/log/audit".

Example

Let's say that you wish to audit successful log-in attempts and changes to roles and policies while ignoring all other activities. Start by creating an override for "audit.ini" and making sure that it contains the following lines:

```
[AuditSettings]
Audit=enabled
LogDir=var/log/my_audit
AuditFileNames []
AuditFileNames[user-login]=login.log
AuditFileNames[role-change]=role_change.log
```

Information about successful log-in attempts will end up in the "login.log" file. Information about role and policy changes will be put in the "role_change.log" file. Both files will be located in the "var/log/my_audit" directory. Each record in these files will contain a time-stamp pointing to the exact date and time when an operation was performed, which user that is associated with it (user name and ID number) and which IP address the request came from. Records related to role and policy changes will have additional information.

Creating new audit functions

This section provides tips for PHP developers who want to create their own audit functions.

Sometimes you may need to create a new audit function, i.e. to make the system log information about a specific operation to a particular audit log file. For example, if you wish to create a new audit function called "my-new-audit" and store information about operations to a file called "info.log", you can do the following:

1. Make sure that the "Audit" setting located in the [AuditSettings] section of an override for "audit.ini" is enabled and add a new element to the "AuditFileNames" configuration array by inserting the following line:

```
AuditFileNames[my-new-audit]=info.log
```

2. In the PHP code which defines the operation that should be logged, you can do something like this:

```
eZAudit::writeAudit( 'my-new-audit', array( 'User id' => $userID,  
      'Comment' => 'The operation XYZ was performed.' ) );
```

Elements like 'Name of something' => <valueOfSomething> define which information that should be written to the "info.log" file when the operation is performed.

For example, a record in the log file can look like this:

```
[ May 23 2007 14:44:04 ] [127.0.0.1] [anonymous:10]  
User id: 10  
Comment: The operation XYZ was performed.
```


4.3 Policy functions

The built-in access control mechanism of eZ Publish is based on roles and policies. A policy is a rule that grants access to a specific function or all functions of a module (page 150). The functions are assigned to the module's views and thus the access requirements for a view are controlled by the functions that are assigned to it.

The following code (taken from the eZ Publish source) shows how the function-view assignments of the "notification" module are specified in "kernel/notification/module.php".

```
<?php

$Module = array( "name" => "eZNotification",
                "variable_params" => true );

$ViewList = array();
$ViewList["settings"] = array(
    "functions" => array( 'use' ),
    "script" => "settings.php",
    'ui_context' => 'administration',
    "default_navigation_part" => 'ezmynavigationpart',
    "params" => array( ),
    'unordered_params' => array( 'offset' => 'Offset' ) );

$ViewList["runfilter"] = array(
    "functions" => array( 'administrate' ),
    "script" => "runfilter.php",
    'ui_context' => 'administration',
    "default_navigation_part" => 'ezsetupnavigationpart',
    "params" => array( ) );

$ViewList["addtonotification"] = array(
    "functions" => array( 'use' ),
    "script" => "addtonotification.php",
    'ui_context' => 'administration',
    "default_navigation_part" => 'ezcontentnavigationpart',
    "params" => array( 'ContentNodeID' ) );

$FunctionList['use'] = array( );
$FunctionList['administrate'] = array( );

?>
```

As the code shows, there are three views and two functions assigned to them. While the "administrate" function is assigned to the "runfilter" view, the "use" function is assigned to the "addtonotification" and "settings" views.

Multiple function assignments

A view can have several functions assigned to it. From version 3.9.3, the system makes use of logical operators ("and", "or") within the function-view assignments. The following examples show how this works.

Example 1

The "tipafriend" view of the "content" module has two functions assigned. The following code is taken from "kernel/content/module.php".

```
$ViewList['tipafriend'] = array(
    'functions' => array( 'tipafriend', 'read' ),
    'default_navigation_part' => 'ezcontentnavigationpart',
    'script' => 'tipafriend.php',
    'params' => array( 'NodeID' ) );
```

The code in this example specifies that a user must be granted access to both the "tipafriend" and "read" functions in order to use the "tipafriend" view (which is a part of the "content" module). Note that there is an alternate way of specifying this, refer to the example below.

```
...
'functions' => array( 'tipafriend and read' ),
...
```

Also, note that the "and" operator can be either "and" or "&&".

Example 2

The "list" view of the "section" module has three functions assigned. The following code is taken from "kernel/section/module.php".

```
$ViewList['list'] = array(
    'functions' => array( 'view or edit or assign' ),
    'script' => 'list.php',
    'default_navigation_part' => 'ezsetupnavigationpart',
    "unordered_params" => array( "offset" => "Offset" ),
    'params' => array( ) );
```

The code above specifies that a user must be granted access to either the "view" or the "edit" or the "assign" function in order to use the "list" view (which is a part of the "section" module). Note that that the "or" operator can be either "or" or "||".

Missing functions

Some modules do not have functions (for example, this is true for the "search" and "collaboration" modules). If this is the case, granting users access to that module means that the users have access to all of the module's views.

But in cases where a module has both views with functions assigned as well as views without functions assigned, only users with access to the entire module will have access to the views were no functions are assigned.

Additional notes for earlier versions

In eZ Publish versions prior to 3.9.3 (except 3.8.9 and later versions of the 3.8 branch), granting access to a function of a module means that the user(s) will get access to the following:

- Views that have the function assigned.
- Views that do not have any functions assigned.

For example, in eZ Publish version 3.9.2, there are no functions assigned to the "discountgroupview" view of the "shop" module. Anonymous users that have access to the "buy" function of the "shop" module can access the "discountgroupview" view (along with other views of the "shop" module that do not have any functions assigned to them). This was changed in versions 3.10.0 beta1, 3.9.3 and 3.8.9 because of security reasons. Refer to the [release announcement](#) for more information.

In order to optimize the functionality of the access permissions when using earlier versions, it is best that modules either have views with functions assigned or views without functions assigned to them, but not both.

Function limitations

A policy (which grants access to a module's function) can be further restricted by function limitations. This can only be done if the function itself supports limitations. A function may support none, one or several limitations. The following code shows how the available limitations for the "diff", "hide" and "tipafriend" functions of the "content" module are specified in "kernel/content/module.php".

```
...
$FunctionList['diff'] = array( 'Class' => $ClassID,
'Section' => $SectionID,
'Owner' => $Assigned,
'Node' => $Node,
'Subtree' => $Subtree);
...
$FunctionList['hide'] = array( 'Subtree' => $Subtree );
```

```
...  
$FunctionList['tipafriend'] = array();  
...
```

As the code shows, the "diff" function supports five limitations, the "hide" function supports one limitation and the "tipafriend" function supports no limitations. Refer to the "Access control (page 161)" section of the "Concept and basics" chapter for an overview of the available function limitations.

4.4 Multi-language

In eZ Publish 3.7 and earlier versions, you have to specify one primary / main language that affects every content object (i.e. each object must exist at least in this language). In addition, you are allowed to specify additional languages which the content objects can be translated to. The multi-language functionality is implemented at the version level and allows an object's version to exist in several languages (a language in this case is referred to as a *translation*). One disadvantage of the old solution is that when several translations are needed, only one translator can work on the object. In other words, the translators must work sequentially and thus wait for each other because only one user is allowed to edit an object's version. This functionality has been changed.

From 3.8, there is no need for primary / main language anymore. You can have for example one article which is only available in English and another article which is only available in French. After choosing the languages for your content objects, it is possible to translate them to any of these languages. The translations of the same object can be created and edited separately and simultaneously by multiple users (a user only edits one version and translation at a time). The next subsections will briefly explain some main principles and terms that will be used when describing the multi-language functionality.

Translatable class attributes

From 3.9, it is possible to translate the class names and the attribute names. In other words, you can for example have "Car" and "Bil" as class names in English and Norwegian along with "Top speed" and "Topp hastighet" as attribute names. Refer to the "Translatable class attributes (page 299)" documentation page for more information.

Locales

A *locale* is a set of country specific settings i.e. language, character sets, number formats, currency format, date and time format, abbreviations of months and weekdays etc. eZ Publish provides many default locale settings where each locale is described in an INI file located in the "share/locale" directory. These configuration files are named according to locale identifiers.

A locale identifier consists of a three-letter language code and a two-letter uppercase country code e.g. "eng-US" (English, USA) or "nor-NO" (Norwegian, Norway). Language and country codes are specified by [ISO 639](#) and [ISO 3166-1 alpha-2](#) standards accordingly.

eZ Publish uses the "eng-GB" locale by default. Please refer to the "Configuring your site locale (page 288)" section for information about setting locale for your site, translating the administration interface, creating custom locales etc.

Default language

From 3.8, the "ContentObjectLocale" INI setting does not specify the primary / main language but the *default language*. This language will be used as the default value in PHP functions that

support an optional parameter for language. The default value of this INI setting is "eng-GB".

Example

Let's say that you have specified "nor-NO" in the "ContentObjectLocale" setting. In this case, if you try to instantiate an object of some class using the "eZContentClass::instantiate()" function and do not explicitly specify the language to use then the Norwegian language will be used.

Translatable country names

From 3.9, it is possible to translate the names of countries to different languages. For example, you can instruct the system to use "Frankrike" instead of "France" and "Norge" instead of "Norway" whenever the list of countries is displayed on a siteaccess where Norwegian locale is used. Refer to the "Translatable country names (page 304)" documentation page for more information.

Translation languages

It is possible to choose the languages that you wish your content to be created in and/or translated to. This set of languages is referred to as *translation languages*. These can be managed via the administration interface (page 284). The maximum number of languages that can be used simultaneously is 30.

Initial/main language

An object can be created in any of the languages that have been added either using the setup wizard or the "Languages" part of the "Setup" tab in the administration interface. When an object is created, its *initial/main language* will be set to the language that was used during creation. For example, if an article is created in Hungarian, its initial/main language will of course be set to Hungarian.

Content that is in the initial/main language can not be removed from the object. However, if the contents of the object exists in several languages, the initial/main language can be changed and thus content that is not in the initial/main language can be removed. Changing the initial/main language and removing languages/translations from an object can be done from within the "Languages" window (in the first three tabs) in the administration interface.

Important note

Please note that the terms "initial language" and "main language" refer to the same thing. While the code and database tables use "initial language", the administration interface uses "main language". This inconsistency will hopefully be fixed in a future release.

Site languages

From 3.8, you can specify which languages the contents of a site should be displayed in. This set of languages is referred to as *site languages*. These languages can be controlled per siteaccess using the "SiteLanguageList[]" configuration setting located under the "[RegionalSettings]" section of the siteaccess "site.ini.append.php" file. You can specify the site languages and their priorities by adding the corresponding locale identifiers to this array. The languages that appear at the top will get higher priority than the others. The first element in this array determines *the most prioritized language*. The system will try to display content in this language first. If an object is not translated to this language then the second prioritized language (specified as the second element of the array) will be displayed. If an object does not exist in this language, the third prioritized language will be used and so on. If an object does not exist in any of the site languages, it will not be shown unless it is always available or if you configure the siteaccess to display untranslated content.

Please note that if the "SiteLanguageList" setting is not specified, the system will use the old "ContentObjectLocale" setting and thus only the default language will be shown.

Example

Let's say that your translation languages are English, French and Norwegian. If you specify two of them as site languages for your public siteaccess (for example English as the most prioritized language and French as the second prioritized one), the system will display content in these two languages to your visitors while Norwegian content will not be shown. If you create one article in English, another one in French and the third one in Norwegian, then only first and second article will be displayed. If you translate the third article from Norwegian into one of the site languages, the translated version of the article will be displayed on your site while the original Norwegian version will still not be shown. If an article is available in both English and French, it will be displayed in English (since English is the most prioritized language for the siteaccess).

Please refer to the "Configuring the site languages (page 292)" section for more information about site languages.

Objects which are always available

Some objects always need to be available even if they do not exist in any of the languages that are configured for a siteaccess. For example, the system must be able to fetch user objects no matter which siteaccess is used. Because of this, a new flag called "always available" has been introduced at the object level. It makes it possible to individually control the availability of the different objects. When an object doesn't exist in any of the site/prioritized languages and it is always available, the system will use the object's initial/main language to display its contents.

The default object availability can be controlled on the class level. By default this setting is enabled for the "Folder", "User", "User group", "Image", "File" etc. classes, so that objects of these classes will be marked as "always available" when created. Changing the default setting at the class level will not affect the existing objects because it simply dictates the initial value for

the "always available" flag which is stored for each object.

Example

Let's say that one of your folders only exists in Norwegian, is marked as always available and contains several articles (the articles are in English, French and Norwegian and none of them are marked as always available). If you specify English and French as site languages for your public siteaccess, this folder will still be displayed since it's always available. Your visitors will thus be able to view the articles that are in it. If the folder is not marked as always available then it will not be displayed and thus your visitors will not be able to read the articles located under it until you translate the folder itself into English or French.

4.4.1 Configuring your site locale

eZ Publish uses the "eng-GB" locale by default. This behavior is determined by the "Locale" INI setting located in the "[RegionalSettings]" block of the "settings/site.ini" configuration file. If you wish to use another locale for your site then you have to override this setting. Please note that the specified locale will be used as the default value for the "l10n" operator unless you explicitly specify the desired locale when using this operator in your templates. The following examples demonstrate how you can set the site locale.

Example 1

Let's say that you need to use "nor-NO" as system locale for all your siteaccesses. The following instructions reveal how this can be done.

1. Open the "site.ini.append.php" configuration file located in the "settings/override" directory and edit it (if the file does not exist, create it).
2. Add the following lines under the "[RegionalSettings]" block:

```
Locale=nor-NO
```

3. Clear the caches.

The system will start to use the locale settings specified in the "share/locale/nor-NO.ini" file for all your siteaccesses.

Example 2

Let's say that you need to use the "nor-NO" locale for one of your siteaccesses. To do this, edit the "site.ini.append.php" file located in the "settings/siteaccess/example/" directory (where "example" is the name of your siteaccess) as described in the previous example and make sure that no locale is specified in the "settings/override/site.ini.append.php" file. After clearing the caches, the "example" siteaccess will start to use the "nor-NO" locale. However, this may not result in the translation of all parts of the interface for this siteaccess (like "Login" and "Sign up" links/buttons etc.) into Norwegian. To do this, you should add the following line under the "[RegionalSettings]" block of the siteaccess "site.ini.append.php" file:

```
TextTranslation=enabled
```

This will instruct the system that the strings marked with "i18n" in the templates should be translated according to the current locale. This means that if you set the "nor-NO" locale for your admin siteaccess and enable the "TextTranslation" setting then everything in the administration interface will be translated into Norwegian. (The "TextTranslation" configuration setting is disabled by default.)

You can also specify different locales for the remaining siteaccesses in the same way, otherwise the default "eng-GB" locale will be used for them.

Adding missing locales

eZ Publish provides many default locale settings where each locale is described in an INI file named by the locale identifier and placed in the "share/locale" directory. To contribute new locales and/or translations to eZ Publish, visit the Community Translation project page here: http://projects.ez.no/ezpublish_translation, log-in, navigate to the team page (http://projects.ez.no/ezpublish_translation/team/members) and request membership. You will be promptly able to contribute new translations and locale to eZ Publish ! ". The following example demonstrates how to add a missing locale.

Example

Let's say that you need to use the "ell-GR" locale on your site. To do this, download Greek translation for eZ Publish from the eZ Publish translation pages http://projects.ez.no/ezpublish_translation (you have to be a member as described in "Adding missing locales above") and unpack it into a temporary location. You should see there a sub-directory called "share" which contains the locale configuration file (share/locale/ell-GR.ini) and the translation file for eZ Publish (share/translations/ell-GR/translation.ts). In addition, the downloaded package may contain a flag icon (share/icons/flags/ell-GR.gif) and/or translation file(s) for some extensions located in the "extension" sub-directory.

Please note that the "translation.ts" file contains eZ Publish specific strings translated into Greek language (the strings that are used in the templates and PHP code). If the "TextTranslation" setting is enabled, the strings from this file will be used for translating different parts of the interface, system messages, warnings etc.

If you copy the "share" sub-directory to the root directory of your eZ Publish installation, set the "ell-GR" locale (as described in the previous two examples) and clear the caches, the system will start to use Greek locale.

Custom locales

In addition to the default locale settings that come with eZ Publish, it is possible to create custom locales. For every custom locale you must have either a translation.ts or a fallback other than "eng-GB". The fallback is necessary when you wish to use the translation cache. The fallback is made in "i18n.ini". For a custom locale, you still need to also update "i18n.ini" by an override.

Note that the locale must fallback to the *untranslated* language.

We strongly recommend this fallback procedure, since setting up this fallback will make the system use the translation cache.

Note: no fallback = no translation cache used = bad performance

The following examples demonstrate how this can be done.

Example 1

Let's say that you wish to use Icelandic locale on your site. You can create a custom locale configuration file for this based on the "eng-GB" locale settings. To do this, do the following:

1. Go to the "share/locale" directory and copy the "eng-GB.ini" configuration file to a new file called "ice-IS.ini".
2. Open this file and edit the locale settings.
3. Set your site locale to "ice-IS".

Example 2

Let's say that you wish to modify the Norwegian locale. The original locale file should not be changed because it will be overwritten next time you upgrade eZ Publish. Instead, you should create a custom locale file based on the original "nor-NO" locale settings. The following text reveals how this can be done.

1. Go to the "share/locale" directory and copy the "nor-NO.ini" configuration file to a new file called "nor-NO@custom.ini".
2. Open the newly copied file and edit the locale settings.
3. Make sure that your siteaccess uses the "nor-NO@custom" locale.
4. Clear the caches.

Standard for naming eZ locales

You must decide the locale code of your language. eZ Publish uses locale codes on the form aaa-AA, where the 3 first lowercase letters describe the language, while the last two uppercase letters describe the country in which the language is spoken. For instance, English as it is spoken in Great Britain would be eng-GB, while US English is eng-US.

Countries are specified by the ISO 3166 Country Code: <http://www.iso.ch/iso/en/prods-services/iso3166ma/index.html>

Language is specified by the ISO 639-2 Language Code: http://www.loc.gov/standards/iso639-2/php/code_list.php

You can also create a variation of a locale, you will for instance find two variations of nor-NO, nor-NO@intl and nor-NO@spraakraad, that are slight modifications of the original.

This list shows the current ini-files with eZ locales in use in eZ Publish installations:

Locales ini file	Language (Country)
cat-ES.ini	Catalan (Spain)
chi-CN.ini	Chinese (China)
chi-HK.ini	Chinese (Hong Kong)
chi-TW.ini	Chinese (Taiwan)
cro-HR.ini	Croatian (Croatia)
cze-CZ.ini	Czech (Czech Republic)
dan-DK.ini	Danish (Denmark)
dut-NL.ini	Dutch (Netherlands)
ell-GR.ini	Greek (Greece)
eng-AU.ini	English (Australia)
eng-CA.ini	English (Canada)
eng-GB.ini	English (Great Britain)
eng-GB@euro.ini	English (Euro support)
eng-NZ.ini	English (New Zealand)
eng-US.ini	English (USA)
esl-ES.ini	Spanish (Spain)
esl-MX.ini	Spanish (Mexico)
fin-FI.ini	Finnish (Finland)
fre-BE.ini	French (Belgium)
fre-CA.ini	French (Canada)
fre-FR.ini	French (France)
ger-DE.ini	German (Germany)
ger-DE@euro.ini	German (Euro support)
heb-IL.ini	Hebrew (Israel)
hin-IN.ini	Hindi (India)
hun-HU.ini	Hungarian (Hungary)
ind-ID.ini	Indonesian (Indonesia)
ita-IT.ini	Italian (Italy)
jpn-JP.ini	Japanese (Japan)
jpn-JP@international.ini	Japanese (Kanji)
kor-KR.ini	Korean (South Korea)
nno-NO.ini	Norwegian - Nynorsk (Norway)
nor-NO.ini	Norwegian - Bokmål (Norway)
nor-NO@intl.ini	Norwegian (intl)
nor-NO@spraakraad.ini	Norwegian (Standard set by Språkrådet)
pol-PL.ini	Polish (Poland)
por-BR.ini	Portuguese (Brazil)
por-MZ.ini	Portuguese (Mozambique)
por-PT.ini	Portuguese (Portugal)
por-PT@euro.ini	Portuguese (Euro support)
rus-RU.ini	Russian (Russia)
ser-SR.ini	Serbian (Srpski)
slk-SK.ini	Slovak (Slovakia)
srp-RS.ini	Serbian (€)
swe-SE.ini	Swedish (Sweden)
tur-TR.ini	Turkish (Turkey)
ukr-UA.ini	Ukrainian (Ukraine)

4.4.2 Configuring the site languages

No site languages are specified by default (i.e. after downloading and unpacking the eZ Publish distribution). During the installation process, the setup wizard allows the user to choose the languages that should be used on the site which is being created. The list of available languages displayed at this step is built using the INI files located in the "share/locale" directory. Use the radio buttons to choose the default language (required), and the checkboxes to choose the additional languages (optional).

Please note that choosing the default language at this step will affect both default language and system locale. Please note that one of the radio buttons will be pre-selected i.e. the default language will be specified according to the language settings of your browser. However, you can choose another language instead. If you select for example "German", then both locale and default language will be set to "ger-DE" and your administration interface will be translated into German (in addition, the "TextTranslation" setting will be enabled).

All the selected languages will be added to the system as translation languages and recorded as site languages for both public and admin site accesses. The default language will be recorded as the most prioritized language. You will be able to use any of these languages for creating and translating your content after the setup wizard is finished. It is also possible to add new translation languages using the administration interface and to change the site languages configuration by editing your configuration settings.

Displaying untranslated content

Since it may be useful to display all translation languages, an additional configuration setting called "ShowUntranslatedObjects" has been added. It can be set to either "enabled" or "disabled". If this setting is enabled, the system will still use the language priorities determined by the "SiteLanguageList[]" array, but it will not filter away languages that are not on the list. In other words, all objects will be displayed regardless of which language they exist in - and objects that exist in a language specified in the priority list will be displayed using the prioritized language.

The "ShowUntranslatedObjects" setting is disabled by default. However, the setup wizard usually enables it for the admin site access. This allows the site administrator to create and edit objects in any of the translation languages even if some of these languages are not listed as site languages.

Example

Let's say that you have selected British English as default language, French and Norwegian as additional languages (look at the following screenshot).

(see figure 4.5)

In this case, you will have the following settings for locale, default language and site languages after the setup wizard is finished:

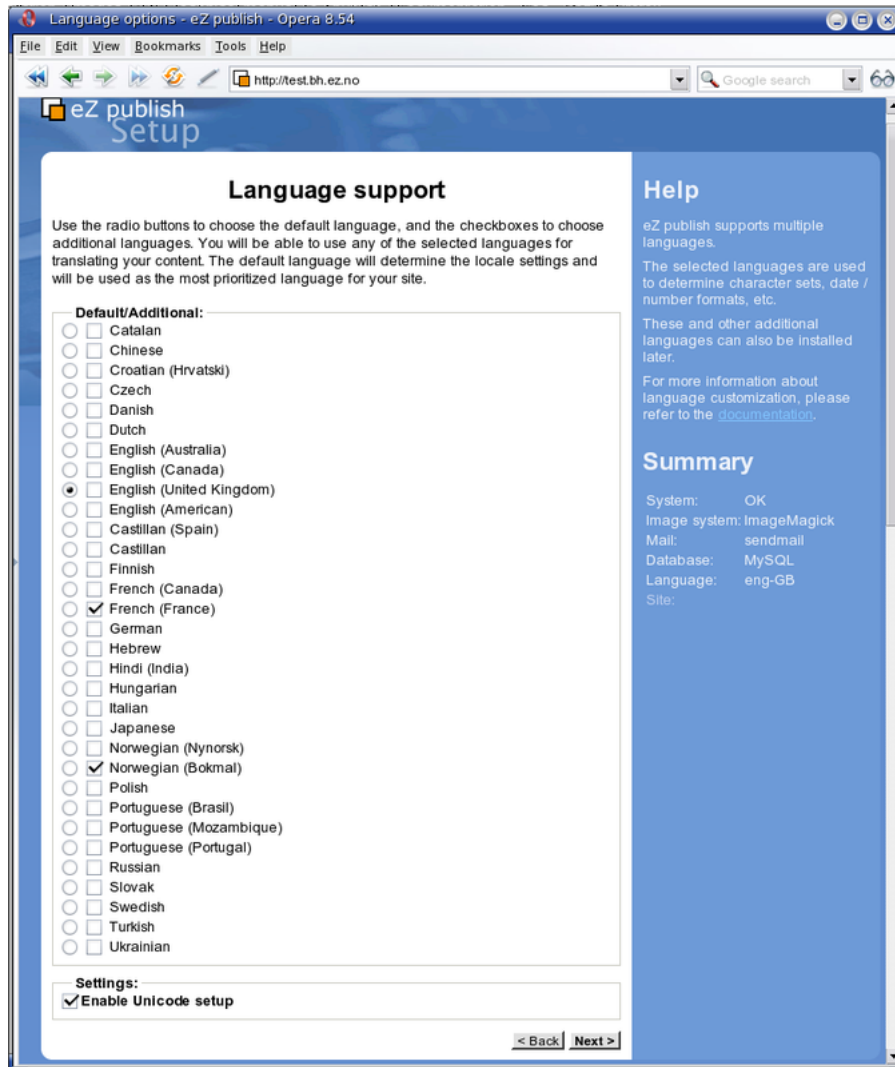


Figure 4.5: The language selection step in the setup wizard.

```
[RegionalSettings]
Locale=eng-GB
ContentObjectLocale=eng-GB
SiteLanguageList []=eng-GB
SiteLanguageList []=fre-FR
SiteLanguageList []=nor-NO
```

This means that the site locale is set to "eng-GB", the default language is English, the most prioritized language is English, the second prioritized language is French and the third prioritized language is Norwegian. The setup wizard will put these settings into the "site.ini.append.php" files for both public and admin siteaccesses. The "TextTranslation" setting will be disabled for both siteaccesses because the "eng-GB" locale is used.

Any of these three languages can be used for creating and translating your content. You can change the site language configuration later by editing the "site.ini.append.php" file for the desired siteaccess.

The setup wizard will add one more line in the "site.ini.append.php" file for the admin siteaccess:

```
ShowUntranslatedObjects=enabled
```

This will tell the system to make all the translation languages available when working with content objects in the administration interface.

You can add new translation languages using the admin interface. Let's go to "Setup - Languages" and add German.

(see figure 4.6)

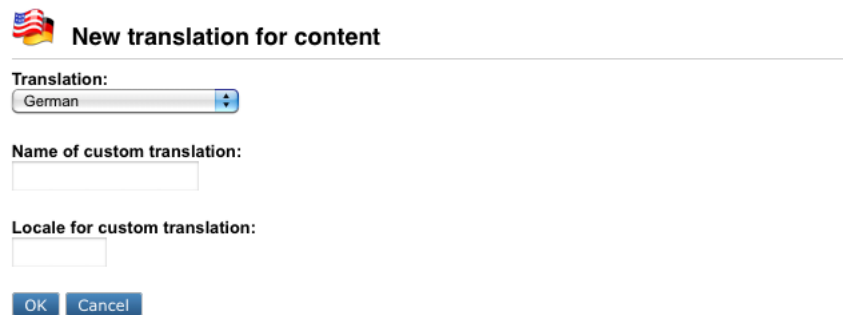


Figure 4.6:

This language will not be displayed on your site (public siteaccess) because it is not included in the list of site languages (i.e. not specified in the "SiteLanguageList[]" array). However, after clearing the caches, German will be displayed as the last item in the drop-down list of available languages for object creation in the administration interface (look at the next screenshot) because the "ShowUntranslatedObjects" setting is enabled for the admin siteaccess.

(see figure 4.7)

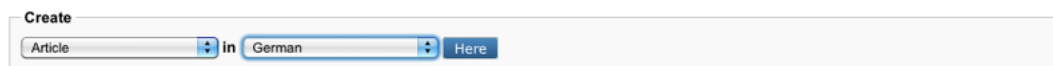


Figure 4.7:

Changing the language priorities

The "SiteLanguageList[]" setting specified in the siteaccess "site.ini.append.php" file contains the prioritized list of site languages where items appearing at the top get higher priority than the

others. The system will try to display content in the most prioritized language first. If an object is not translated to this language then the second prioritized language will be displayed. If an object does not exist in this language then the third prioritized language will be used and so on. If an object does not exist in any of the site languages, it will not be shown unless it is always available or if you configure the siteaccess to display untranslated content.

To change the site language priorities, open the configuration file, edit it and re-arrange the elements of this array in the desired way.

Example

Let's say that the following settings are specified in the "site.ini.append.php" file for your public siteaccess:

```
[RegionalSettings]
SiteLanguageList []
SiteLanguageList []=eng-GB
SiteLanguageList []=fre-FR
SiteLanguageList []=ger-DE
SiteLanguageList []=nor-NO
```

If an article exists in French and Norwegian languages, the system will follow the prioritized list of site languages and display the article in French which is the second prioritized language. This behavior will not change if you translate this article into German (the third prioritized language). However, if you translate the article into English (which is the most prioritized language), then it will be displayed in English.

If you move the line "SiteLanguageList []=nor-NO" to the top of the list, then Norwegian will become the most prioritized language. This will instruct the system to display content in Norwegian and use other site languages only when a Norwegian translation is not available.

Using several public siteaccesses

In the previous example only one public siteaccess was used. A multi-language site typically uses several public siteaccesses. If your site content exists in for example English and French then it is recommended to have two public siteaccesses with the following language configuration:

Siteaccess "gb"	Siteaccess "fr"
<pre>[RegionalSettings] SiteLanguageList [] SiteLanguageList []=eng-GB</pre>	<pre>[RegionalSettings] SiteLanguageList [] SiteLanguageList []=fre-FR SiteLanguageList []=eng-GB</pre>

If an article exists only in English, it will be displayed to the visitors of both siteaccesses (because English is the only site language for the "gb" siteaccess and the second prioritized language for

the "fr" siteaccess). If you translate this article into French, it will be shown in French when viewing the "fr" siteaccess (since French is the most prioritized language for this siteaccess). If an article exists only in French, it will be available for the visitors of the "fr" siteaccess but it will not show up in the "gb" siteaccess.

Now, let's say that you wish to start using for example Norwegian language on your site. In this case, you will probably add Norwegian as a new translation language, create a new siteaccess called "no" and specify the following settings in the "site.ini.append.php" file of the newly created siteaccess:

```
[RegionalSettings]
SiteLanguageList []
SiteLanguageList []=nor-NO
```

This will tell the system to use Norwegian as the only site language for this siteaccess. In other words, if an article does not exist in Norwegian, it will not be displayed.

Of course, it is possible to add the following line to these settings:

```
SiteLanguageList []=eng-GB
```

In this case Norwegian will be the most prioritized language for the "no" siteaccess and English will be the second prioritized one (look at the next table).

	Siteaccess "gb"	Siteaccess "fr"	Siteaccess "no"
The most prioritized language	eng-GB	fre-FR	nor-NO
The second prioritized language	-	eng-GB	eng-GB
The third prioritized language	-	-	-

Articles that exist only in English will be displayed in English to the visitors of all three siteaccesses. If an article exists only in Norwegian, it will be shown only on the "no" siteaccess.

Let's create a new article called "Lundi" (Monday) in French. This article will be displayed to the visitors of the "fr" siteaccess but not to the visitors of the "gb" and "no" siteaccesses (because French is not listed as site language for these siteaccesses). If you translate this article into Norwegian then it will become available as "Mandag" (Monday) when viewing the "no" siteaccess but still invisible for the users of the "gb" siteaccess. If you add English translation for this article, it will become available as "Monday" for the visitors of the "gb" siteaccess. However, nothing will change for the "fr" and "no" siteaccesses because English is their second prioritized language.

4.4.3 Managing the translation languages

The administration interface allows you to manage the translation languages (page 297) for your site. This can be done by manipulating the global translation list. To access the list of translation languages, click the "Setup" tab in the administration interface and select the "Languages" link on the left. (This interface can also be accessed by requesting `"/content/translations"` in the URL.)

The following screenshot shows how this list looks like.

(see figure 4.8)

Available languages for translation of content (2)

<input type="checkbox"/> Language	Country/region	Locale	Translations	Classes translations
<input type="checkbox"/> English (United Kingdom)	United Kingdom	eng-GB	57	39
<input type="checkbox"/> French (France)	France	fre-FR	0	0

Remove selected Add language

Figure 4.8:

The last column of the list contains information about the number of translations i.e. how many content objects are translated into each language. The screenshot shows a situation when all the objects exist in English but they are not translated to French. If you click on a language name, the system will display information about this language and its locale settings.

The next subsections explain how the translation languages can be added and/or removed using this interface.

Adding a new language

You can add a new translation language by clicking the "Add language" button and selecting the desired language from the drop-down list called "Translation" (look at the next screenshot). Please note that the contents of this list depends on the available locales represented by the INI files in the "share/locale" directory. If you wish to use a language which is not available here then you need to add the missing locale (page 288) first.

(see figure 4.9)


Click "OK" to save your changes. After clearing the caches, you will be able to use this language for your content objects.

Removing a language

You can only remove a language if there are no content objects using it (when the "Translations" column contains "0" for a language).

To remove one or more languages from the system, select the languages that you wish to remove (use the check-boxes located in the first column) and click the "Remove selected" button.

Share information

 **New translation for content**

Translation:

Name of custom translation:

Locale for custom translation:

Figure 4.9:

4.4.4 Translatable class attributes

In eZ Publish 3.9.0 and later versions, it is possible to translate the names of the attributes when editing the different classes. This allows the system to display the attribute labels in the correct language when users are working with (both editing or viewing) the different translations. For example, if a class is being used to store information about cars in both English and Norwegian, it is a good idea to translate the names of the class attributes so that the "Color" attribute of the cars would appear as "Color" or "Farge" depending on whether the object is being edited in English or Norwegian.

A class can be created using any of the languages that have been added either using the setup wizard or the "Languages" part of the "Setup" tab in the administration interface. When a class is created, its main language will be set to the language that was used during creation. The class name and the names of the class attributes that are in the main language can not be removed from the class. However, if the class exists in several languages, the main language can be changed and thus the class name and the names of the class attributes that are not in the main language can be removed. Changing the main language and removing languages/translations from a class can be done from within the "Translations" window in the class view interface.

Creating classes in different languages

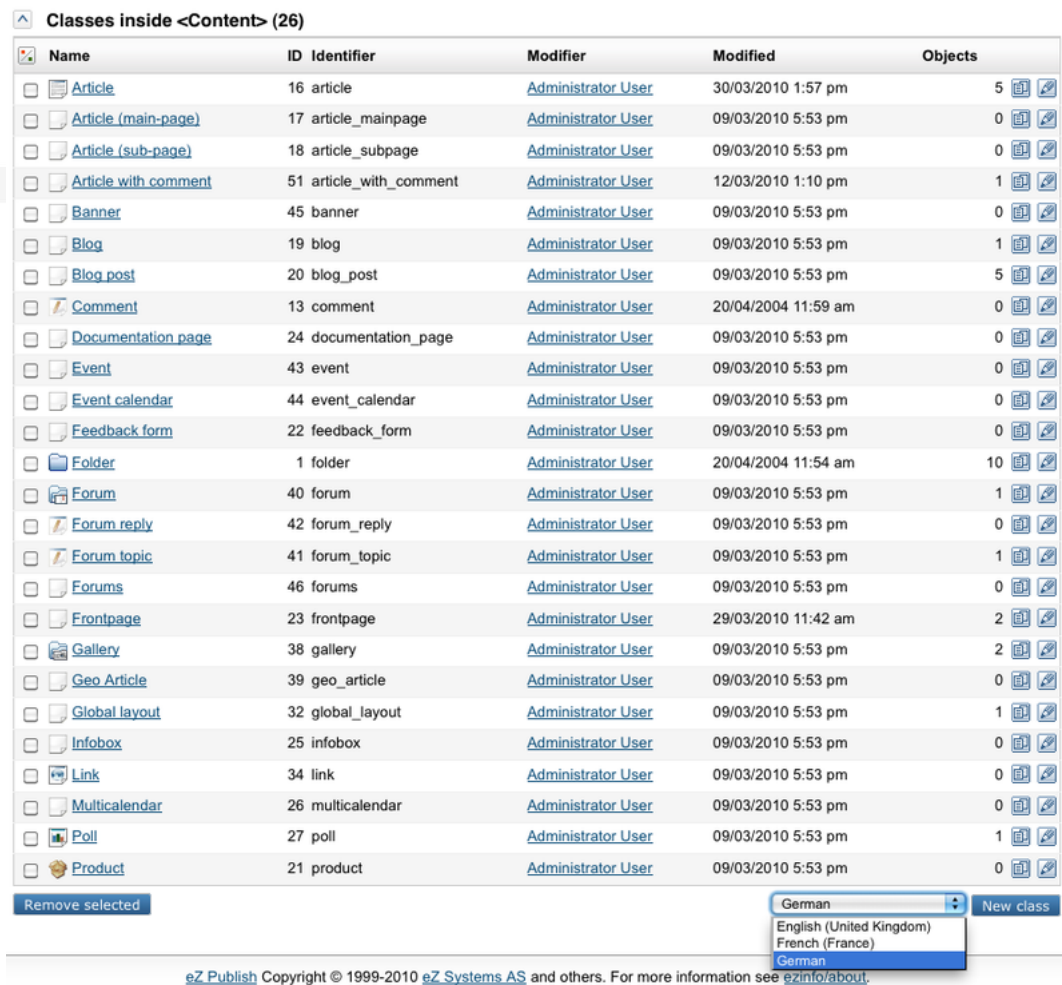
The administration interface allows you to create content classes from scratch using any of the translation languages. The following text reveals how this can be done.

1. Access the "Setup" tab in the administration interface, click "Classes" on the left and select the class group that you wish to add a new class to. You should see the list of classes assigned to this group.
2. Use the drop-down list of languages located in the bottom of the "Classes inside <group_name>" window to choose the desired language for the class that you wish to create. Click the "New class" button (see the following screenshot).

(see figure 4.10)

If the desired language is not listed in the drop-down list, make sure it exists in the global translation list. You can add new languages to this list as described in the "Managing the translation languages (page 297)" section. Note that the newly added languages will become available after the caches have been cleared.

3. You will be taken to the class edit interface, where the language that the class is being edited in will be shown in the top right corner. Specify name, identifier, object name pattern and container flag for the newly created class and add the desired attributes using the drop-down list located in the bottom of the class edit interface. After adding the attributes, click "OK" to save the class.



Classes inside <Content> (26)

Name	ID	Identifier	Modifier	Modified	Objects
Article	16	article	Administrator User	30/03/2010 1:57 pm	5
Article (main-page)	17	article_mainpage	Administrator User	09/03/2010 5:53 pm	0
Article (sub-page)	18	article_subpage	Administrator User	09/03/2010 5:53 pm	0
Article with comment	51	article_with_comment	Administrator User	12/03/2010 1:10 pm	1
Banner	45	banner	Administrator User	09/03/2010 5:53 pm	0
Blog	19	blog	Administrator User	09/03/2010 5:53 pm	1
Blog post	20	blog_post	Administrator User	09/03/2010 5:53 pm	5
Comment	13	comment	Administrator User	20/04/2004 11:59 am	0
Documentation page	24	documentation_page	Administrator User	09/03/2010 5:53 pm	0
Event	43	event	Administrator User	09/03/2010 5:53 pm	0
Event calendar	44	event_calendar	Administrator User	09/03/2010 5:53 pm	0
Feedback form	22	feedback_form	Administrator User	09/03/2010 5:53 pm	0
Folder	1	folder	Administrator User	20/04/2004 11:54 am	10
Forum	40	forum	Administrator User	09/03/2010 5:53 pm	1
Forum reply	42	forum_reply	Administrator User	09/03/2010 5:53 pm	0
Forum topic	41	forum_topic	Administrator User	09/03/2010 5:53 pm	1
Forums	46	forums	Administrator User	09/03/2010 5:53 pm	0
Frontpage	23	frontpage	Administrator User	29/03/2010 11:42 am	2
Gallery	38	gallery	Administrator User	09/03/2010 5:53 pm	2
Geo Article	39	geo_article	Administrator User	09/03/2010 5:53 pm	0
Global layout	32	global_layout	Administrator User	09/03/2010 5:53 pm	1
Infobox	25	infobox	Administrator User	09/03/2010 5:53 pm	0
Link	34	link	Administrator User	09/03/2010 5:53 pm	0
Multicalendar	26	multicalendar	Administrator User	09/03/2010 5:53 pm	0
Poll	27	poll	Administrator User	09/03/2010 5:53 pm	1
Product	21	product	Administrator User	09/03/2010 5:53 pm	0

Remove selected

German
English (United Kingdom)
French (France)
German

New class

eZ Publish Copyright © 1999-2010 eZ Systems AS and others. For more information see [ezinfo/about](#).

Figure 4.10:

Translating classes to different languages

The administration interface allows you to translate the names of content classes and their attributes to any of the translation languages. The following text reveals how this can be done.

1. In the administration interface, locate the class that you wish to edit and click on the name of the class. You will be taken to the class view interface.
2. Select the "Another language" item from the drop-down list and click the "Edit" button as shown in the following screenshot.

(see figure 4.11)

Then you will be taken to the "Edit <New Class>" screen shown below:

(see figure 4.12)



Figure 4.11:

Figure 4.12:

The system will display the language selection interface for content classes (see the following screenshot).

(see figure 4.13)

Figure 4.13:

Use the language selection radio buttons to select the language that you wish to translate the names of the class attributes (in the screenshot above, Norwegian is selected). It is also

possible to choose which existing language the newly translated names should be based on. You can select one of the existing languages or "None". When a language is chosen instead of "None", the system will copy the existing class attribute names from the selected language and allow them to be modified/translated (otherwise, you will have to type in everything from scratch into empty fields).

3. After clicking the "Edit" button, the system will bring up the class edit interface where you should specify the class name and the names of the class attributes in the selected language. When finished, click "OK" to save your changes.

Editing classes in different languages

You can edit a content class in any of the languages that it exists in. The following text reveals how this can be done.

Using the "Edit" button

1. In the administration interface, locate the class that you wish to edit and click on the name of the class. You will be taken to the class view interface.
2. Select the desired language from the drop-down list and click the "Edit" button. The system will bring up the class edit interface where you can change the class name and the names of the class attributes in the selected language. When finished, click "OK" to save your changes.

Using the translations window

1. In the administration interface, locate the class that you wish to edit and click on the name of the class. You will be taken to the class view interface.
2. The horizontally aligned switches in the upper area control the visibility of the different windows. Click on the "Translation" switch to enable the translations window (see the following screenshot).

(see figure 4.14)

Click on the "Translations" window to activate it and make it visible at the bottom of the screen. The following screenshot shows how this window for a class that exists in two languages.

(see figure 4.15)

Locate the language that you wish to edit and click on the language's corresponding edit icon (on the right hand side). The system will bring up the class edit interface.

Changing the main language

If a class exists in several languages then you can choose which of them will be the main language.

You are here: [Class groups](#) / [Content](#) / Documentation page

[Class groups](#) [Override templates](#) [Translations](#)

Setup

- Cache management
- Classes
- Collected information
- Extensions
- Ini settings
- Languages
- Menu management
- PDF export
- Packages
- RAD
- RSS
- Search statistics
- Sections
- Sessions
- States
- System information

Documentation page [0]

Last modified: 15/09/2010 2:17 pm, Administrator User English (United Kingdom)

Name: Documentation page

Identifier: documentation_seite

Description:

Object name pattern: <title>

URL alias name pattern:

Container: Yes

Default object availability: Not available

Default sorting of children: Path String / Ascending

Object count: 0

Figure 4.14:

Translations (2)

Existing languages

Language	Locale	Main
<input checked="" type="checkbox"/> English (United Kingdom)	eng-GB	<input checked="" type="radio"/>
<input type="checkbox"/> German	ger-DE	<input type="radio"/>

[Remove selected](#) [Set main](#)

Figure 4.15:

1. In the administration interface, locate the class that you wish to edit and click on the name of the class. You will be taken to the class view interface.
2. Enable the translations window, select the desired language using the radio buttons and click the "Set main" button.

Removing languages

It is possible to remove languages/translations from a class (except the main language). This can be done from within the "Translations" window in the class view interface. When either one or several languages are selected using the check-boxes (on the left hand side), the "Remove selected" button can be used to carry out the actual removal of the selected translations.

4.4.5 Translatable country names

From 3.9, it is possible to translate the names of countries to different languages. For example, you can instruct the system to use "Frankrike" instead of "France" and "Norge" instead of "Norway" whenever the list of countries is displayed on a siteaccess where Norwegian locale is used. The following example demonstrates how this can be done.

Example

If you wish to replace the English country names of France and Norway with the Norwegian ones on a siteaccess that uses the Norwegian locale, you can do the following.

1. Go to the "share/locale" directory and copy the "nor-NO.ini" configuration file to a new file called "nor-NO@custom.ini".
2. Add the following lines under the "[RegionalSettings]" block:

```
[CountryNames]
Countries []
Countries [FR]=Frankrike
Countries [NO]=Norge
```

3. Set your site locale to "nor-NO@custom" (refer to the "Configuring your site locale (page 288)" documentation page for more information about setting the site locale).

After clearing the caches, the system will use Norwegian country names for France and Norway whenever the list of countries is displayed. The following screenshot shows how a translated country name will appear in the "Default selection" drop-down list when an attribute of the "ezcountry" datatype is being edited.

(see figure 4.16)

9. Country [Country] (id:188)

Name:
Country

Identifier:
country

Required Searchable Information collector Disable translation

Multiple choice

Default selection:

- New Caledonia
- New Zealand
- Nicaragua
- Niger
- Nigeria
- Niue
- Norfolk Island
- Northern Mariana Islands
- Norge**
- Oman
- Other
- Pakistan
- Palau
- Palestinian Territory, Occupied
- Panama
- Papua New Guinea
- Paraguay
- Peru
- Philippines
- Pitcairn

Figure 4.16: List of countries containing translated country names.

4.4.6 Multi-lingual objects

The following text describes how you can create new multilingual objects, make an object always available, set the initial/main language for an object and so on.

Creating new objects

The administration interface allows you to create content objects in any of the translation languages. Use the drop-down list of languages located in the "Create here" interface to choose the desired initial/main language for the object that you wish to create and click the "Create here" button (look at the next screenshot).

(see figure 4.17)

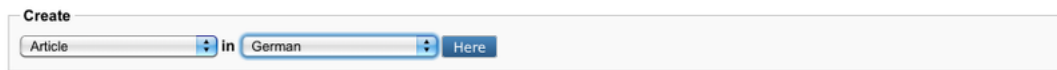


Figure 4.17:

If the desired language is not listed in the drop-down box, do the following:

1. Go to the list of translation languages and add the desired language if it is not listed there as described in the "Managing the translation languages" section.
2. Make sure that the "site.ini.append.php" file of your admin siteaccess contains the following line under the "[RegionalSettings]" block:

```
ShowUntranslatedObjects=enabled
```

The language will become available after clearing the caches.

Changing the initial/main language

If an object exists in several languages then you can choose which of them will be the initial/main language (page 284). Select the desired translation in the translations window using the radio buttons and click the "Set main" button.

Changing the object availability

To make an object always available, select the "Use the main language if there is no prioritized translation" check-box located in the translations window of the object view interface and click the "Update" button.

Default object availability for a class

It is possible to set the default object availability on the class level. By default this setting is enabled for the "Folder", "User", "User group", "Image", "File" etc. classes, so that the new objects of these classes will be marked as "always available" when created. Note that this can be reconfigured for each individual object regardless of the class setting. The following example demonstrates how this can be done.

Example

Let's say that you are going to create a set of articles in English that should be displayed on any siteaccess no matter which site languages are specified for these siteaccesses. You can enable the default object availability setting for your "Article" class so that each newly created article will become "always available" by default. The following instructions reveal how to do this.

1. Access the "Setup" tab in the administration interface, click "Classes" on the left and select the "Content" class group. You should see the list of classes assigned to this group as shown in the following screenshot.

(see figure 4.18)

Class groups (4)

Name	Modifier	Modified
<input type="checkbox"/> Content	Administrator User	06/10/2002 6:35 pm
<input type="checkbox"/> Users	Administrator User	06/10/2002 6:35 pm
<input type="checkbox"/> Media	Administrator User	06/10/2002 6:35 pm
<input type="checkbox"/> Setup	Administrator User	13/04/2004 2:07 pm

Remove selected New class group

Recently modified classes

Name	ID	Identifier	Modifier	Modified	Objects
<input type="checkbox"/> Documentation page	24	documentation_seite	Administrator User	15/09/2010 2:17 pm	0
<input type="checkbox"/> Article	16	article	Administrator User	30/03/2010 1:57 pm	5
<input type="checkbox"/> Frontpage	23	frontpage	Administrator User	29/03/2010 11:42 am	2
<input type="checkbox"/> Article with comment	51	article_with_comment	Administrator User	12/03/2010 1:10 pm	1
<input type="checkbox"/> Infobox	25	infobox	Administrator User	09/03/2010 5:53 pm	0
<input type="checkbox"/> Multicalendar	26	multicalendar	Administrator User	09/03/2010 5:53 pm	0
<input type="checkbox"/> Poll	27	poll	Administrator User	09/03/2010 5:53 pm	1
<input type="checkbox"/> File	28	file	Administrator User	09/03/2010 5:53 pm	0
<input type="checkbox"/> Flash	29	flash	Administrator User	09/03/2010 5:53 pm	0
<input type="checkbox"/> Flash recorder	30	flash_recorder	Administrator User	09/03/2010 5:53 pm	1

Figure 4.18:

Find the "Article" class there and click on the edit icon located in the same line of the list. You will be taken to the class edit interface.

2. Select the "Default object availability" check-box as shown in the screenshot below and click the "OK" button to save your changes.

(see figure 4.19)

OK Apply Cancel Text line Add attribute

Edit <Documentation page> (0)

Last modified: 15/09/2010 2:47 pm, Administrator User English (United Kingdom)

Name:
Documentation page

Identifier:
documentation_seite

Description:

Object name pattern:
<title>

URL alias name pattern:

Container:

Default sorting of children:
Path String Ascending

Default object availability:

Figure 4.19:

Please note that the changes will not affect any of the existing articles. Only new articles will be affected.

4.4.7 Working with translations

You can use the translations window to view the languages that the object exists in. The following text reveals how you can create, edit and remove the object translations.

Editing a translation

All content editing is done through the object edit interface. This interface will automatically be displayed whenever you're editing existing or creating new objects. If an object exists in several languages then you can choose which translation to edit. The following text reveals how you can edit a translation using different approaches.

Using the translations window

1. Use the administration interface to navigate to the object that you wish to edit. In other words, make sure that the object is being displayed.
2. Enable the translations window and locate the language that you wish to edit. Click on the language's corresponding edit icon (on the right hand side). The system will bring up the edit interface.

Using the "Sub items" window

1. Use the administration interface to navigate to the node/object which contains the one that you wish to edit. In other words, make sure that the parent node is being displayed.
2. Look at the "Sub items" window and locate the node/object that you wish to edit. Click on the node's corresponding edit icon (on the right hand side). You will be taken to the *language selection interface* which is described below.
3. Use the radio buttons located in the "Existing languages" frame to select the language that you wish to edit and click the "Edit" button. The system will bring up the edit interface.

The language selection interface

The language selection interface (full or reduced) appears when you have to choose which translation you wish to edit or create. The following screenshot shows the language selection interface for a folder that exists in English.

(see figure 4.20)

As you can see from the screenshot above, the language selection radio buttons are divided into two groups. The "Existing languages" group contains the languages that are already used by the object. This list makes it possible to select an existing translation for editing. The "New languages" group contains a list of the translation languages that are not used by the object. The latter makes it possible to translate the contents of the object into a language that it does not



Figure 4.20:

exist in yet. When adding a new translation, it is possible to choose which existing translation it should be based on. You can select one of the existing languages or "None". When a language is chosen instead of "None", the main part of the edit page will contain translation interface instead of the standard edit interface.

Using the tree menu and the context menu

1. Use the tree menu on the left to locate the object that you wish to edit.
2. Click on the object's icon in order to bring up the context menu.
3. Access the "Edit in" sub-menu and select the language that you wish to edit as shown in the following screenshot.

(see figure 4.21)

The screenshot above shows the content structure pop-up menu for a folder that exists in English and French. After selecting a language, the system will display the edit interface.

Using the "Edit" button

1. Use the administration interface to navigate to the node (page) that you wish to edit. In other words, make sure that the node is being displayed.
2. Use the drop-down list of languages located in the preview window to select the language that you wish to edit and click the "Edit" button (look at the next screenshot).

(see figure 4.22)

The system will bring up the edit interface.

Using the bookmarks

1. Make sure that your bookmarks are being displayed (use the "+" sign to open the window).
2. Locate the object that you wish to edit and click on its icon to bring up the context menu.
3. Access the "Edit in" sub-menu and select the language that you wish to edit. The system will bring up the edit interface.

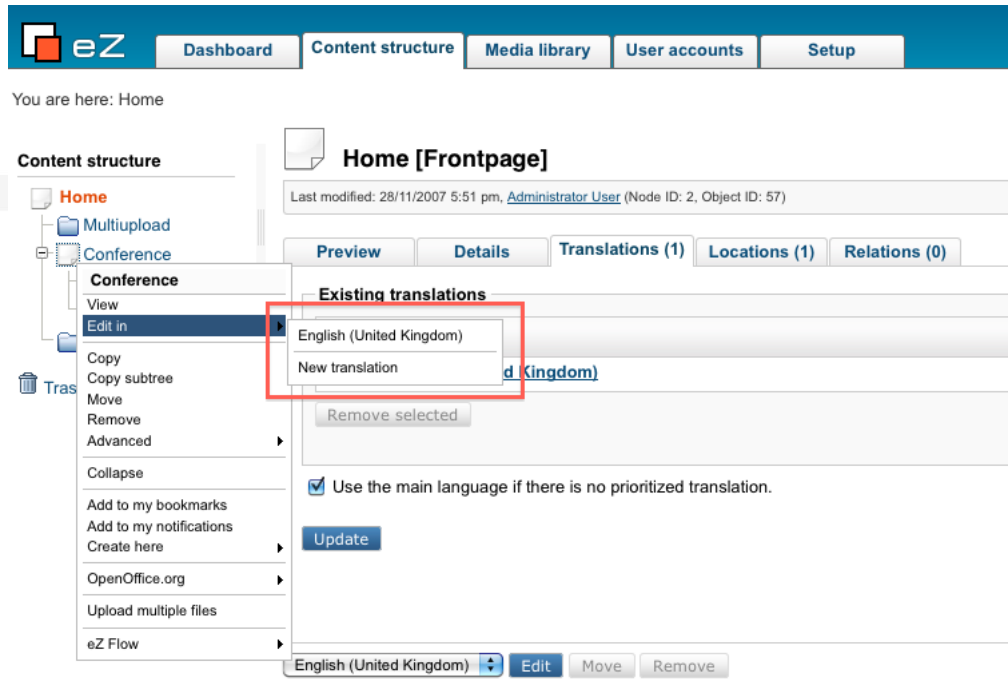


Figure 4.21:

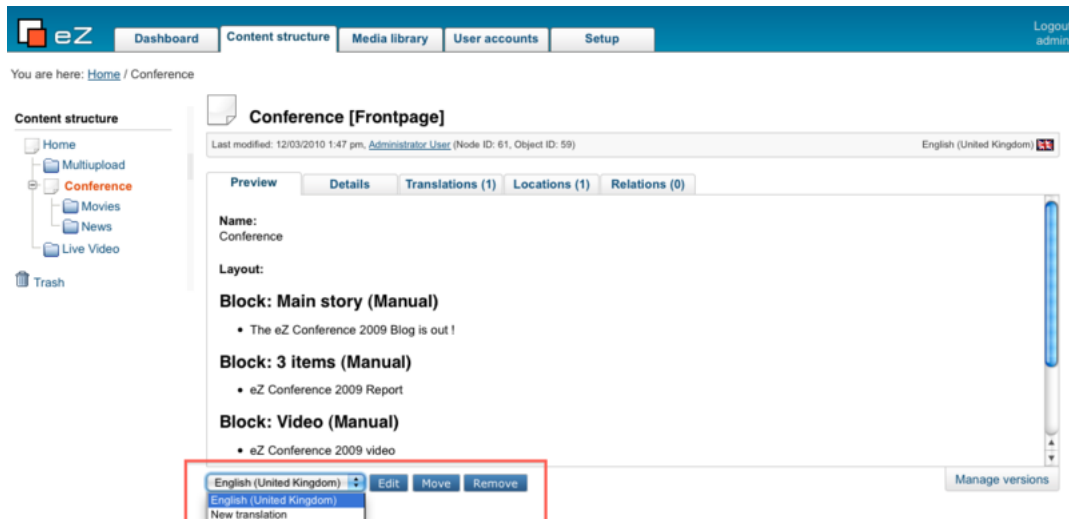


Figure 4.22:

Editing multiple languages

It is possible to edit two or more translations/languages of the same object. Internally the system actually edits two or more versions of the same object. A draft only contains object attribute data

for one language. When the draft is published, the system will copy all other languages from the previously published version.

The following screenshot shows the draft list interface when the user is editing two translations of the same article (this interface can be accessed by clicking the "Dashboard" tab and selecting the "My drafts" link on the left).

(see figure 4.23)

My drafts (2)

10 25 50

<input type="checkbox"/>	Name	Type	Section	Language	Modified	<input type="checkbox"/>
<input type="checkbox"/>	Nachrichten	Article	Standard	German	15/09/2010 3:35 pm	<input type="checkbox"/>
<input type="checkbox"/>	News	Article	Standard	English (United Kingdom)	15/09/2010 3:31 pm	<input type="checkbox"/>

Remove selected Remove all

Figure 4.23:

The translations of the same object can be created and edited separately and simultaneously by multiple users (a user only edits one version and language at a time).

Adding a new translation

You can translate the objects into any of the translation languages using the administration interface. The following text reveals how you can translate an object using different approaches.

Using the "Sub items" window

1. Use the administration interface to navigate to the node/object which contains the one that you wish to edit. In other words, make sure that the parent node is being displayed.
2. Look at the "Sub items" window and locate the node/object that you wish to edit. Click on the node's corresponding edit icon (on the right hand side). You will be taken to the language selection interface. Select the desired parameters in the "New languages" frame and click the "Edit" button. The system will bring up the edit interface.

Using the tree menu and the context menu

1. Use the tree menu on the left to locate the object that you wish to edit.
2. Click on the object's icon in order to bring up the context menu.
3. Access the "Edit in" sub-menu and select the "Another language" item. You will be taken to the reduced language selection interface. It contains a list of languages in which the object does not exist (look at the following screenshot) and a list of languages that the new translation can be based on.

(see figure 4.24)

Edit <Article (main-page)>

New languages

Select the language you want to add:

French (France)

German

Select the language the added translation will be based on:

None

English (United Kingdom)

Edit **Cancel**

Figure 4.24:

Select the language that you wish to add and optionally one of the existing languages to be used as original text during translation. After the "Edit" button is clicked, the system will bring up the edit interface.

Using the "Edit" button

1. Use the administration interface to navigate to the object that you wish to edit. In other words, make sure that the object is being displayed.
2. Select the "Another language" item from the drop-down list of languages which is located in the preview window and click the "Edit" button. The system will display the reduced language selection interface (described above). Select the desired parameters and click the "Edit" button. The system will bring up the edit interface.

Using the bookmarks

1. Make sure that your bookmarks are being displayed (use the "+" sign to open the window).
2. Locate the object that you wish to edit and click on its icon to bring up the context menu.
3. Access the "Edit in" sub-menu and select the "Another language" item. The system will display the reduced language selection interface (described above). Select the desired parameters and click the "Edit" button. The system will bring up the edit interface.

4.4.8 The bit-field algorithm

The following text reveals some technical details related to the bit-field algorithm that is used for language filtering and prioritizing.

The system stores information about all the translation languages in the "ezcontent_language" database table. These languages are identified by powers of 2 i.e. their ID numbers are 2, 4, 8, 16, 32 etc. The value 1 (2^0) is used for marking the objects always available. When an object is marked always available, it will be shown even though it does not exist in a language that is specified using the prioritized language list (the "SiteLanguageList" configuration array).

The "always_available" field

The "ezcontentclass" table in the database includes the "always_available" field (0 by default) which controls whether new instances (objects) of a class should be set to "always available" or not. If this value is set to 1 for a class, then all newly created instances of that class will be always available. Note that this can be changed on the object level later on. The class setting only controls the initial value of the "always available" flag of the objects.

The "language_mask" field

When storing information about a content object in the "ezcontentobject" database table, the system uses a special bit-field called "language_mask" to identify languages in which the last published version of an object exist. This field contains the sum of the ID numbers of these languages plus 1 if an object is always available. When a new object is created, the sum of the initial language ID and the default "always_available" value (specified in the class) will be recorded to the object's "language_mask" field.

The "language_mask" bit-field is updated every time an object's language configuration changes. This typically happens when a translation is added or removed to/from an object.

Example

Let's say that you have two translation languages with the following ID numbers:

Language name	ID
English (United Kingdom)	2
French (France)	4

This allows the following possible values of "language_mask" for your content objects:

Language mask	Bitmap	Languages
2	00010	The object exists in English.
3	00011	The object exists in English

		and is always available.
4	00100	The object exists in French.
5	00101	The object exists in French and is always available.
6	00110	The object exists in English and French.
7	00111	The object exists in English and French and is always available.

When storing information about an object's version in the "ezcontentobject_version" database table, the "language_mask" bit-field contains the sum of the ID numbers of the languages in which a version exists plus 1.

The "initial_language_id" field

The "ezcontentobject" table in the database includes the "initial_language_id" field which is used for storing the ID number of the object's initial language.

When storing information about an object's version in the "ezcontentobject_version" database table, the system records the ID number of the language which the version was edited in to a special bit-field called "initial_language_id".

The "language_id" field

When storing information about an object attribute in the "ezcontentobject_attribute" database table, the system uses the "language_code" field to store the language code of the translation that the attribute belongs to (for example "eng-GB"). The "language_id" bit-field represents the same information in terms of language ID numbers i.e. this field contains the ID number of the translation language.

4.4.9 Language-based permissions

The "create" and "edit" functions of the "content" module support limitation on the language level. For example, it is possible to configure the system so that a group of users are allowed to create and translate objects using English and Norwegian while another group of users are only allowed to translate existing content into French. The "read" function does not support limitation on the language level and thus all translations of an object can be viewed by users who have read access to it.

Content/create

The "language" limitation of the "create" function controls which languages that are allowed to be used when objects are created. The following screenshot shows the edit interface for a policy that only allows the creation of French articles within the standard section.

(see figure 4.25)

Create a new policy for the <New role> role

Welcome to the policy wizard. This three-step wizard will help you set up a new policy. The policy will be added to the role that is currently being edited. The wizard can be aborted at any stage by using the "Cancel" button.

Step one: select module [completed]

Selected module:
Content

Selected access method:
Limited

Step two: select function [completed]

Selected function:
Create

Selected access method:
Limited

Step three: set function limitations

Instructions:

- Set the desired function limitations using the controls below.
- Click the "OK" button to finish the wizard. The policy will be added to the role that is currently being edited.

Properties

Class:	Section:	ParentOwner:	ParentGroup:	ParentClass:	ParentDepth:	Language:
Any	Any	Any	Any	Any	Any	Any
Article	Design	Self	Self	Article	1	English (United Kingdom)
Article (main-page)	Media	Self or anonymous users per HTTP session		Article (main-page)	2	French (France)
Article (sub-page)	Restricted			Article (sub-page)	3	German
Article with comment	Secret section			Article with comment	4	
Banner	Setup			Banner	5	
Blog	Standard			Blog	6	
Blog post	Users			Blog post	7	

Nodes (0)

The node list is empty.

Remove selected Add nodes

Figure 4.25:

Content/edit

The "language" limitation of the "edit" function controls which translations of objects that can be edited. It also controls which translations that can be added to objects. The following screenshot shows the edit interface of a policy that only allows editing French content (articles) or adding a French translation to existing articles.

(see figure 4.26)

Create a new policy for the <New role> role

Welcome to the policy wizard. This three-step wizard will help you set up a new policy. The policy will be added to the role that is currently being edited. The wizard can be aborted at any stage by using the "Cancel" button.

Step one: select module [completed]

Selected module:
Content

Selected access method:
Limited

Step two: select function [completed]

Selected function:
Edit

Selected access method:
Limited

Step three: set function limitations

Instructions:

1. Set the desired function limitations using the controls below.
2. Click the "OK" button to finish the wizard. The policy will be added to the role that is currently being edited.


Properties

Class:	Section:	Owner:	Group:	Language:
Any	Any	Any	Any	Any
Article	Design	Self	Self	English (United Kingdom)
Article (main-page)	Media	Self or anonymous users per HTTP session		French (France)
Article (sub-page)	Restricted			German
Article with comment	Secret section			
Banner	Setup			
Blog	Standard			
Blog post	Users			

Figure 4.26:

Combined with the "content/read" function (which does not support any language limitations), the policy used in the example above will provide a configuration that allows any article to be translated from any language to French. The combination is shown in the following screenshot. Note that users who only do translation work do not need to have access to the "create" function of the "content" module.

(see figure 4.27)

 **Edit <Translator> [Role]**

Name:

Policies

<input checked="" type="checkbox"/>	Module	Function	Limitations	<input checked="" type="checkbox"/>
<input type="checkbox"/>	content	edit	Class(Article) , Section(Standard) , Language(French (France))	<input checked="" type="checkbox"/>
<input type="checkbox"/>	content	create	Class(Article) , Section(Standard) , Language(French (France))	<input checked="" type="checkbox"/>

Figure 4.27:

4.5 Multi-language support for URL aliases

In eZ Publish 3.10, a new feature that makes it possible to use multilingual virtual URLs (also known as nice URLs or URL aliases) was introduced. This feature allows URL aliases to exist in several translation languages.

Auto-generated aliases

From 3.10, the automated virtual URL generation mechanism allows URL aliases to exist in several languages, depending on which languages the actual objects exist in. In other words, the URL aliases for nodes are now created in accordance with the existing translations of the objects encapsulated by the nodes. When a new translation is added to an object, the system will automatically generate a new set of URL aliases (based on the translations) for the node(s) that encapsulate that object.

A new field "URL alias name pattern" has been added to the class edit interface. It controls how the virtual URLs of the nodes will be generated when the objects (instances of the classes) are created.

It is not possible to create, edit or remove auto-generated aliases using the administration interface. They are updated automatically when objects are changed. The only way to change an auto-generated alias is to edit the object itself in the corresponding language.

URL history entries

When the name of an object is changed, the system will take care of changing the auto-generated URL aliases for the associated nodes. In addition, an internal redirection will be created, which will make sure that the old URL still works. In other words, instead of deleting the old URL alias from the database, the system will convert it into a *URL history entry*. The old virtual URL will continue to redirect until a new node is created that uses the same URL. In this case, the old virtual URL will be deleted.

Note that the internal redirection mechanism is intentionally hidden from the user. You cannot view or manage URL history entries using the administration interface.

Example

Let's say that a folder called "Computers" contains an article called "Monitors", which can be accessed at "http://www.example.com/Computers/Monitors". If you rename the folder to "Hardware", the new URLs for accessing the folder and the article will be "http://www.example.com/Hardware" and "http://www.example.com/Hardware/Monitors". If someone tries to use one of the old URLs, the system will automatically redirect the user to the corresponding new URL. If you then rename the article into "LCD", the following three URLs will redirect the user to "http://www.example.com/Hardware/LCD":

- `http://www.example.com/Computers/Monitors`

- `http://www.example.com/Computers/LCD`
- `http://www.example.com/Hardware/Monitors`

Manual/user-defined aliases

The following two types of virtual URLs can be managed using the administration interface (page 326):

- Global URL aliases
- Node URL aliases

The list of *global URL aliases* contains all user-defined virtual URLs, except from those that are created for destinations (system URLs) like `content/view/full/node_id`, where `node_id` is the ID number of a node. These are called *node URL aliases* and can be managed separately for individual nodes.

While global aliases always start from the root of the site, an alias of a node can either start from the parent node or from the root of the site. This is controlled by the "Relative to parent" flag.

A node URL alias applies only to the specific node that it references; in other words, a user-defined alias of a node does not apply to the URLs for the node's children. Refer to the example below for more information.

Example

Let's say that a folder called "Norway" contains two articles that can be accessed using the following URLs:

- `http://www.example.com/norway/oslo`
- `http://www.example.com/norway/bergen`

and you have created a new URL alias "Kingdom" for this folder. In this case, the "Norway" folder will be accessible through any of the following URLs:

- `http://www.example.com/norway`
- `http://www.example.com/kingdom`

However, the following two URLs will bring up a "Module not found" error:

- `http://www.example.com/kingdom/oslo`
- `http://www.example.com/kingdom/bergen`

Redirection of URL aliases and wildcards

In 4.0.0, it is possible to choose whether an alias will work as a "direct" or "forward" one ("direct" == the entered URL in the address bar of a browser stays the same, "forward" == the system will redirect to the original URL), but only for URL wildcards. This is controlled by the "Redirecting URL" checkbox when creating URL wildcards.

URL aliases also gained a new feature. You have now the possibility to choose if a URL alias should be direct or re-direct. Previously aliases have always re-directed (HTTP 301). Versions prior to 3.10 did not redirect URLs pointing to modules. With this new feature this behavior is back.

Find an example of redirection of URL aliases in the screenshot below:

(see figure 4.28)

Create new alias

URL alias name: Destination:

Language:

Alias should redirect to its destination
 With *Alias should redirect to its destination* checked eZ Publish will redirect to the destination using a HTTP 301 response. Un-check it and the URL will stay the same — no redirection will be performed.

Place alias on the site root
 The new alias be placed under [Home](#).

Figure 4.28:

Use the "Alias should redirect to its destination" check-box to redirect the alias.

Wildcard based URL forwarding

eZ Publish supports wildcard based URL forwarding. This means that you can create a URL alias that contains one or more asterisks (*) and the system will automatically replace the corresponding parts in URLs according to the destination URL specified. For example, you can create a URL alias called "pictures/*/*" and specify "media/images/{1}/{2}" as a destination. In this case, a URL like "http://www.example.com/pictures/home/photo/" will load "http://www.example.com/media/images/home/photo/". In other words, you will be able to use "pictures" instead of "media/images" in the URLs when accessing content nodes that are located two or more levels beneath the "media/images" node.

 It is possible to choose whether an alias will work as a "direct" or "forward" one. In the example above, a "direct" alias means that when someone enters "http://www.example.com/

pictures/home/photo/" in the address bar of a browser, the entered address will stay the same while the actual node will be displayed. If the alias is of the "forward" type, the system will redirect to "http://www.example.com/media/images/home/photo/" instead.

Wildcard URL aliases can be managed using the administration interface.

Availability

An alias is only available for a siteaccess if the language of the alias matches one of the site languages specified for this siteaccess. If a siteaccess is configured to display untranslated content, then aliases in all languages will be available.

If an object is always available, the URL aliases for the object's node assignments will be available for all siteaccesses. This is true for both auto-generated and user-defined aliases.

Aliases which are always available

Some global aliases always need to be available regardless of which languages that are configured for a siteaccess. Because of this, a new flag called "Include in other languages" has been introduced for global aliases. It makes it possible to individually control the availability of the different aliases.

Languages

Multilingual URL aliases do not control which languages the requested pages will be displayed in. When a virtual URL of a node is requested, the system will figure out the correct language based on the language configuration of the current siteaccess (refer to the example below).

Example

If you create an article called "Company" and translate it into French, there will be two auto-generated URL aliases called "Company" and "Compagnie".

Let's say that you have two public siteaccesses called "gb" and "fr" with the following language configuration:

Siteaccess "gb"	Siteaccess "fr"
[RegionalSettings] SiteLanguageList [] SiteLanguageList [] =eng-GB SiteLanguageList [] =fre-FR	[RegionalSettings] SiteLanguageList [] SiteLanguageList [] =fre-FR SiteLanguageList [] =eng-GB

As the table shows, the "gb" siteaccess is configured to use English as the most prioritized language and French as a second one. This means that both "Company" and "Compagnie" aliases

will work. The system will bring up the English version of the article when one of the following URLs is requested:

- `http://www.example.com/gb/Company`
- `http://www.example.com/gb/Compagnie`

Note that if you configure only one language (English) for this siteaccess, the French alias will not be available.

While the most prioritized language for the "fr" siteaccess is French, the second one is English. This means that both aliases will work and the corresponding URLs will bring up the French version:

- `http://www.example.com/fr/Company`
- `http://www.example.com/fr/Compagnie`

Character transformation

The multilingual URL aliases feature uses 3 types/methods of character transformation for URLs. Their usage is controlled by the "TransformationGroup" directive located in the [URLTranslator] section of an override for "site.ini". The following table reveals the available transformation methods.

Name	Description
urlalias_compat	This method supports lowercase Latin letters from "a" to "z", digits and underscores in URLs. It provides the same behavior as in eZ Publish 3.9.x and earlier versions. Capital characters are not preserved.
urlalias	This method supports more characters, but the URLs are still restricted to the ASCII table (with a few exceptions). Capital characters are preserved.
urlalias_iri	This method allows all Unicode characters in the URLs (with a few exceptions). It preserves the original text as much as possible, which results in more user-friendly URLs. Multiple whitespaces are converted to one. Capital characters are preserved. This is the recommended method for both single- and multi-language sites.

If you use the "urlalias_iri" type of transformation for URL aliases, be aware of the fact that some web browsers use [percent encoding](#) for Unicode characters in the URLs. For example, if a visitor enters a URL like "http://www.example.no/Ostehvel" in the address bar of a

browser, it might be automatically converted to "http://www.example.no/Osteh%C3%B8vel". However, this won't prevent eZ Publish from serving the requested web page. Users of Mozilla Firefox can disable this behavior by typing [about:config](#) in the browser's address bar to edit the "network.standard-url.escape-utf8" preference.

Refer to the following example to learn how the multilingual URL alias feature actually works.

Example

Let's say that we have the following site structure:

- Company (node ID: 10)
 -
 - About us (node ID: 11)
 - Contacts (node ID: 12)

If node 10 ("Company") gets translated into French, it will get the second alias "Compagnie". The structure of the site will look like this:

- Company|Compagnie (node ID: 10)
 -
 - About us (node ID: 11)
 - Contacts (node ID: 12)

At this point, node 10 can be accessed by using both aliases for all siteaccesses that have both English and French on the list of site languages. If a siteaccess is configured to use English as the most prioritized language, both aliases will bring up the English version. If the most prioritized language is French, both aliases will bring up the French version of the company page for this siteaccess.

The "About" page (node 11) can be accessed using either "Company/About" or "Compagnie/About" as the URL. The "Company/About" alias will work for any siteaccess that has English on the list of site languages. The "Compagnie/About" alias will only work for the siteaccesses that are configured to use both English and French languages. In both cases, it is the English translation that will be displayed (since the object only exists in English). If you edit the "About" page and enable the "Always available" flag, the page will be accessible through both aliases for all siteaccesses regardless of their language configuration (even if a siteaccess does not have English on the list of site languages).

If the "Contacts" page (node 12) is translated into German, it will get the second alias "Kontakten". In this case, the structure of the site will look like this:

- Company|Compagnie (node - id 10)
 -

- About us (node - id 11)
- Contacts|Kontakten (node - id 12)

At this point, it will be possible to access the page "Contacts|Kontakten" (node 12) by using one of the four URL aliases listed below. The following table shows which language configuration of a siteaccess is required for each alias to work.

URL alias	Site languages that must be enabled
"Company/Contacts"	English
"Compagnie/Contacts"	English and French
"Company/Kontakten"	English and German
"Compagnie/Kontakten"	French and German

4.5.1 Managing URL aliases

The administration interface makes it possible to easily manage the virtual URLs (page 153) that belong to a site. Management happens through the manipulation of two lists. While one of them is related to node URL aliases, the other deals with global aliases. In addition, it is possible to manage rules of wildcard based URL forwarding (so-called "wildcard aliases") using the URL wildcards interface.

Managing node aliases

The interface for managing the URL aliases of content nodes can be accessed by selecting the "Manage URL aliases" item from the "Advanced" section of the context sensitive menu for each content node. It can also be accessed by requesting "content/urlalias/<node_id>" directly (where "node_id" must be replaced with the actual ID number of the desired node). The following screenshot the interface for managing a node's URL aliases.

(see figure 4.29)

URL aliases for <eZ Publish> (3)

URL alias	Language	Type
<input type="checkbox"/> /eZ-Web-Page	English (United Kingdom)	Redirect
<input type="checkbox"/> /page-web	French (France)	Redirect
<input type="checkbox"/> /webseite	German	Redirect

Remove selected Remove all

Generated aliases (1)

Note that these entries are automatically generated from the name of the object. To change these names you must edit the object in the specific language and publish the changes.

URL alias	Language
/eZ-Publish	English (United Kingdom)

Create new alias

URL alias name:

Destination:

Language:

Alias should redirect to its destination

With *Alias should redirect to its destination* checked eZ Publish will redirect to the destination using a HTTP 301 response. Un-check it and the URL will stay the same — no redirection will be performed.

Place alias on the site root

The new alias be placed under [Documentation](#).

Create

Figure 4.29:

This interface provides an overview of all URL aliases that belong to the selected node. In addition, it can be used to create new and remove the existing aliases. The example shows a list of virtual URLs for the "eZ Publish" node. There are three manual aliases: "eZ-Web-Page", "page-web" and "webseite". While the "page-web" alias is associated with French language, the "webseite" alias is associated with German and "eZ-Web-Page" alias is in English. This means that the same page (node) can be accessed using any of these aliases if both French, German and English languages are configured for a siteaccess.

The drop-down list can be used to select the site language that the alias should be associated with. For example, if "Spanish" is selected, the alias will be available for all siteaccesses that have Spanish on the list of site languages. The drop-down list contains all the languages that are configured for the admin siteaccess. If the "ShowUntranslatedObjects" setting is enabled, then all translation languages will be listed. For example, it will be possible to create an alias associated with Spanish even though the actual object does not exist in this language. Note that multilingual aliases do not control which language the requested page will be displayed in (this depends on the language configuration of the current siteaccess).

The "Place alias on the site root" check-box can be used to specify where the alias/URL to be added should start. If checked, it will start from the root node. Otherwise, the provided alias is created from the parent node. For example, if you are adding a new URL alias called "test" to a node "City" located at "/country/state/city", the new URL alias will either be "/country/state/test" or just "/test" depending on whether the "Place alias on the site root" check-box was checked or unchecked.

The "Generated aliases" window

In the example above, the "eZ-Publish" node exists in both English, German and French languages and thus it has three auto-generated aliases "eZ-Web-Page", "page-web" and "webzeite". These are automatically created by the system based on the existing translations of the actual object. The list of auto-generated aliases is shown in the "Generated aliases" window located in the upper part of the interface. The "eZ-Publish" page can be accessed using any of these aliases if both French, German and English languages are configured for a siteaccess.

Note that the "Generated aliases" window displays only one URL for each language that the actual object exists in even though the parent node has several aliases in different languages. In other words, it can happen that not all the possible combinations are shown. For example, if a new node "Employees" is created beneath the "Company" node, it can be accessed using one of the following URL aliases:

- Company/Employees
- Compagnie/Employees
- articles/company_info/Employees
- MaCompagnie/Employees
- mycompany/Employees

However, only one of these URLs will be displayed in the list of auto-generated aliases for the "Employees" node. The system will pick one of the auto-generated aliases of the parent node (either "Company" or "Compagnie"), depending of the siteaccess' language configuration. If the most prioritized language for the admin siteaccess is English, only the "Company/Employees" alias will be displayed in the "Generated aliases" section for the "About" node. If the most prioritized language is French, then "Compagnie/Employees" will be displayed. The corresponding alias will be displayed using bold characters in the list of auto-generated aliases for the parent node. The screenshot above shows a situation when English is configured as the most prioritized language for the admin siteaccess and thus the "Company" alias is displayed using bold characters.

Creating a new node alias

To create a new alias, first select which site language that the alias should be associated with. Type in the desired text for the new alias into the input field and click the "Create" button. It is possible to create virtual URLs that make it look like if a node is situated at a different location in the tree. For example, you can create a URL alias "my_dummy_folder/my_article" for a node called "Article" which in reality is located inside a folder called "Articles". Note that in this case, the "Relative to parent" checkbox must be unchecked.

Further notes

Let's say that you have a node (somewhere in the tree, the location does not matter) called "About us". As previously mentioned, you can create an imaginary URL (with bogus / non-existing parent(s)) for it. For example, you can create "company/about_us" and it will work (the system will bring up the "About us" node). Assuming that the "Company" node does not exist from before, if someone requests only "company", the system will return an "Object not found" page. However, if a node called "Company" is created, the system will automatically make a URL alias for it (most likely "company") and thus the "company" alias will work (it will bring up the "Company" node).

Managing global aliases

The interface for managing global aliases was introduced a long time ago. However, it was changed in eZ Publish 3.10. This interface can be reached by clicking the "URL translator" link under the "Setup" tab in the Administration interface. The following image shows the URL translator interface.

(see figure 4.30)

As the screenshot shows, it is similar to the interface used to manipulate the URL aliases for individual nodes. The list displays all aliases in the system. The list is sorted by the text of the aliases (not the actual path that the alias is created for).

In the example above, three global aliases for the "search" view of the "content" module have been added. While the first alias is associated with the German language, the second one is in

Globally defined URL aliases (3)

10 25 50 100

<input checked="" type="checkbox"/>	URL alias	Destination	Language	Always available	Type
<input type="checkbox"/>	/Findet-mich	content/search	German	yes	Redirect
<input type="checkbox"/>	/findme	content/search	English (United Kingdom)	yes	Redirect
<input type="checkbox"/>	/trouve-moi	content/search	French (France)	yes	Redirect

[Remove selected](#) [Remove all](#)

Create new alias

New URL alias:

Destination (path to existing functionality or resource):

Language

Include in other languages

Alias should redirect to its destination
 With *Alias should redirect to its destination* checked eZ Publish will redirect to the destination using a HTTP 301 response. Un-check it and the URL will stay the same — no redirection will be performed.

[Create](#)

Figure 4.30:

English and the third one is in French. This makes it possible to access the "search" view using the "Findet-mich" "findme" or "trouve-moi" aliases if the current siteaccess is configured to use both German, English and French. In other words "http://www.example.com/content/search", "http://www.example.com/findme" and "http://www.example.com/trouve-moi" will bring up the search interface. The "Always available" column indicates whether the alias is always available (page 319) or not. In the screenshot above, the "findme" alias is always available (it will work for all siteaccesses regardless of their language configuration).

Note that unlike before 3.10, this list does no longer show aliases for nodes. The node aliases can be viewed and edited (individually, for every node) from within the node alias interface which was described earlier.

Creating a new global alias

To create a new global alias, input the desired virtual URL and the path to existing functionality or resource; this can be a module/view-combination or the address of a node. The specified alias will always start at the root of the site. The language drop-down list can be used to select which

site language the alias should be associated with. If the "Include in other languages" checkbox is selected, the newly created alias will work for all site accesses regardless of their language configuration.

Note that it is possible to use the global aliases interface to create aliases to nodes (to for example "content/view/full/<node_id>" or a virtual URL). However, such aliases will not appear in the global list. They will automatically appear in the node aliases interface for the corresponding node.

Managing wildcard URL aliases

The interface for managing wildcard aliases can be reached by clicking the "URL wildcards" link under the "Setup" tab in the administration interface. The following image shows how the URL wildcards interface looks like.

(see figure 4.31)

Defined URL aliases with wildcard(5)

10 25 50 100

<input checked="" type="checkbox"/> URL alias wildcard	Destination	Type
<input type="checkbox"/> ez_publish/user_manual/current/*	ez_publish/user_manual/4_0/{1}	Forward
<input type="checkbox"/> ez_publish/user_manual/current	ez_publish/user_manual/4_0	Forward
<input type="checkbox"/> ez_publish/technical_manual/current	ez_publish/technical_manual/4_0	Forward
<input type="checkbox"/> ez_publish/technical_manual/4_0/*	eZ-Publish/Technical-manual/4.x/{1}	Forward
<input type="checkbox"/> ez_publish/user_manual/4_0/*	eZ-Publish/User-manual/4.x/{1}	Forward

Remove selected Remove all

Create new URL forwarding with wildcard

New URL wildcard:

Destination:

Redirecting URL

Create

Figure 4.31:

The interface displays all wildcard aliases in the system and enables you to create new ones. In the example above, five wildcard aliases have been added.

When someone enters a URL in the browser address bar, the system will search for a node that matches this URL according to the specified rules of wildcard based URL-forwarding. In eZ Publish, the only wildcard character that can be used in aliases is asterisk (*). It can be repeated several times within one alias. Note that all wildcard characters within one alias are automatically

assigned numbers: 1, 2, 3, ... These are used when you specify a destination for this alias: {1}, {2}, {3}, ...

The "Type" column indicates whether the alias is a "direct" or "forward" one. In the screenshot above, all aliases are of the "forward" type, which means the system will redirect users to the original URL of the node that is being accessed (destination). In other words, when someone enters a URL like "http://www.example.com/pictures/home/photo/" in the address bar of a browser, the system will redirect the user to "http://www.example.com/media/images/home/photo/".

Creating wildcard aliases

To create a new wildcard alias, input the desired text of the alias and the destination address into the corresponding fields. If the "Redirecting URL" check-box is selected, the newly created alias will work as "forward" one. After you click the Create-button, the newly added alias will appear on the list.

4.5.2 URL transformation rules

When a site administrator enters a value for a new virtual URL, the system will perform cleanup of the input by using so-called URL transformation rules. This is done in order to avoid problems with certain characters and to ensure that the alias conforms the standards and the other URLs of the site. If an inputted alias is modified, the user will be notified.

Note that in eZ Publish 3.10, the transformation of entered/generated aliases has changed.

Unicode support

In versions prior to 3.10, URL transformation rules were more restrictive and only supported some ASCII characters (lowercase Latin letters from "a" to "z", digits and underscores). This caused problems for many non-western languages that use different alphabets, some of them which are difficult to transliterate.

From eZ Publish 3.10, it is possible to enable Unicode support for the URLs and thus no transliteration needs to be performed since most characters are allowed. The following characters are not allowed: ampersand, semi-colon, forward slash, colon, equal sign, question mark, square brackets, parenthesis and the plus sign. Note that spaces are only allowed as word separators. These characters are not allowed in order to avoid miscellaneous problems (related to the HTTP protocol).

The Unicode characters are encoded using the [IRI](#) standard. The text is encoded using [UTF-8](#) before further encoding is performed. The resulting URL will contain characters that are compatible with the HTTP protocol and which will work in all existing browsers/clients. Note that modern browsers will decode the URL and display the characters using Unicode.

Dash/underscore/space

In versions prior to 3.10, only underscores were allowed as separators of words. From 3.10, it is possible to choose which word separator that should be used. This can be done by changing the value of the "WordSeparator" configuration directive located in the [URLTranslator] section of an override for "site.ini". It can be set to either "dash", "underscore" or "space". Note that this setting will be ignored when the "urlalias_compat" transformation method is used (since it only supports underscores as separators).

Case sensitivity

When the "urlalias" or "urlalias_iri" transformation method is used, the URLs will consist of mixed cases (uppercase and lowercase characters). This is different from the traditional/old behavior where every letter was converted to lowercase. Instead, the system will preserve the cases and store the URL aliases accordingly. However, the URLs themselves will not be case sensitive. For example, the URL alias for a node called "About Us" will be "About-Us" (assuming that the word separator is a dash). The "About Us" node will be accessible regardless of how the URL is specified when it comes to lowercase and uppercase letters. In other words, the node will be accessible

through all of the following URLs: "www.example.com/about-us", "www.example.com/About-us", "www.example.com/ABOUT-US"; and so on.

Note that if there are two nodes with (almost) identical names within the same location (for example "My article" and "My Article" inside a folder called "News"), the system will generate unique URL aliases for newly introduced conflicting nodes by attaching numbers to their URL aliases. For example, if a node called "My article" already exists and "My Article" is created at the same location, the URL alias of the second ("My Article") node will be "My-Article2". If a third "MY Article" node is introduced, its URL alias will be "MY-Article3"; and so on.

Alias text filtering

Support for filtering was implemented in order to introduce more flexibility when it comes to the generation of the aliases. The filters are performed by the system on the URLs before the result is transformed to a valid alias. The filters can be created as extensions. The following text explains how to create a new filter.

- Open the "site.ini" override and add a new extension (f.ex. "myfilters") under the [URL-Translator] section.

```
Extensions []
Extensions []=myfilters
```

- Add a new filter in the "Filters[]" array (f.ex. "StripWords") under the [URLTranslator] section..

```
Extensions []
Extensions []=myfilters

Filters []
Filters []=StripWords
```

The system will search for the "stripwords.php" file containing the "StripWords" filter class.

- Create a file called "stripwords.php" located in the "extension/myfilters/urlfilters" directory. Note that all filters must be placed inside the "urlfilters" directory located within an extension's directory. Make sure that the newly created file contains the following lines:

```
<?php
class StripWords
{
    function process( $text, $languageObject, $caller )
    {
        return str_replace( "hell", "", $text );
    }
}
?>
```

The filter class "StripWords" implements a method called "process" which has three parameters: the text to filter, the language object (eZContentLanguage) and the object which called the filter process. The method returns a filtered version of the text. In this example, all occurrences of the word "hell" are removed (replaced with nothing). In other words, after this filter is introduced, newly created URLs will not contain the word "hell".

Refer to the "[URLTranslator]" section of the "site.ini" for more information about the "Filters" setting.

4.5.3 Custom transformation commands

In order to transform the URLs according to specific needs, it is possible to create and use so-called custom commands. The commands can be created as extensions and added to the system using an override of the "transform.ini" configuration file. The following text explains how to create a custom transformation command for URLs.

Let's say that for some reason, we would like all URLs to be reversed. This can be achieved by creating a custom transformation command.

1. Creating a new extension

A new file must be create and placed inside the directory of an extension. The file must contain a class that has a static method called "executeCommand" with three parameters: "\$text", "\$command" and "\$charsetName".

- \$text - The input text to transform
- \$command - The name of the command to execute, this can be used to keep multiple commands in one function
- \$charsetName - The name of the charset in use for \$text, usually not needed

In this example, we will create a "myreverse.php" file under the "extension/myextension/transformation" directory and put the following lines of code into it:

```
<?php
class MyReverse
{
    function executeCommand( $text, $command, $charsetName )
    {
        $text = strrev( $text );
        return $text;
    }
}
?>
```

As the code shows, the function will return a reversed version of the inputted text.

2. Registering a new transformation command in the "transform.ini" file

The command must be registered in the "transform.ini" configuration file. To do that, you need to add a new line into the "Commands[]" array located under the "[Extensions]" section. The line must contain the path to the PHP file, a colon (used for separation) and the class name that should be used. The following example demonstrates how this can be done.


```
[Extensions]
Commands []
Commands [my_reverse]=extension/myextension/transformation/
myreverse.php:MyReverse
```

3. Adding a new command to the corresponding transformation group

The newly created command must be added in the "transform.ini" file to one of the groups that will be used for URL text transformation. In the example below, the custom command "my_reverse" is added to the "urlalias_iri" transformation group.

```
[urlalias_iri]

Commands []
Commands []=url_cleanup_iri
Commands []=my_reverse
```

From now on, newly generated URLs will be reversed because they will be processed by the custom "my_reverse" command that we added to the list of commands to be performed when the "urlalias_iri" transformation method is used.

4.6 Clustering

The clustering feature makes it possible to run an eZ Publish site on several web servers. A site that is running on a cluster of servers will have better performance and will be able to handle more traffic.

Before eZ Publish clustering was implemented, the only way to support multiple servers was to store all cache files and images locally on separate file systems (one for each web server) and use "rsync" or NFS to synchronize caches & binary files. This was far from perfect, and induced many limitations. Instead, you can configure the system to store all content related caches, images and binary files in the database. This ensures that all the cluster nodes use the same cache files and have access to the same images and binary files. In other words, when content is updated, changes are automatically and instantly made available for every web servers in the cluster.

Supported database types

The clustering code is optimized for MySQL databases and requires the InnoDB storage engine. This storage engine will be used when creating the database tables needed for clustering. Contact your database administrator if you are unsure about whether InnoDB is available on your server.

Version 1.8 of the eZ Publish Extension for Oracle Database makes it possible to use Oracle as a database for eZ Publish version 4.0 and later and also includes support for the clustering functionality. Note that the clustering functionality provided by this extension may differ slightly from the generic implementation included in a standard eZ Publish distribution.

For now, eZ Publish does not support clustering for PostgreSQL databases. Also it is important to keep in mind that the supported databases depend on the cluster file handler that is used. For instance, MySQL is always supported, whereas Oracle is only supported for eZ DB File Handler. As the eZ Publish development moves forward more database handlers will of course be made available.

How it works

Data that must be synchronized between the different servers is stored using the database. However custom templates and design items will not be stored on the database. The following overview will give an overview of which data is saved where:

Data stored using the database includes:

- Binary files
- Image and image alias files
- Caches related to content:
 - Content view cache

Images and image aliases

The approach described above is also used when it comes to images and image aliases (image variations). However, the solution is a bit more complicated because images are usually served directly by the web server (for instance Apache). Since the web server isn't able to communicate with the database, the images need to be served using a PHP script called "index_image.php". This is true for all content images, but not for images that are related to design.

Note that you'll need to add new rewrite rules in order to instruct Apache to use "index_image.php" when serving images. This is explained in the chapters setting it up for an eZ DB FileHandler (page 345) and Setting it up for an eZ DFS FileHandler (page 351).

Notes about clearing the caches

Since eZ Publish 3.10 clearing the caches does not lead to the physical removal of cache files when using DB based handlers anymore (since this operation can be quite time consuming). The system will mark the cache files invalid instead of removing them physically from the database or file system. This can be done by either marking each particular cache file expired or setting the global expiry (the latter typically happens when a significant amount of changes is needed, e.g. when clearing all the caches of a specific type). The global expiry is a time-stamp that is used as an expiry value for all the caches in the system. If the global expiry is set to a certain date, all cache files that are older than this date will not be used. Note that the system will re-write old/expired cache file entries when re-creating the caches.

In order to physically remove the cache files from the database, the "ezcache.php" script needs to be run with the "--purge" option. The following example shows how to remove the content caches that are more than two days old:

```
$php bin/php/ezcache.php --clear-id=content --purge --expiry='-2 days'
```

Note that "\$php" should be replaced by the path to your php executable.

Extra connections in MySQL

The new clustering code available since eZ Publish 3.10 performs an extra connection when writing content to the database. (This connection checks whether the file has been modified since the write lock was acquired; if it has been modified, there is thus no longer a need to write.) Because of this, the maximum number of database connections in MySQL must be increased by 30-50%. If persistent connections are enabled, the cluster code will no longer share connections with normal database calls, so the maximum number of connections previously used will have to be doubled.

Oracle-specific differences

If you use the clustering functionality provided by the eZ Publish Extension for Oracle Database, note that the system may behave differently from what is described above. If all content related

caches are stored in an Oracle database, clearing the caches will always lead to their physical removal; the "ezcache.php" script will also physically remove the cache entries from the database, even when executed without the "--purge" option.

Cluster file handlers

The cluster file handler mechanism makes it possible to store, retrieve, rename, delete, etc. files using the database. The following file handlers are known to the system by default (click on the links for more information):

1. eZFS (located in the "kernel/classes/clusterfilehandlers/ezfsfilehandler.php" directory of the eZ Publish installation)
2. eZFS2 (located in the "kernel/private/classes/clusterfilehandlers/ezfs2filehandler.php" directory of the eZ Publish installation)
3. eZDB (located in the "kernel/classes/clusterfilehandlers/ezdbfilehandler.php" directory of the eZ Publish installation)
4. eZDFS (located in the "kernel/private/classes/clusterfilehandlers/ezdfsfilehandler.php" directory of the eZ Publish installation)

Note that eZFS and eZFS2 file handlers do not allow actual eZ publish clustering by using multiple servers. Use eZDB and eZDFS for cluster file handling.

Additional HTTP header

Since eZ Publish 3.9 an additional HTTP header called "Served-by" is supported. This feature was added for the purpose of testing and debugging. It is typically useful when you need to check, from the client side, which server handled the request. The following example shows a part of a server response that contains this header:

```
...
Last-Modified: Fri, 29 Jun 2007 09:35:54 GMT
Served-by: 62.70.12.230
Content-Language: en-GB
...
```

Limitation on some file systems when storing large number of content files

eZ Publish stores all disc related content (eg Images, PDF's etc) in var/storage like the structure from content tree, creating one folder for each object. In most file systems used under Linux (especially ext2 + ext3) there exists a hard LIMIT TO 32000 directories per folder. So it is not possible to store more as 31999 objects under one folder.

To get around this limitation without changing the file system, you can split your content tree so that you don't have more than 32k content files (example: images) in the same folder. Examples of file systems that supports more file/folder entries per folder.

- ReiserFS: roughly 1.2 million per directory
- ZFS: 2^{48} (a really big number: 281474976710656)!

Please refer to the requirements page (page [18](#)) regarding supported setups.

4.6.1 Cluster File Handlers

The cluster file handler mechanism makes it possible to store, retrieve, rename, delete, etc. files using the database. The following file handlers are known to the system by default:

1. eZFS (located in the "kernel/classes/clusterfilehandlers/ezfsfilehandler.php" directory of the eZ Publish installation)
2. eZFS2 (located in the "kernel/private/classes/clusterfilehandlers/ezfs2filehandler.php" directory of the eZ Publish installation)
3. eZDB (located in the "kernel/classes/clusterfilehandlers/ezdbfilehandler.php" directory of the eZ Publish installation)
4. eZDFS (located in the "kernel/private/classes/clusterfilehandlers/ezdfsfilehandler.php" directory of the eZ Publish installation)

Note that eZFS and eZFS2 file handlers do not allow actual eZ publish clustering by using multiple servers. Use eZDB and eZDFS for cluster file handling.

eZ FS File Handler

This is the default file handler which makes it possible to use the file system when dealing with files.

eZ FS2 File Handler

This is the enhanced standard file handler, with better concurrency handling. It requires linux or PHP 5.3 on windows, and is still considered experimental.

eZ DB File Handler

This is the database file handler. It makes it possible to use the database when dealing with files (in a cluster environment, this would typically be images, uploaded binary files and content-related caches, etc.). It is split into different back-ends that are compatible with the supported database engines. The default back-ends are located in the "kernel/classes/clusterfilehandlers/dbbackends" directory (currently only the back-end for MySQL).

Cache files are copied locally when used by a front-end. When using eZ DB File Handler both the metadata and the binary data will be stored using the database, but the metadata will be stored in the ezdbfile table and the binary data is split in chunks and will be stored in ezdbfile_data table.

Supported database

Currently supported databases for this file handler are MySQL and Oracle (when using the eZOracle extension).

eZ DFS File Handler

This is the Distributed File System handler with a DB overlay. This handler is required for NFS-based architectures. It clusters by storing the cluster files mainly on NFS (the distributed file system), while the file metadata (size, mtime, expiry status) are maintained in a database table similar to the one used by eZ DB file handler. NFS is used to read and write the reference copy of clustered files. Cache files are copied locally when used by a front-end, whereas images and binary files (when accessed directly via the browser) will be streamed directly from NFS.

Note: reverse proxies

High traffic on the binary files will not be handled well by the cluster database. In case of high traffic it is recommended to use Varnish or Squid.

Supported database

Currently only MySQL is supported as database for this file handler.

Specific eZ DFS global architecture configuration

The two most important aspects of the eZ DFS architecture are the cluster database and the NFS mount point. The first aspect implies that the database structure must be created manually. The definition of this table can be found in the eZ DFS MySQL driver class file located in the root of your eZ Publish installation here:

```
kernel/private/classes/clusterfilehandlers/dfsbackends/mysql.php
```

Since eZ DFS is based on NFS, each eZ Publish installation sharing the same relational database must use the same cluster database and each should have a local mount point to the same NFS export. The NFS server has to be available and writeable by the web-server's user on each eZ Publish server. Also it is recommended that each eZ Publish server is configured in the exact same way. Refer to your system and server manual on how to configure this for your system. It is required that each eZ Publish installation sets the NFS mount point in the global override of their settings/file.ini configuration file to the same location. This must be done in the configuration group "[eZDFSClusteringSettings]" setting "MountPointPath=". The NFS mount point is a local folder on each eZ Publish server that links to the network file system where the handler stores the files.

It is important to know that var directories should never be shared amongst instances, since they will then automatically be synchronized. This is valid for both eZ DB and eZ DFS, because it is the cluster handler that takes care of synchronizing data from and to the centralized repository. For more information visit the chapter "setting it up for a eZDFS file handler (page [351](#))".

4.6.2 Setting it up for an eZDBFileHandler

The following instructions reveal how you can configure eZ Publish to store images, binary files and content-related caches in the database using the eZ DB File Handler.

Before going any further, please read the note: known issue added at the end of this page regarding a known issue when using eZ Publish in a clustered environment.

1. Clear the caches (optional)

It is recommended (but not required) to clear all eZ Publish caches before enabling the clustering functionality. This can be done by running the following command from the root of your eZ Publish installation (if you are using multiple servers, run this command from each server):

```
$php bin/php/ezcache.php --clear-all --purge
```

”\$php” should be replaced by the path to your php executable.

After running the script, make sure that all cache files have been cleared by inspecting the contents of the various cache subdirectories within the ”var” directory (typically the ”var/cache/” and ”var/<name_of_siteaccess>/cache/” directories). If there are any cache files left, remove them manually.

2. Modify the ”file.ini” settings

Add the following lines to an override for the ”file.ini” configuration file (”settings/override/file.ini.append.php” or ”settings/siteaccess/ezwebin_site/file.ini.append.php” where ”ezwebin_site” is the name of your siteaccess):

```
[ClusteringSettings]
FileHandler=eZDBFileHandler
DBBackend=eZDBFileHandlerMysqliBackend
DBHost=localhost
DBPort=3306
DBName=name
DBUser=user
DBPassword=pass
```

First define the proper file handler, which in this case is ”FileHandler=eZDBFileHandler”. Specifying ”eZ DB File Handler” in the ”FileHandler” configuration setting will instruct eZ Publish to use the database specified here for storing images, binary files and content-related caches.

Next replace ”localhost”, ”name” (for example ”DBName=Cluster”), ”user” and ”pass” by actual host name, database name, user name and password. In most cases these values will be the same as ”Server”, ”Database”, ”User”, ”Password” settings specified under the [DatabaseSettings] block of your ”site.ini.append.php” configuration file.

The "DBBackend" setting specifies which back-end should be used by the eZ DB File Handler. For the database back-end setting (DBBackend=) the previously used value "mysql" is deprecated, so you need to use "eZDBFileHandlerMysqliBackend" for MySQL or "eZDBFileHandlerOracleBackend" for Oracle (this requires the eZ Oracle extension).

3. Create a new script for serving images

When clustering your installation all images (except design images) will be served by PHP. Your web-server (e.g Apache) will be instructed to use a specific PHP script called "index_cluster.php" for handling images, which will make the serving of images faster because the system does not have to read the configuration from the database. "index_cluster.php" doesn't exist by default so it must be created manually and must include "index_image.php" along with a collection of configuration settings.

Create the "index_cluster.php" inside the eZ Publish root directory and make sure that it contains the following lines. (The default contents of this file can also be found in the comments and example at the top of your installation's "index_image.php" file):

```
<?php
define( 'STORAGE_BACKEND',      'mysqli'      );
define( 'STORAGE_HOST',        'localhost'   );
define( 'STORAGE_PORT',        3306           );
define( 'STORAGE_SOCKET',      ''            );
define( 'STORAGE_USER',        'user'         );
define( 'STORAGE_PASS',        'pass'         );
define( 'STORAGE_DB',          'name'        );
define( 'STORAGE_CHUNK_SIZE',  65535          );
include_once( 'index_image.php' );
?>
```

Note: Make sure you specify the same database settings as indicated under the "[ClusteringSettings]" block in your "file.ini.append.php" configuration file.

With this script the inclusion sequence will work as follows. First the rewrite rules (to be added in step 7) will redirect the images requested to the newly created "index_cluster.php". The custom image file "http://site.com/var/storage/images/foo.jpg" will be transformed to "http://site.com/index_cluster.php/var/storage/images/foo.jpg". Because "index_cluster.php" contains the set of settings described below, the "index_image.php" file (which is part of eZ Publish distribution) will be included when "index_cluster.php" is executed. This index_image.php includes a file named "index_image_<STORAGE_BACKEND>.php". This means that if, for example, STORAGE_BACKEND is set to 'mysqli' in "index_cluster.php", the included file will be "index_image_mysqli.php". This "index_image_<STORAGE_BACKEND>.php" is the final script that will read and stream the files to the browser.

4. Create new database tables

The database table structure required to hold clustered file information needs to be created. This must be done manually, either on the same MySQL server or on the one used for the relational database or on a different one. Keep in mind that for large scale websites, a dedicated MySQL server will improve performances. The definitions of this table can be found inside the comment blocks in the beginning of the "mysqli.php" file located in the following sub-directory:

```
(...)/kernel/classes/clusterfilehandlers/dbbackends/mysqli.php
```

The MySQL table definitions when using eZ DB File Handler can also be imported directly from:

```
(...)/kernel/sql/mysqli/cluster_schema.sql
```

5. Import files to the database

The files stored in the "var" directory need to be copied to the database. To do this, go to the root directory of eZ Publish and run the following script (replace "ezwebin_site" by the actual name of your siteaccess):

```
$php bin/php/clusterize.php -s ezwebin_site
```

Note that "\$php" should be replaced by the path to your php executable.

The script will import your files, images and image aliases ([image variations](#)) that are stored under the "var" directory to the cluster database. eZ DB stores both metadata and data in tables, the metadata in ezdbfile, the binary data will however be split into chunks and stored in ezdbfile_data.

Keep in mind that this process might take some time, depending on the amount of files that need to be imported.

6. Compile the templates (optional)

Since all caches now are empty, you should re-compile the templates. Note that this step can be skipped and thus the templates will be compiled on-demand when the site is browsed. Go to the root directory of eZ Publish and run this command:

```
$php bin/php/eztc.php -s ezwebin_site
```

Note that "\$php" should be replaced by the path to your php executable.

Replace "ezwebin_site" by the actual name of your siteaccess. Repeat this step for all siteaccesses that are in use.

7. Update the Apache configuration

Apache needs to know which PHP script to use when serving images, in this case "index_cluster.php". The script simply fetches the images from the database and serves them. By adding the RewriteRules mentioned below every request for a content image or binary file will be rewritten to "index_cluster.php", which will then deliver the files directly through HTTP from the NFS server. These rules are the same for eZ DB and eZ DFS. So add the following rewrite rules to the ".htaccess" file before the other/existing rules:

```
RewriteRule ^/var/([^/]+)/?storage/images-versioned/.*/index_cluster.php [L]
RewriteRule ^/var/([^/]+)/?storage/images/.*/index_cluster.php [L]
RewriteRule ^/var/([^/]+)/?cache/public/(stylesheets|javascript) /
index_cluster.php [L]
```

If no ".htaccess" file is used, add the same rules above the existing rewrite rules for eZ Publish in your Apache configuration file because these rules need to be found before the standard eZ Publish rewrite rules.

8. Restart Apache and test the site

Restart the Apache web server. After it has been restarted, the system should be up and running in cluster mode. Verify that the site works correctly, content images are displayed and content binary files are accessible (open the site pages in a web browser, log in to the administration interface, try clicking around and so on).

If for example a page of your website does not work correctly because its images are not displayed, your rewrite rules or your "index_cluster.php" file might be configured incorrectly. To locate the error, load the image directly in the browser (by, for example, choosing "open image in a new tab"). If instead of the image "Module not found" is displayed, then your rewrite rules are not correctly configured. If a PHP error is shown, your "index_cluster.php" is most likely configured wrong.

To test and troubleshoot your website, it can be useful to have more debug information regarding the cluster. This is an optional configuration but to enable it, create an override of the debug.ini file and enable "kernel-clustering" in the [GeneralCondition] block like this:

```
[GeneralCondition]
(...)
kernel-clustering=enabled
(...)
```

9. Remove the imported files from the file system

If the site works correctly, you can remove the original content images and binary files from the file system (since they have been successfully imported to the database). To do this, you need to inspect the contents of the various storage sub-directories within the "var" directory (typically the

"var/storage/" and "var/<name_of_siteaccess>/storage/" directories). If there are any content images and binary files left, remove them manually or by using the following command from the root of your eZ Publish installation:

```
$php bin/php/ezcache.php --clear-all
```

Note that "\$php" should be replaced by the path to your php executable.

If you configured multiple servers, execute the command from each server.

Note

The "clusterize.php" file mentioned in step "5. Import files to database" can also be used with a "-r" option. This will automatically remove the imported files after they have been clustered. Using it will make this step "9. Remove the imported files from the file system" obsolete. But keep in mind that using the "-r" option is some what advanced so use with caution.

Note: known issue

When using a database based file handler (eZ DB or eZ DFS) the following bug will occur if all of the conditions listed here are true:

- You use MySQL
- You use different databases for the content and cluster tables
- You use the same host, port, user name and password for both databases
- The port is explicitly specified in both site.ini and file.ini.

The bug is that eZ Publish will look for content tables in the cluster database, which means that all page requests will fail. Although a solution has been proposed, it has not yet been approved at the time of this writing. So for the moment the quickest workaround is to use different user names for the two databases.

For more information regarding this issue, please visit <http://issues.ez.no/13927>

Limitation on some file systems when storing large number of content files

eZ Publish stores all disc related content (eg Images, PDF's etc) in var/storage like the structure from content tree, creating one folder for each object. In most file systems used under Linux (especially ext2 + ext3) there exists a hard LIMIT TO 32000 directories per folder. So it is not possible to store more as 31999 objects under one folder.

To get around this limitation without changing the file system, you can split your content tree so that you don't have more than 32k content files (example: images) in the same folder.

Examples of file systems that supports more file/folder entries per folder.

- ReiserFS: roughly 1.2 million per directory
- ZFS: 2^{48} (a really big number: 281474976710656)!

4.6.3 Setting it up for an eZDFSFileHandler

The following instructions reveal how you can configure eZ Publish to store images, binary files and content-related caches in the database when using a eZ DFS File Handler. Before going any further, please read the note: known issue added at the end of this page regarding a known issue when using eZ Publish in a clustered environment.

1. Clear the caches (optional)

It is recommended (but not required) to clear all eZ Publish caches before enabling the clustering functionality. This can be done by running the following command from the root of your eZ Publish installation (if you are using multiple servers, run this command from each server):

```
$php bin/php/ezcache.php --clear-all --purge
```

”\$php” should be replaced by the path to your php executable.

After running the script, make sure that all cache files have been cleared by inspecting the contents of the various cache sub-directories within the ”var” directory (typically the ”var/cache/” and ”var/<name_of_siteaccess>/cache/” directories). If there are any cache files left, remove them manually.

2. Modify the ”file.ini” settings

Add the following lines to an override for the ”file.ini” configuration file (”settings/override/file.ini.append.php” or ”settings/siteaccess/ezwebin_site/file.ini.append.php” where ”ezwebin_site” is the name of your siteaccess):

```
[ClusteringSettings]
FileHandler=eZDFSFileHandler
```

First define the proper file handler, which here is ”FileHandler=eZDFSFileHandler”.

When using eZDFSFileHandler configure the settings in the [eZDFSClusteringSettings] block in the same override file. It is necessary to define (”var/nfsmount” is just an example) the path to the NFS mount point (a local folder) and set the database back-end setting to ”eZDFSFileHandlerMySQLBackend” (MySQL is the only database supported for eZDFS) as shown here:

```
[eZDFSClusteringSettings]
MountPointPath=var/nfsmount
DBBackend=eZDFSFileHandlerMySQLBackend
DBHost=dbhost
DBPort=3306
DBSocket=
DBName=cluster
```



```
DBUser=root
DBPassword=
DBConnectRetries=3
DBExecuteRetries=20
```

Replace "dbhost", "name" (for example "DBName=Cluster"), "user" and "pass" by actual host name, database name, user name and password. In most cases these values will be the same as "Server", "Database", "User", "Password" settings specified under the [\[DatabaseSettings\]](#) block of your "site.ini.append.php" configuration file.

3. Create a new script for serving images

When clustering your installation all images (except design images) will be served by PHP. Your web-server (e.g Apache) will be instructed to use a specific PHP script called "index_cluster.php" for handling images, which will make the serving of images faster because the system does not have to read the configuration from the database. "index_cluster.php" doesn't exist by default so it must be created manually and must include "index_image.php" along with a collection of configuration settings.

Create the "index_cluster.php" inside the eZ Publish root directory and make sure that it contains the following lines. (The default contents of this file can also be found in the comments and example at the top of your installations "index_image.php" file):

```
<?php
define( 'STORAGE_BACKEND',          'dfsmysqli'      );
define( 'STORAGE_HOST',             'localhost'     );
define( 'STORAGE_PORT',             3306            );
define( 'STORAGE_SOCKET',           false           );
define( 'STORAGE_USER',             'user'          );
define( 'STORAGE_PASS',             'pass'          );
define( 'STORAGE_DB',               'name'          );
define( 'MOUNT_POINT_PATH',         'var/nfsmount' );
include_once( 'index_image.php' );
?>
```

Note: Make sure you specify the same database settings as indicated under the "[ClusteringSettings]" block in your "file.ini.append.php" configuration file.

The location 'var/nfsmount' is just an example, you can set another location if you wish for example /mnt/nfs.

With this script the inclusion sequence will work as follows. First the rewrite rules (to be added in step 7) will redirect the images requested to the newly created "index_cluster.php". The custom image file "http://site.com/var/storage/images/foo.jpg" will be transformed to "http://site.com/index_cluster.php/var/storage/images/foo.jpg". Because "index_cluster.php" contains the set of settings described below, the "index_image.php" file (which is part of eZ Publish

distribution) will be included when "index_cluster.php" is executed. This index_image.php includes a file named "index_image_<STORAGE_BACKEND>.php". This means that if, for example, STORAGE_BACKEND is set to 'mysql' in "index_cluster.php", the included file will be "index_image_mysql.php". This "index_image_<STORAGE_BACKEND>.php" is the final script that will read and stream the files to the browser.

4. Create new database tables

The database table structure required to hold clustered file information needs to be created. This must be done manually, either on the same MySQL server or on the one used for the relational database or on a different one. Keep in mind that for large scale websites, a dedicated MySQL server can improve performance. The definitions of this table can be found inside the comment blocks in the beginning of the "mysql.php" file located in the following sub-directory:

```
(...)/kernel/private/classes/clusterfilehandlers/dfsbackends/mysql.php
```

5. Import files to the cluster

You need to copy the files stored in the "var" directory to the cluster. To do this, go to the root directory of eZ Publish and run the following script (replace "ezwebin_site" by the actual name of your siteaccess):

```
$php bin/php/clusterize.php -s ezwebin_site
```

Note that "\$php" should be replaced by the path to your php executable.

The meta data will be stored on the database, whereas the files themselves are copied to the configured NFS mount point using a structure exactly similar as that of the "var" directory.

Keep in mind that this process might take some time, depending on the amount of files that need to be imported.

Note: Moving files to NFS must always be done with clusterize.php, as doing it manually will produce an incomplete/invalid content repository, and will make it impossible to clusterize files.

6. Compile the templates (optional)

Since all caches now are empty, you should re-compile the templates. Note that this step can be skipped and thus the templates will be compiled on-demand when the site is browsed. Go to the root directory of eZ Publish and run this command:

```
$php bin/php/eztc.php -s ezwebin_site
```

Note that "\$php" should be replaced by the path to your php executable.

Replace "ezwebin_site" by the actual name of your siteaccess. Repeat this step for all siteaccesses that are in use.

7. Update the Apache configuration

Apache needs to know which PHP script to use when serving images, in this case `index_cluster.php`. The script simply fetches the images from the database and serves them. By adding the RewriteRules mentioned below every request for a content image or binary file will be rewritten to `index_cluster.php`, which will then deliver the files directly through HTTP from the NFS server. These rules are the same for eZDFS and eZDB. So add the following rewrite rules to the ".htaccess" file before the other/existing rules:

```
RewriteRule ^/var/([^/]+)/?storage/images-versioned/.*/index_cluster.php [L]
RewriteRule ^/var/([^/]+)/?storage/images/.*/index_cluster.php [L]
RewriteRule ^/var/([^/]+)/?cache/public/(stylesheets|javascript) /
index_cluster.php [L]
RewriteRule ^/index_cluster.php - [L]
```

If no ".htaccess" file is used, add the same rules above the existing rewrite rules for eZ Publish in your Apache configuration file because these rules need to be found before the standard eZ Publish rewrite

8. Restart Apache and test the site

Restart the Apache web server. After it has been restarted, the system should be up and running in cluster mode. Verify that the site works correctly, content images are displayed and content binary files are accessible (open the site pages in a web browser, log in to the administration interface, try clicking around and so on).

If for example a page of your website does not work correctly because its images are not displayed, your rewrite rules or your "index_cluster.php" file might be configured incorrectly. To locate the error, load the image directly in the browser (by, for example, choosing "open image in a new tab"). If instead of the image "Module not found" is displayed, then your rewrite rules are not correctly configured. If a PHP error is shown, your "index_cluster.php" is most likely configured wrong.

To test and troubleshoot your website, it can be useful to have more debug information regarding the cluster. This is an optional configuration but to enable it, create an override of the `debug.ini` file and enable "kernel-clustering" in the [GeneralCondition] block like this:

```
[GeneralCondition]
(...)
kernel-clustering=enabled
(...)
```

9. Remove the imported files from the file system

If the site works correctly, you can remove the original content images and binary files from the file system (since they have been successfully imported to the database). To do this, you need to

inspect the contents of the various storage sub-directories within the "var" directory (typically the "var/storage/" and "var/<name_of_siteaccess>/storage/" directories). If there are any content images and binary files left, remove them manually or by using the following command from the root of your eZ Publish installation:

```
$php bin/php/ezcache.php --clear-all
```

Note that "\$php" should be replaced by the path to your php executable. If you configure multiple servers, execute the command from each server.

Note

The "clusterize.php" file mentioned in step "5. Import files to database" can also be used with a "-r" option. This will automatically remove the imported files after they have been clustered. Using it will make this step "9. Remove the imported files from the file system" obsolete. But keep in mind that using the "-r" option is some what advanced so use with caution.

Note: known issue

When using a database based file handler (eZ DB or eZ DFS) the following bug will occur if all of the conditions listed here are true:

- You use MySQL
- You use different databases for the content and cluster tables
- You use the same host, port, user name and password for both databases
- The port is explicitly specified in both site.ini and file.ini.

The bug is that eZ Publish will look for content tables in the cluster database, which means that all page requests will fail. Although a solution has been proposed, it has not yet been approved at the time of this writing. So for the moment the quickest workaround is to use different user names for the two databases.

For more information regarding this issue, please visit <http://issues.ez.no/13927>

Limitation on some file systems when storing large number of content files

eZ Publish stores all disc related content (eg Images, PDF's etc) in var/storage like the structure from content tree, creating one folder for each object. In most file systems used under Linux (especially ext2 + ext3) there exists a hard LIMIT TO 32000 directories per folder. So it is not possible to store more as 31999 objects under one folder.

To get around this limitation without changing the file system, you can split your content tree so that you don't have more than 32k content files (example: images) in the same folder.

Examples of file systems that supports more file/folder entries per folder.

- ReiserFS: roughly 1.2 million per directory
- ZFS: 2^{48} (a really big number: 281474976710656)!

4.6.4 Reverting a cluster setup

You can always switch from using the clustering (page 337) functionality to the standard configuration where images, binary files and content-related caches are stored on a file system. The following instructions explain how this can be done, assuming that your eZ Publish site is currently configured to store images and other files in a MySQL database.

Reverting to non cluster setup

1. Clear the caches (optional)

It is recommended (but not required) to clear all eZ Publish caches before disabling the clustering functionality. This can be done by running the "bin/php/ezcache.php" script as shown below:

```
$php bin/php/ezcache.php --clear-all --purge
```

Note that "\$php" should be replaced by the path to your php executable.

2. Unclusterize images and other files

You need to copy images, binary files and content-related caches from the database to the file system. This can be done by running the "bin/php/clusterize.php" script with the "-u" parameter. Make sure that you have enough disk space on the file system before starting this operation.

The following example shows how to run the script (replace "ezwebin_site" with the actual name of your siteaccess):

```
$php bin/php/clusterize.php -u -s ezwebin_site
```

Note that "\$php" should be replaced by the path to your php executable.

The script will go through the files stored in the database and copy them into the appropriate subdirectories of the "var" directory.

3. Modify the "file.ini" settings

Open the "settings/override/file.ini.append.php" configuration file and edit it. Comment all lines located under the [ClusteringSettings] block by adding "#" as shown below:

```
[ClusteringSettings]
#FileHandler=eZDBFileHandler
#DBBackend=eZDBFileHandlerMysqliBackend
#DBHost=localhost
#DBPort=3306
```

```
#DBSocket=
#DBName=name
#DBUser=user
#DBPassword=pass
#DBChunkSize=65535
```

If the file does not exist in the "settings/override" directory, update the "settings/siteaccess/ezwebin_site/file.ini.append.php" file instead (where "ezwebin_site" is the name of your siteaccess).

Note that you can also remove the entire [ClusteringSettings] block instead of commenting the lines one by one.

By commenting out the entire entry you will revert to the default setting in "settings/file.ini" where the default FileHandler is eZ FS File Handler.

However if you prefer not to add #'s or delete the entry you can also set the FileHandler setting to eZFSFileHandler for the same result.

```
[ClusteringSettings]

FileHandler=eZFSFileHandler
```

4. (Optional) Disable the script for serving images

Rename or remove the "index_cluster.php" file located in the root directory of your eZ Publish installation. Until now, this script was used to serve images stored in your MySQL database. This step is not required if you properly perform "step 5" and remove the cluster rewrite rules, because once the rewrite rules are removed Apache will no longer use the script anyway.

5. Update the Apache configuration

Comment out or remove the following lines in your ".htaccess" file:

```
RewriteRule ^/var/([^/]+)/?storage/images-versioned/.*/index_cluster.php [L]
RewriteRule ^/var/([^/]+)/?storage/images/.*/index_cluster.php [L]
RewriteRule ^/var/([^/]+)/?cache/public/(stylesheets|javascript) /
index_cluster.php [L]
```

If no ".htaccess" file is used, you need to comment out the same lines in the Apache configuration file.

6. Restart Apache and test the site

Restart the Apache web server. After it has been restarted, your site should be using images, binary files and content-related caches stored in the "var" directory. Verify that the site works

correctly, content images are displayed and content binary files are accessible (open the site pages in a web browser, log in to the administration interface, try clicking around and so on).

7. Remove the clustering tables from the database

If the site works correctly, you can remove the database tables that were previously used for storing images, binary files and content-related caches. This can be done by executing the following SQL queries:

- With eZDB
DROP TABLE ezdbfile; DROP TABLE ezdbfile_data;
- With eZDFS
DROP TABLE ezdfsfile;
Note that the files on the NFS share need to be removed manually.

Changing to another Cluster file handler

1. Revert to non cluster mode

When changing from one cluster file handler to another it is best to first un-cluster by following the steps described in "Reverting to non cluster setup" before changing to another cluster file handler.

2. Setup cluster

Follow the steps described in "Setting it up for an eZ DB FileHandler (page 345)" or "Setting it up for an eZ DFS FileHandler (page 351)" to cluster your installation with the new cluster file handler.

4.6.5 Maintenance

When a new version of an object is created when using eZ Publish 4.2 and 4.3, the images of the first version are put into storage. But when the first version is deleted, the images are not removed completely, but remain in storage. This implies that these images are still accessible with their URL and that they have been indexed by Google. To prevent for example copyright issues, it is recommended that these images are deleted as well. Therefore it is necessary to clear expired binary items from your database cluster installations (such as eZ DB and eZ DFS). This can be done with one of two ways, either by running a script or a cronjob, depending on how regular you wish or need to delete expired images.

Use the first method for occasional maintenance. You can for example use this after upgrading to eZ Publish 4.3 in order to clean up your existing setup. Run the following script (how to do this depends on your system):

```
bin/php/clusterbinarypurge.php
```

If needed you can use the following options (Use --help to view all possible options):

- ` dry-run`: to display deleted files, without deleting anything (only applies to the bin script)
- ` iteration-sleep`: to pause time, in seconds, between each batch of delete operations. The default value is 1 second. Can be used as a floating point number.
- ` iteration-limit`: to set how many files should be deleted by SQL fetch. The default value is 100, but note that increasing this value will increase the memory footprint.
- ` memory-monitoring`: to monitor memory usage in `var/log/clusterbinarypurge.log`.

The second way is by means of a cronjob and should be used for regular maintenance. The necessary cronjob is located here:

```
cronjobs/clusterbinarypurge.php
```

This cronjob is by default part of a cronjob named `cluster_maintenance`. Run it with the following script from the root of the eZ Publish directory:

```
$php runcronjobs.php cluster_maintenance -s ezwebin_site
```

Note that `"$php"` should be replaced by the path to your php executable. Also how to run a script can differ depending on your system. Please consult your systems manual for more information.

It is recommended to run this cronjob weekly, although it can be done more frequently for websites where binary files are removed frequently. For more information please visit our [Issue Tracker: issue 015793](#).

Clean a DFS cluster install from stale DB files

A new script has been added for the purpose of cleaning a DFS cluster install from stale files in DB that do not exist on NFS and the opposite too. Run the following script to clean stale DB files:

```
bin/php/dfscleanup.php
```

Options:

- -S Check files on DFS share against files in the database
- -B Checks files in database against files on DFS share
- -D Delete nonexistent files

It is recommended to first run it WITHOUT the -D switch to see which files are going to be removed. For more information please visit our Issue Tracker: [#018508](#)

4.6.6 Cluster Configuration Settings

The cluster settings are located in an override of your "settings/file.ini" file (most likely "settings/override/file.ini.append.php" or "settings/siteaccess/<SITE_ACCESS>/file.ini.append.php")

When using the eZDB file handler the settings must be configured in the [ClusteringSettings] block. When using the eZDFS file handler the settings in the [eZDFSClusteringSettings] block must also be configured along with setting "FileHandler=eZDFSFileHandler" in [ClusterSettings] in the same file.

Note: It is recommended to use a distinct database server if you are clustering for a high traffic website

[ClusteringSettings]

FileHandler

Possible configuration settings are:

- FileHandler=eZFSFileHandler
- FileHandler=eZDBFileHandler
- FileHandler=eZDFSFileHandler

This setting sets the cluster file handler. For more information regarding the cluster file handlers visit chapter "cluster file handlers (page 342)". Since eZ Publish 4.1 the names of the file handlers have changed. For instance 'ezdb' is no longer recognised, the correct setting here would be 'eZDBFileHandler'. When using an eZ DFS File Handler please refer to the settings in [eZDFSClusteringSettings].

DBBackend

Possible configuration settings are:

- DBBackend=eZDBFileHandlerMysqlBackend
- DBBackend=eZDBFileHandlerOracleBackend

Use this setting to define the database backend used by the eZDB file handler. When using an eZDFSFileHandler please refer to the settings in [eZDFSClusteringSettings].

For the database backend setting (DBBackend=) the previously used value "mysql" is deprecated. Possible values are "eZDBFileHandlerMysqlBackend" when using MySQL or "eZDBFileHandlerOracleBackend" when using Oracle (this will require the ezoracle extension)

DBHost

Possible configuration setting is:

- DBHost=<string>

Use the actual host name as value. In most cases this value will be the same as the "Server" setting specified under the [\[DatabaseSettings\]](#) block of your "site.ini.append.php" configuration file. When using an eZ DFS File Handler please refer to the settings in [\[eZDFSClusteringSettings\]](#).

Note: It is recommended to use a distinct database server if you are clustering for a high traffic website.

Default value is:

```
DBHost=localhost
```

DBPort

Possible configuration setting is:

- DBPort=<integer>

This setting sets which port to use. When using an eZ DFS File Handler please refer to the settings in [\[eZDFSClusteringSettings\]](#).

Default value is:

```
DBPort=3306
```

DBSocket

Possible configuration setting:

- DBSocket=

Make sure that the "DBSocket" setting is correct. Leave it empty if you have "Socket=disabled" under the [\[DatabaseSettings\]](#) block in "site.ini.append.php". When using and eZDFSFileHandler please refer to the settings in [\[eZDFSClusteringSettings\]](#).

Default value is:

```
DBSocket=
```

DBName

Possible configuration setting:

- DBName=<string>

Use the actual database name as value. In most cases this value will be the same as the "Database" setting specified under the [\[DatabaseSettings\]](#) block of your "site.ini.append.php" configuration file. When using an eZ DFS File Handler please refer to the settings in [\[eZDFSClusteringSettings\]](#).

Default value is:

```
DBName=cluster
```

DBUser

Possible configuration setting:

- DBUser=<string>

Use the user name as value. In most cases this value will be the same as the "User" setting specified under the [\[DatabaseSettings\]](#) block of your "site.ini.append.php" configuration file. When using an eZ DFS File Handler please refer to the settings in [\[eZDFSClusteringSettings\]](#).

Default value is:

```
DBUser=root
```

DBPassword

Possible configuration setting:

- DBPassword=<string>

Use the actual password as value. In most cases this value will be the same as the "Password" setting specified under the [\[DatabaseSettings\]](#) block of your "site.ini.append.php" configuration file. When using an eZ DFS File Handler please refer to the settings in [\[eZDFSClusteringSettings\]](#).

Default value is:

```
DBPassword=
```

DBChunkSize

Possible configuration setting:

- DBChunkSize=<integer>

This setting determines the size of the blocks (in bytes) into which files are split when they are fetched and inserted from the database. This setting is only valid when using eZDB.

Default value is:

```
DBChunkSize=65535
```

DBConnectRetries

Possible configuration setting:

- DBConnectRetries=<integer>

When using an eZ DFS File Handler please refer to the settings in [eZDFSClusteringSettings].

Default value is:

```
DBConnectRetries=3
```

DBExecuteRetries

Possible configuration setting:

- DBExecuteRetries=<integer>

When using an eZ DFS File Handler please refer to the settings in [eZDFSClusteringSettings].

Default value is:

```
DBExecuteRetries=20
```

NonExistantStaleCacheHandling

Possible values are:

- NonExistantStaleCacheHandling[]=wait
- NonExistantStaleCacheHandling[]=generate

Possible key settings are:

- [viewcache]
- [cacheblock]
- [misc]

This is an advanced setting and should only be customized if it is specifically required. This setting defines what happens when a requested cache file is already being generated and no expired cache file exists (for instance if the content is new).

There are two possible values:

- The default value is "wait", which places the process in a wait loop for a limited time until the file is done generating.
- The second value is "generate" which lets the requesting process generate its own data without storing the result.

Use the [key] to define the type of cache that is impacted by the setting. The following three cache types are allowed as key:

- viewcache
- cacheblock
- misc: this is used for any cache that is not viewcache nor cacheblock.

Example configuration setting is:

```
NonExistantStaleCacheHandling[viewcache]=wait
```

[eZDFSClusteringSettings]

Specific configuration settings when using the eZ DFS File Handler. Most are similar to the ones mentioned in [ClusteringSettings], except "MountPointPath", "DBBackend" and "DBSocket".

MountPointPath

Possible configuration setting:

- MountPointPath= <string>

Since the eZ DFS File Handler is based on NFS, each eZ Publish installation sharing the same relational database must use the same cluster database and each should have a local mount point to the same NFS export. The location of the local mount point must be set here.

Example setting is:

```
MountPointPath=var/nfsmount
```

DBBackend

Possible configuration setting:

- DBBackend=eZDFSFileHandlerMySQLBackend

Use this setting to define the database back-end used by the eZ DB File Handler. Currently only MySQL is supported as database for the eZDFS file handler.

Default value is:

```
DBBackend=eZDFSFileHandlerMySQLBackend
```

DBSocket

Possible configuration setting:

- DBSocket=False

Default and value is:

```
DBSocket=False
```


4.7 Packages

From 3.8, the standard packages are not included in the eZ Publish distribution itself. They are distributed separately as ".ezpkg" files. The files can be downloaded automatically by the setup wizard from the remote repository or manually from [packages download page](#).

A *package* is a collection of items grouped together and stored in the specific format for the purpose of easy installation and removal. The system makes it possible to create packages and export them to ".ezpkg" files. This is the common way of how the packages are distributed. When an ".ezpkg" file is imported, it will become available under the system repository part of the eZ Publish installation. Most of the packages can be installed and uninstalled. The following table reveals the complete list of package operations that can be supported in the administration interface.

Operation	Description
Create new package	It is possible to export your class definitions, content objects, settings, design styles etc. by creating new packages of different types. The newly created packages will be stored under the "Local" system repository.
Import new package	To import a new package, you need to select the desired ".ezpkg" file locally. The system will then upload the file, unpack it and place the resulting package under an appropriate internal repository within the installation.
Remove selected	You can remove packages from the system repository. Please note that the package itself will not be removed. Only the package files will be removed from the internal repository.
Install	It is possible to install packages that are located under internal repositories. When you install a package, the system will create content classes and content objects, apply the settings specified in it and so on. (Please note that installing site packages and design packages is not supported.)
Export to file	A package located under the system repository can be exported to ".ezpkg" file. The system will ask you where to store the newly created file.
Uninstall	When you uninstall a package, all the changes made during its installation will be reverted.

Remote repository

Packages from *remote repository* can be downloaded by the setup wizard during the system installation process. However, the administration interface is unable to download packages from this repository.

The system will use the [eZ Systems packages repository](#) as the default remote repository. If you wish to use another remote repository, you need to specify its corresponding address using the "RemotePackagesIndexURL" configuration setting located in the "[RepositorySettings]" section of the "settings/override/package.ini.append.php" file.

System / internal repository

The default behavior is that all packages are stored in the "var/storage/packages" directory. This directory is the *main system repository* and its sub-directories are called "system repositories" or "internal repositories". The name of a sub-directory also functions as the actual name of a repository. The packages are sorted by their vendor. For example, the packages downloaded from [ez.no](#) will be stored under the "ez_systems" internal repository (var/storage/packages/ez_systems). Packages that have no vendor and packages created locally will reside under the "Local" repository (var/storage/packages/local).

It is possible to choose another location of the main system repository inside the "var/storage" directory. The following example shows how to change "settings/override/package.ini.append.php" in order to force the system to use "var/storage/importedpackages" as the main system repository:

```
RepositoryDirectory=importedpackages
```

4.7.1 Package types

The following package types are supported:

- content class packages
- content object packages
- extension packages
- site style packages (design packages)
- site packages

Content class and content object packages

A *content class package* allows the storage of class definitions. If you create several classes and need to use them on other installations, you can export these class definitions into a content class package. The package itself can be then exported into ".ezpkg" file. This file can be then imported and installed on other eZ Publish installations.

Content object packages are used for storing actual content objects. If you create some objects and need to use them on other installations, you can export these objects into a content object package.

The package creation wizard will ask you to select nodes and/or subtrees that will be included to the content object package that is being created. It is possible to include class definitions for the objects being exported and related templates from one or several siteaccesses to the package. The selected objects can be exported together with all their versions and languages or you can specify custom parameters. You may choose to keep all node assignments or only main nodes for the objects being imported and specify what to do with related objects.

Datatypes serialization support

In eZ Publish 3.8 all the built-in datatypes are compatible with the package system. Both object and class serialization are supported. If you use an additional datatype that does not support serialization then you will see a warning when trying to export/import class definitions and/or content objects containing attributes of this datatype.

Extension packages

These packages store extension files. If you create an extension and need to use it on other installations, you can export it into an extension package. The administration interface makes it possible to create extension packages.

Design / site style packages

Site style packages store site design themes. Such packages make it possible to change the look and feel of the site easily. A site style package includes non-content specific images (logos, banners, graphical layout elements etc.) and two CSS files ("site-colors.css" containing styles like color codes and background image details for the pagelayout (page 180) and "classes-colors.css" that determines styles for class templates).

Please note that site style packages can not be installed or uninstalled. If you import several design packages, you will be able to switch between them using the administration interface. The next subsection explains how this can be done.

Changing the site style theme

Let's say that you have imported a new site style package. To change the look and feel of your site according to the imported design theme, do the following:

1. Click the "Design" tab in the administration interface and select "Look and Feel" from the menu on the left.
2. Select the desired site style from the "Sitestyle" list.
3. Click the "Send for publishing" button to save your changes.
4. Go to the actual site and refresh the page. If you can't see any changes then you should clear eZ Publish caches.

Site packages

The special packages provide basic site examples like "News", "Shop", "Gallery" etc. mostly for the purpose of demonstration and learning. Site packages do not contain any objects. However, they contain dependencies to other packages plus specific settings and scripts. These packages can not be installed or uninstalled. Site packages can be thought of as "meta packages" that are used only in the setup wizard when you are installing eZ Publish. (If you remove a site package from internal repository, this will not affect the behavior of the installed system.) It is not possible to create site packages using the administration interface.

The setup wizard automatically fetches the list of available site packages from remote and internal repositories and asks the user to choose one. It will automatically download the selected site package and all its dependent packages, import them to the system and display a list of successfully imported packages. (This step will be omitted if all these packages are already stored under internal repositories.) All dependent packages except for the site style package will be automatically installed.

Example

The "Shop site" package v.2.0.6 contains dependencies to the following packages:

- Three content object packages called "Products", "Multi-price products", "Dynamic VAT products".
- A site style package called "Theme 04".
- An extension package called "ezpaypal extension".

Choosing the "Shop site" package in the setup wizard will result in downloading this package and five dependent packages from <http://packages.ez.no/ezpublish/4.4>. The downloaded ".ezpkg" files will be unpacked into the "var/storage/packages/ez_systems" directory (i.e. these packages will be imported to the system). The wizard will then automatically install the "Products", "Multi-price products", "Dynamic VAT products" and "ezpaypal extension" packages. The "Shop site" and "Theme 04" packages will not be installed (as site package and site style package). When the setup wizard is finished, you can safely remove the "shop_site" package manually or using the administration interface. The removal of a site package will not affect any of its dependent packages.

4.7.2 Creating new packages

The administration interface allows you to export your class definitions, content objects, settings, design styles etc. into packages of different types. This functionality is implemented using the built-in package creation handlers for the following types of packages:

- Content class packages
- Content object packages
- Extension packages
- Site style packages (design packages)

The built-in package creation handlers are stored in the "kernel/classes/packagecreators" directory. Please note that there is no package creation handler for site packages and thus it is not possible to create such packages using the administration interface. The packages created locally are stored under the "Local" system repository.

The next subsections explain how to create packages of different types.

Content class packages

The following example demonstrates how to create a content class package.

1. Click the "Setup" tab in the administration interface and access the "Packages" link on the left. You will be taken to the list of packages located under the "Local" system repository. (This interface can also be accessed by requesting "/package/list" in the URL.)

(see figure 4.32)

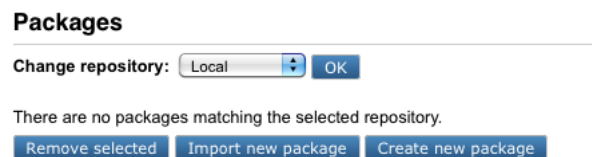


Figure 4.32:

2. Click the "Create new package" button located under the list of packages. The system will bring up the package creation dialog where you can choose between four available package creation wizards. (This interface can also be accessed by requesting "/package/create" in the URL.)

(see figure 4.33)

Choose the "Content class export" wizard as shown in the screenshot and click the "Create package" button.

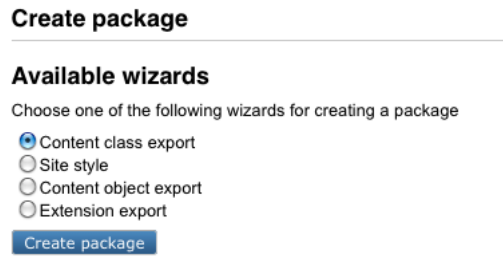


Figure 4.33:

- The package wizard starts with asking what classes to include to the package being created (look at the next screenshot).

Select the desired class(es) from the list and click the "Next" button.

(see figure 4.34)

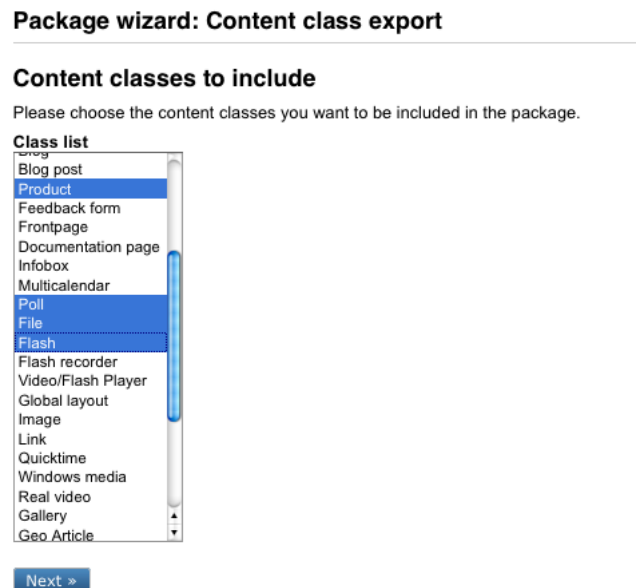


Figure 4.34:

- Now, it's time to enter some information about the content class package. Give it a name and enter some text to describe the package as shown below. Click the "Next" button.

(see figure 4.35)

- The system also needs some information about the package maintainer. Enter this information and click the "Next" button.

(see figure 4.36)

Package wizard: Content class export**Package information**

Provide some basic information for your package.

Package name

4_classes

Summary

Export of 4 content classes

Description

This package contains exported definitions of the following content classes:

- Product
- Poll
- File
- Flash

Version

1.0

License

GPL

Package host

geir-arne-waalers-macbook-pro.local:8081

Packager

Administrator User

Next >

Figure 4.35:

Package wizard: Content class export**Package maintainer**

Provide information about the maintainer of the package.

Name

Administrator User

Email

mail@example.com

Role

Lead

Next >

Figure 4.36:

6. In the last step, you can optionally enter some information about the changes you applied to this version of the package (look at the next screenshot).

(see figure 4.37)

 After clicking the "Continue" button the wizard will create the package and display its summary.

Package wizard: Content class export

Package changelog
Please provide information on the changes.

Name
Administrator User

Email
mail@example.com

Changes
Start an entry with a marker (- (dash) or * (asterisk)) at the beginning of the line. The change will continue to the next change marker.

- Creation of package.

[Continue](#)

Figure 4.37:

Content object packages

The following example demonstrates how to create a content object package.

1. Click the "Setup" tab in the administration interface, select the "Packages" link on the left and access the "Create new package" button located under the list of packages. In the package creation dialog choose the "Content object export" wizard as shown in the screenshot below and click the "Create package" button.

(see figure 4.38)

Create package

Available wizards
Choose one of the following wizards for creating a package

Content class export
 Site style
 Content object export
 Extension export

[Create package](#)

Figure 4.38:

2. The package wizard starts from asking which objects to include to the package being created (look at the next screenshot).

(see figure 4.39)

The following text describes how you can use the "Add node", "Add subtree" and "Remove selected" buttons for choosing the desired objects.

Package wizard: Content object export**Content objects to include**

Choose the objects to include in the package.

There are currently no objects selected for export



Figure 4.39:

- The "Add node" button makes it possible to add individual objects to the package which is being created. When you click this button, you will see a dialog called "Choose node for export". This dialog will display the nodes that are located inside the "Content structure" tree. Use the list to select the nodes (which encapsulate the objects) that you want to include in the package. The following screenshot shows this dialog where the node which encapsulates an article node called "Global zone" is selected.

(see figure 4.40)

Choose node for export

Please choose the node(s) you wish to export.



Figure 4.40:

Please note that it is possible to select multiple nodes/objects at the same time. You can navigate the list by clicking on the names of the nodes. If the desired node is located outside the "Content structure" tree, click the up arrow icon/button until it brings you to the root of the tree. (This operation will allow you to for example switch to the "Media library" tree and select image objects that are located there.) It is possible to reconfigure how the list is displayed. For example, you can set the quantity of objects per page by clicking the "10" / "25" and "50" links. If you wish to browse image objects as thumbnails, simply click the "Thumbnail" button. After selecting the desired node(s) click the "OK" button to save your choice.

- The "Add subtree" button makes it possible to add whole subtrees to the package which is being created. When you click this button, you will see a dialog called "Choose subtree for export". This dialog is very similar to the node choosing dialog described above. The only difference is that selecting a node here means that the whole subtree located under it will be included in the package. Let's select for example a subtree located under one of the folders (look at the next screenshot).

(see figure 4.41)



Figure 4.41:

After selecting the desired subtree(s), click "OK".

- The wizard will display the selected objects/nodes and subtrees as shown in the following screenshot.

(see figure 4.42)

Package wizard: Content object export

Content objects to include

Choose the objects to include in the package.

Selected nodes

Node	Export type
<input type="checkbox"/> Multiupload	subtree
<input type="checkbox"/> Live Video	subtree

Remove selected Add subtree Add node

Next >

Figure 4.42:

If you have mistakenly chosen item(s) that you don't want to be included to the package, use the check-boxes to select these items and click the "Remove selected" button. If everything is correct, click the "Next" button.

3. In the next dialog you should specify the desired export properties for the objects being added to the package. It is possible to include not only the actual objects but also their class definitions and related templates (the templates can be taken from one or several siteaccesses). The selected objects can be included together with all their versions and languages or you can specify custom parameters. You may choose to keep all node assignments or only main nodes for the objects being imported and specify what to do with related objects. The following screenshot shows this dialog box.

(see figure 4.43)

Package wizard: Content object export

Content object limits
Specify export properties. The default settings will most likely be suitable for your needs.

Miscellaneous

- Include class definitions.
- Include templates related to exported objects.

Select templates from the following siteaccesses

ezflow_site
eng
fre

Versions

- Published version
- All versions

Languages
Select languages to export

English (United Kingdom)
Français (France)
Deutsch (Deutschland)

Node assignments

- Keep all in selected nodes
- Main only

Related objects

- Keep all in selected nodes
- None

Figure 4.43:

Choose the desired properties and click the "Next" button.

4. The rest three steps of the "Content object export" wizard allow you to enter information about the package itself, its maintainer and changes made in the current version. These are already described above for "Content class export" wizard.

Extension packages

The following example demonstrates how to create an extension package.

1. Go to "Setup - Packages" in the administration interface and click the "Create new package" button located under the list of packages. In the package creation dialog choose the "Extension export" wizard and click the "Create package" button.
2. The wizard will display the list of existing extensions. Select the extension that you wish to export to the package (as shown in the following screenshot) and click the "Next" button.
(see figure 4.44)

Package wizard: Extension export

Extensions to include

Please select the extensions to be exported.

- ezcomments
- ezflow
- ezgmaplocation
- ezie
- ezjscore
- ezmultiupload
- ezodf
- ezoe
- ezscriptmonitor
- ezstarrating
- ezwebin
- ezwt

Next >

Figure 4.44:

3. The rest three steps of the "Extension export" wizard allow you to enter information about the package itself, its maintainer and changes made in the current version. These are already described above for "Content class export" wizard.

Design / site style packages

The following example demonstrates how to create a site style package.

1. Go to "Setup - Packages" in the administration interface and click the "Create new package" button located under the list of packages. In the package creation dialog choose the "Site style" wizard and click the "Create package" button.
2. The wizard will ask you for a thumbnail image, which should be a screenshot or an icon that depicts the look and feel of your theme. The image should be 120px wide and 103px high. The following screenshot shows this dialog.

(see figure 4.45)

Choose the image file and click the "Next" button.

Package wizard: Site style**Package thumbnail**

Please select a thumbnail file to be included in the package, if you do not want to have a thumbnail simply click Next.

Figure 4.45:

- The next dialog requests that you provide two CSS files: the "site-colors.css" file containing styles like color codes and background image details for the pagelayout (page 180) and "classes-colors.css" that determines styles for class templates. Choose these files as shown in the screenshot below (the actual file names do not matter) and click the "Next" button.

(see figure 4.46)

Package wizard: Site style**CSS files**

Please select the site CSS file to be included in the package.

Please select the classes CSS file to be included in the package.

Figure 4.46:

- If you use images in your theme, you can upload them in the next screen (look at the screenshot).

(see figure 4.47)

Click "Next" when you finish adding images.

- The rest three steps of the "Site style" wizard allow you to enter information about the package itself, its maintainer and changes made in the current version. These are already described above for "Content class export" wizard.

Site packages

It is impossible to create site packages using the administration interface (i.e. there is no package creation handler for these package types). They can only be created manually, which means that a package creator will have to edit the "package.xml" file.

Package wizard: Site style

Image files

Select an image file to be included in the package then click Next.
Click "Next" without choosing an image to continue to the next step.

Currently added image files

- 1000px-Fedora_diagram.png
- ez_systems_intranet_logo.jpg

Figure 4.47:

4.7.3 Exporting packages to files

A package located under an internal repository can be exported to an ".ezpkg" file. The following list reveals how this can be done.

1. Select the "Setup" tab in the administration interface and click the "Packages" link on the left. You will be taken to the list of packages located under the "Local" system repository (look at the next screenshot).

(see figure 4.48)

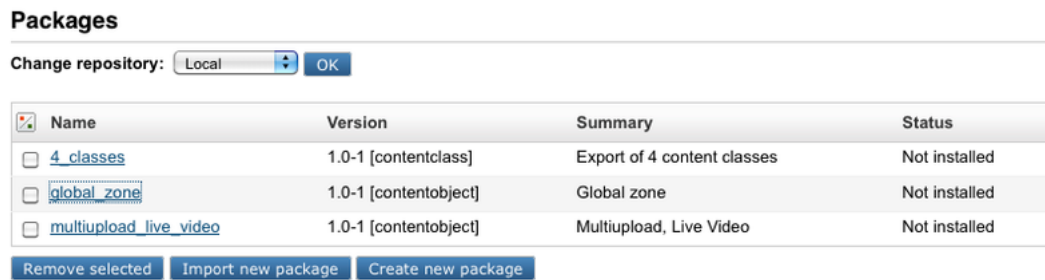


Figure 4.48:

This interface can also be accessed by requesting "/package/list" in the URL. If you wish to view packages from another internal repository, select the name of repository from the drop-down list and click the "Change repository" button.

2. Find the package you wish to export and click on its name. The system will display the package summary as shown in the following screenshot.

(see figure 4.49)

Click the "Export to file" button in order to download the ".ezpkg" file.

global_zone-1.0-1 [contentobject] - Not installed

Summary

Global zone

State

stable

License

GPL

Maintainers

[Administrator User](#) (lead)

Description

This package contains the following nodes : Global zone - node

Documents

[LICENCE](#)

Changelog

(15/09/2010 5:04 pm) [Administrator User](#)

- Creation of package.

[Install](#) [Export to file](#)

Figure 4.49:

4.7.4 Importing packages to the system

A package that is stored as an ".ezpkg" file can be imported to the system - i.e. uploaded, unpacked and placed under an appropriate internal repository within the installation. The following example demonstrates how to import a site style package.

1. Go to "Setup - Packages" in the administration interface and click the "Import new package" button located under the list of packages (look at the next screenshot).

(see figure 4.50)

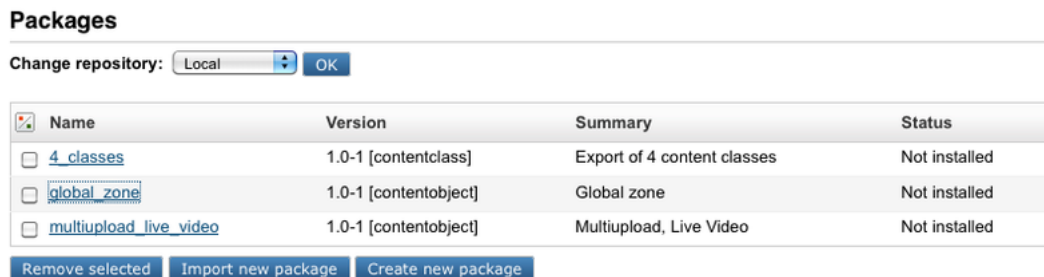


Figure 4.50:

2. Choose the desired ".ezpkg" file on your local computer (as shown in the following screenshot) and click the "Import package" button.

(see figure 4.51)



Figure 4.51:

 The system will import the package from the ".ezpkg" file and show you the package summary.

Please refer to the "Changing the site style theme" sub-section to learn how the design theme from the imported site style package can be applied to your site.

4.7.5 Removing packages from repository

The following instructions reveal how you can remove packages from the system repository.

1. Go to "Setup - Packages" in your administration interface, select an internal repository and click the "Change repository" button.
2. Select the package(s) you that wish to remove (as shown in the screenshot below) and click the "Remove selected" button.

(see figure 4.52)

Packages

Change repository: eZ-systems

Name	Version	Summary	Status
<input type="checkbox"/> ezflow_classes	2.1-0beta2 [contentclass]	ezflow content classes	Installed
<input type="checkbox"/> ezflow_democontent	2.1-0beta2 [contentobject]	Home, Conference, Discussion Forum, Conference Blog, Live Video	Installed
<input type="checkbox"/> ezflow_design	2.1-0beta2 [sitestyle]	eZ Flow design	Imported
<input type="checkbox"/> ezflow_extension	2.1-0beta2 [extension]	ezflow_extension	Installed
<input type="checkbox"/> ezflow_site	2.1-0beta2 [site]	eZ Flow	Imported
<input type="checkbox"/> ezgmaplocation_extension	1.1-0beta2 [extension]	ezgmaplocation extension	Installed
<input type="checkbox"/> ezstarrating_extension	1.1-0beta2 [extension]	ezstarrating extension	Installed
<input type="checkbox"/> ezwebin_extension	1.6-0beta2 [extension]	ezwebin extension	Installed
<input checked="" type="checkbox"/> ezwt_extension	1.2-0beta2 [extension]	ezwt extension	Installed

Figure 4.52:

The selected package(s) will be removed from the repository.

Please note that if you remove an installed package, it will not be uninstalled. Only the package files will be removed from the internal repository.

4.7.6 Installing packages

It is possible to install packages that are located under internal repositories. Note that this is not true for site packages and design packages. When you install a package, the system will create content classes and content objects, apply configuration settings and so on. Please note that from eZ Publish 3.8, information about installed packages is stored in the "ezpackage" table within the database.

The following sub-sections explain how to install packages of different types.

Content class packages

The following example demonstrates how to install a content class package.

1. Go to "Setup - Packages" in your administration interface, select the internal repository containing the package you wish to install and click the "Change repository" button. Find the desired package and click on its name.

(see figure 4.53)

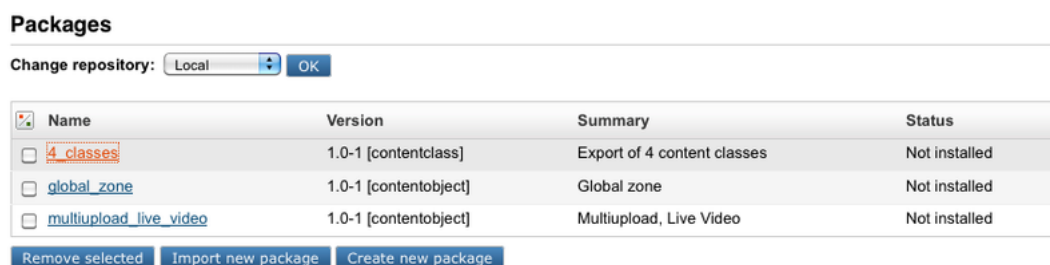


Figure 4.53:

The system will display the package summary as shown in the following screenshot.

(see figure 4.54)

Click the "Install" button.

2. The system starts by showing a list of items that will be created during the package installation (look at the next screenshot). Read this information carefully and click the "Install package" button to continue. Use the "Skip installation" button to abort the operation. (see figure 4.55)
3. If some of the classes being installed already exist, the system will ask the user how this installation conflict should be handled (as shown in the screenshot below).

(see figure 4.56)

If you wish to replace the existing class with the new one, note that all the content objects of the existing class will be removed as well. Use this option only if you know what you're

4_classes-1.0-1 [contentclass] - Not installed**Summary**

Export of 4 content classes

State

stable

License

GPL

Maintainers[Administrator User](#) (lead)**Description**

This package contains exported definitions of the following content classes: - Product - Poll - File - Flash

Documents[LICENCE](#)**Changelog***(15/09/2010 4:47 pm)* [Administrator User](#)

- Creation of package.

[Install](#)[Export to file](#)

Figure 4.54:

Install package

The package can be installed on your system. Installing the package will copy files, create content classes etc., depending on the package. If you do not want to install the package at this time, you can do so later on the view page for the package.

Install items

- Content class 'Product' (product)
- Content class 'Poll' (poll)
- Content class 'File' (file)
- Content class 'Flash' (flash)

[Install package](#)[Skip installation](#)

Figure 4.55:

Installing package '4_classes'.**Class 'Product (imported)' already exists.****Please choose action:**

- Replace existing class
- Skip installing this class
- Keep existing and create a new one

 Use this choice for all the items[Continue](#)[Cancel installation](#)

Figure 4.56:

doing. The remaining options make it possible to skip installing the class or create a new one (in both cases, the existing class and its objects will stay untouched).

After clicking the "Continue" button, the system will install the package and display a summary.

Content object packages

The following example demonstrates how to install a content object package. (Since handling the class installation conflicts is already described in the previous sub-section, let's suppose that no class definitions are included in the package that is being installed.)

1. Go to "Setup - Packages" in your administration interface, select the internal repository containing the package you wish to install and click the "Change repository" button. (see figure 4.57)

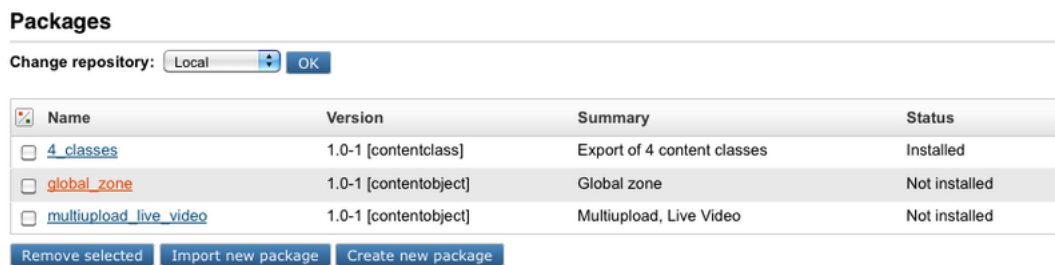


Figure 4.57:

Find the package you wish to install, click on its name, then click the "Install" button.

2. The system will display a list of items that will be created during the package installation (look at the next screenshot). (see figure 4.58)

Install package

The package can be installed on your system. Installing the package will copy files, create content classes etc., depending on the package. If you do not want to install the package at this time, you can do so later on the view page for the package.

Install items

- Content class 'Global layout' (global_layout)
- Content object Global zone (global_layout)

Figure 4.58:

Read this information carefully click the "Install package" button. Use the "Skip installation" to abort the operation.

- If the package contains not only actual content objects but also templates related to these objects, the system will ask which siteaccess these templates should be added to (look at the next screenshot).

(see figure 4.59)

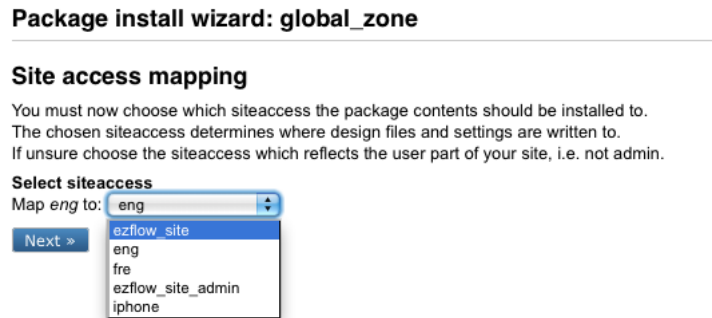


Figure 4.59:

Make your choice and click the "Next" button.

- The next dialog reveals where the installed objects will be located and allow to choose another location if needed.

(see figure 4.60)

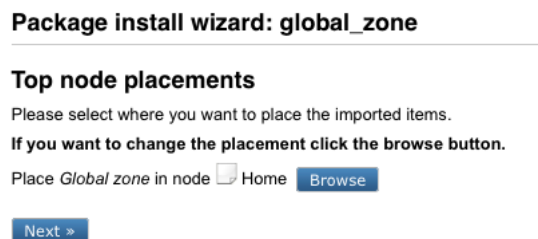


Figure 4.60:

Choose the desired location and click the "Continue" button.

- If some of the objects being installed already exist (i.e. there is another object with the same remote_id), the system will ask how this installation conflict should be handled (see the screenshot below).

(see figure 4.61)

After clicking the "Continue" button, the system will install the package and display a summary.

Installing package 'global_zone'.**Object 'Global zone' already exists.****Please choose action:**

- Replace existing object
- Skip object
- Keep existing and create a new one
- Update existing object

 Use this choice for all the items

Figure 4.61:

Extension packages

The following example demonstrates how to install an extension package.

1. Go to "Setup - Packages" in your administration interface, select the internal repository containing the package you wish to install and click the "Change repository" button. Find the package you wish to install, click on its name and then click the "Install" button.
2. The system will show a list of items that will be created during the package installation (look at the next screenshot).

(see figure 4.62)

Package install wizard: global_zone**Advanced options**
 Use content object modification and publication dates from the package.

Figure 4.62:

Click the "Install package" button to continue. Use the "Skip installation" button to abort the operation.

3. If some of the items being installed already exist, the system will ask how this installation conflict should be handled as shown in the screenshot below.

(see figure 4.63)

After clicking the "Continue" button, the system will install the package and display a summary.

Installing package 'global_zone'.

Object 'Global zone' already exists.

Please choose action:

- Replace existing object
- Skip object
- Keep existing and create a new one
- Update existing object

Use this choice for all the items

Figure 4.63:

4.7.7 Uninstalling packages

Installed packages can be uninstalled. For example, let's say that you have chosen the "Shop" standard site package in the setup wizard while installing eZ Publish and then you decided to use only multi-price products (page 483), not simple price ones. In this case, you can safely uninstall one of the dependent packages which is called "Products" (this content object package contains simple price products needed for the "Shop" site). The following instructions reveal how this can be done.

1. Go to "Setup - Packages" in your administration interface, select the "ez_systems" repository from the drop-down list and click the "Change repository" button. Find the installed package that you wish to uninstall and click on its name. The system will display the package summary as shown in the following screenshot.

(see figure 4.64)

The screenshot shows a package summary for 'ezwt-1.0-1 [extension] - Installed'. The summary includes the following information:

- Summary:** ezwt extension
- State:** stable
- License:** GPL
- Maintainers:** Administrator User (lead)
- Description:** This package contains the ezwt eZ Publish extension
- Documents:** LICENCE, Changelog
- Changelog:** (17/09/2010 1:19 pm) Administrator User
 - Creation of package.

At the bottom of the summary, there are two buttons: 'Uninstall' and 'Export to file'.

Figure 4.64:

Click the "Uninstall" button.

2. The system will display a list of items that will be removed (look at the next screenshot).

(see figure 4.65)

Click the "Uninstall package" button to continue or use "Skip installation" to abort the operation. If one of the listed content classes can not be removed, it will be skipped automatically. For example, the "Folder" content class will not be removed if it is used by one of the top level nodes (page 129).

3. If some of the items being removed have been modified after the package installation, the system will ask for confirmation before removing them.

Uninstall package

The package can be uninstalled from your system. Uninstalling the package will remove any installed files, content classes etc., depending on the package. If you do not want to uninstall the package at this time, you can do so later on the view page for the package. You can also remove the package without uninstalling it from the package list.

Uninstall items

- Extension 'ezwt'

[Uninstall package](#) [Skip uninstallation](#)

Figure 4.65:

 You can choose to keep or remove these items. Make your choice and click the "Continue" button.

4. The system will display a summary. Please note that uninstalled packages will still be available under the system repository (they will be marked as "Imported") and can be installed later if needed.

4.7.8 package.xml format

This section describes important XML tags that are used by the packages system. These tags are used in a "package.xml" file that is located in a package directory.

Items to be installed

The package installation process is determined by the list of items to install. This list is always specified between the <install> and </install> XML tags. (Please note that using two <install> tags inside one "package.xml" file is not allowed.)

An item to install is specified using the <item> XML tag. The following table reveals the list of the tag's attributes.

Attribute	Value
type	The type of the item ("ezcontentclass", "ezcontentobject", "ezextension", "ezinstallscript", etc.)
filename	The name of the ".xml" file that contains information about the item (without the file extension).
sub-directory	The name of the subdirectory which contains the item's ".xml" file.

Example

Let's say that a content class package contains three content classes called "myarticle", "myfolder" and "myproduct". These classes are described in the files called "class-myarticle.xml", "class-myfolder.xml" and "class-myproduct.xml" that are located in the "myclassdir" subdirectory under the package directory. In this case, the "package.xml" file located in the package directory may contain the following lines:

```
<install>
<item type="ezcontentclass"
      filename="class-myarticle"
      sub-directory="myclassdir" />
<item type="ezcontentclass"
      filename="class-myfolder"
      sub-directory="myclassdir" />
<item type="ezcontentclass"
      filename="class-myproduct"
      sub-directory="myclassdir" />
</install>
```

This will instruct the system to install three items of the "ezcontentclass" type in the following order:

- myarticle
- myfolder
- myproduct

Items to be uninstalled

The package uninstallation process is determined by the list of items to uninstall. This list is always specified between the `<uninstall>` and `</uninstall>` XML tags. (Please note that using two `<uninstall>` tags inside one "package.xml" file is not allowed.)

An item to uninstall is specified using the `<item>` XML tag (as described above).

Dependent packages that are required

A site package usually contains dependencies to other packages. Choosing a site package in the setup wizard will result in downloading, importing and installing its dependent packages. The list of dependent packages that the site package requires is always specified between the `<requires>` and `</requires>` XML tags located within the `<dependencies>` and `</dependencies>` pair.

Dependent packages are specified using the `<require>` XML tag. The following table reveals the list of the tag's attributes.

Attribute	Value
type	The type of the package (usually "ezpackage").
name	The internal name of the package.
min-version	The minimal version of the package that can be used.

Example

The "package.xml" file of the "News" site package may contain the following lines:

```
<dependencies>
  <provides />
  <requires>
    <require type="ezpackage"
      name="news"
      min-version="1.0" />
    <require type="ezpackage"
      name="media"
      min-version="1.0-3" />
    <require type="ezpackage"
      name="t01"
```

```
        min-version="1.0" />  
</requires>  
<obsoletes />  
<conflicts />  
</dependencies>
```

This means that the "News" site package requires the following three dependent packages:

- news (version 1.0 or higher)
- media (version 1.0-3 or higher)
- t01 (version 1.0 or higher)

Choosing the "News" site package in the setup wizard will result in downloading the site package itself and all its dependent packages. All dependent packages except from the site style package will be automatically installed.

4.7.9 Custom install scripts

Packages that can be installed via the administration interface may include specific custom install and uninstall scripts. A custom script can be called at any stage during the package installation/uninstallation process. These scripts can be interactive and are capable of displaying extra wizard steps. Interactive scripts are based on a "wizard step" mechanism provided by the class "eZPackageInstallationHandler".

The following example demonstrates how to implement a custom install script for a package.

Example

Let's say that you need to add some additional post-install step(s) to a content object package called "Products" which is located under the "ez_systems" internal repository (in the "var/storage/packages/ez_systems/products" directory). To implement a post-install interactive script for this package, do the following:

1. Create the following new sub-directories in the package directory:
 - post-install
 - post-install/templates
2. Open the "package.xml" file located in the package directory and edit it. Find the list of items to install which is specified between the <install> and </install> XML tags and add the following item in the end of this list:

```
<item type="ezinstallscript"
      filename="myinstallscript"
      sub-directory="post-install" />
```

3. Create a file called "myinstallscript.xml" in the "post-install" directory (this file must contain a description of your install script) and add the following lines to it:

```
<?xml version="1.0" encoding="UTF-8"?>
<install-script filename="myinstallscript.php"
               classname="myInstallScript"
               description="This is my custom install step" />
```

This will tell the system that additional install step(s) is implemented in a PHP class called "myInstallScript" located in the "post-install/myinstallscript.php" file. The text description of the install script will be displayed in the beginning of the package installation process (as shown in the following screenshot).

(see figure 4.66)

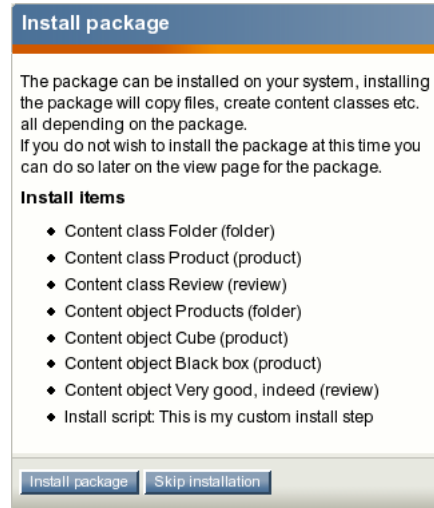


Figure 4.66: Displaying a custom install script in the list of items during the package installation process

4. Create a file called "myownstep.tpl" in the "post-install/templates" directory (this file will contain a template for the additional install step implemented by your install script) and add the following lines to it:

```
<form method="post" action={'package/install'|ezurl}>

  {include uri="design:package/install/error.tpl"}
  {include uri="design:package/install_header.tpl"}

  <p>This is my custom step</p>

  <label>You may even click the checkbox if you want</label>

  <div class="block">
    <input class="button" type="checkbox" name="MyCheckBox" />
  </div>

  {include uri="design:package/navigator.tpl"}

</form>
```

The last step of the package installation will be displayed according to this template (look at the next screenshot).

(see figure 4.67)

5. Create a new file called "myinstallscript.php" in the "post-install" directory. This file must contain a PHP class called "myInstallScript" where all the steps are implemented

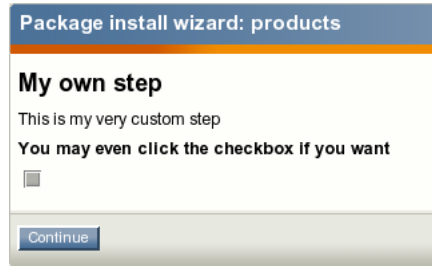


Figure 4.67: Displaying a custom wizard step during the package installation process

(according to the description given in the "post-install/myinstallscript.xml" file). Add the following lines to the "myinstallscript.php" file:

```
<?php
class myInstallScript extends eZInstallScriptPackageInstaller
{
    function myInstallScript( &$amp;package, $type, $installItem )
    {
        eZDebug::writeDebug( $installItem, "Hello from myInstallScript" );

        $steps = array();
        $steps[] = array(
            'id' => 'my_own_step',
            'name' => 'My own step',
            'methods' => array( 'initialize' => 'initializeMyOwnStep',
                'validate' => 'validateMyOwnStep',
                'commit' => 'commitMyOwnStep' ),
            'template' => 'myownstep.tpl' );
        $this->EZPackageInstallationHandler( $package,
            $type,
            $installItem,
            'My own custom step',
            $steps );
    }

    // Function that is called before the step is displayed.
    // You can use it to set variables for your template.

    function initializeMyOwnStep( &$amp;package, &$amp;http, $step, &$amp;persistentData,
        &$amp;tpl, &$amp;module )
    {
        eZDebug::writeDebug( "Hello from initializeMyOwnStep()" );
        return true;
    }
}
```

```
// This function is called after user has submitted the form.
// If this function returns "false", the step will be
// displayed again.

function validateMyOwnStep( &$package, &$http, $currentStepID, &$stepMap,
&$persistentData, &$errorList )
{
    eZDebug::writeDebug( "Hello from validateMyOwnStep()" );
    return true;
}

// This function is called after the form is submitted
// and validated.

function commitMyOwnStep( &$package, &$http, $step, &$persistentData, &$tpl )
{
    eZDebug::writeDebug( "Hello from commitMyOwnStep()" );
    return true;
}
}
?>
```

4.8 Cronjobs

Some features of eZ Publish depend on a maintenance script that takes care of various tasks behind the scenes. This script is located in the root of the eZ Publish directory and should be executed at regular intervals. The script is called "runcronjobs.php". Among other things, it processes workflows (page 165), checks / validates URLs, sends out notification e-mails, etc. Although eZ Publish works without a periodical execution of "runcronjobs.php", it is still recommended to have it running in the background. Some features, for example the notification system (page 509), will not work if the script is not running.

The most common practice is to instruct the operating system to automatically run the script every 30-60 minutes. However, some tasks should be executed more frequently than others and thus it is a good idea to divide the cronjobs into groups/sets and run them separately. Please refer to the "Configuring cronjobs (page 411)" and "Running cronjobs (page 414)" sections for further details.

4.8.1 The cronjob scripts

The "cronjobs" directory contains miscellaneous *cronjob scripts* used for automated periodic and scheduled maintenance. These are described below.

Name	Description	Frequency	Default status
basket_cleanup.php	Cleans up shopping baskets for removed user sessions.	Once a week.	Enabled
clusterbinarypurge.php	Clears expired binary items from DB-based cluster installations (eZDB and eZDFS)	Once a week.	Enabled
hide.php	Hides nodes when a specified date and time is reached.	Not less than once a day.	Enabled
indexcontent.php	Performs delayed search indexing of newly added content objects.	Not less than once a day.	Enabled
old_drafts_cleanup.php	Removes old/unused drafts.	Once a month.	Disabled
internal_drafts_cleanup.php	Removes unused internal drafts.	Once a day.	Enabled
ldapusermanage.php	Synchronizes user account information with an LDAP server.	Not less than once a day.	Disabled
linkcheck.php	Validates published URLs.	Once a week.	Enabled
notification.php	Sends notifications to subscribed users.	Every 15-30 minutes.	Enabled
rssimport.php	Imports RSS feeds.	Not less than once a day.	Enabled
subtreeexpirycleanup.php	Removes expired cache blocks with the "subtree_expiry" parameter.	Not less than once a day.	Enabled
unpublish.php	Removes content objects when a specified date and time is reached.	Not less than once a day.	Enabled
updateviewcount.php	Updates the page view statistics by parsing the	Not less than once a day.	Disabled

	Apache log files.		
workflow.php	Processes the work-flows.	Every 15-30 min-utes.	Enabled

Cleaning up expired data for webshop

The eZ Publish webshop (page 167) functionality allows your customers to put products into their shopping baskets. The items in the basket can then be purchased by initiating the checkout process. The system stores information about a user's shopping basket in a database table called "ezbasket". Information about sessions is stored in the "ezsession" table.

If a user adds products to his basket and then stops shopping (for example closes the browser window) without initiating the checkout process, the session of that user will expire after a while. Expired sessions can be removed either automatically by eZ Publish or manually by the site administrator. When a user's session is removed from the database, the system will not take care of the shopping basket that was created during this session. In other words, the system will remove an entry from the "ezsession" table, but the corresponding entry in the "ezbasket" table (if any) will remain untouched. This behavior is controlled by the "BasketCleanup" setting located in the "[Session]" section of the "site.ini" configuration file (or its override). If it is set to "cronjob" (default), you will have to remove unused baskets periodically by running the "basket_cleanup.php" cronjob script.

Please note that removing unused baskets usually takes a lot of time on sites with many visitors. It is recommended to run the basket cleanup cronjob once a week. If you wish to run it together with other (more frequent) cronjobs, use the "BasketCleanupAverageFrequency" setting located at the same place to specify how often the baskets will actually be cleaned up when the "basket_cleanup.php" cronjob is executed. If you wish to run "basket_cleanup.php" separately from other cronjobs, add the following line to the "[Session]" section of your "cronjob.ini.append.php" file:

```
BasketCleanupAverageFrequency=1
```

Please note that there is no need to run this cronjob if your site does not use the webshop functionality (or if it does but the "BasketCleanup" setting is set to "pageload").

Hiding nodes at specific times

The system can automatically hide a node when a specified date is reached. For example, you may wish that an article published on your site should become invisible within a couple of days/weeks/months. However, you are not interested in removing the article, you just want to hide it. In this case, you will have to add a new attribute using the "Date and time" datatype to your article class and configure the hide cronjob. The following text describes how this can be done.

Adding an attribute to the "Article" class

Go to "Setup - Classes" in the administration interface and select the "Content" class group to view the list of classes assigned to this group. Find the article class and click its corresponding

edit icon/button. You will be taken to the class edit interface. Select the "Date and time" datatype from the drop-down list located in the bottom, click the "Add attribute" button and edit the newly added attribute as shown in the following screenshot.

(see figure 4.68)

Figure 4.68:

Note that both the name and the identifier can be set to anything (you specify what to use in the ini file - see further below) and click "OK". The system will add a new field called "Hide date" (the name of the newly added attribute in the example above) to the class and thus it will appear in the edit interface for the objects (in this case articles). When the articles are edited, this field can be used to specify when the cronjob should hide the nodes. If the attribute is left blank, the article will not be affected by the hide cronjob. Please note that the "hide.php" cronjob must be run periodically for this to work.

Configuring the "hide" cronjob

Add the following lines to your "content.ini.append.php" configuration file:

```
[HideSettings]
RootNodeList []=2
HideDateAttributeList[article]=hide_date
```

You should specify the identifier of the newly added attribute in the "HideDateAttributeList" configuration array using the class identifier as a key. In addition, you need to specify the ID number of the parent node for the articles using the "RootNodeList" configuration directive.

Delayed search indexing

If the delayed indexing feature is enabled, newly and re-published objects will not be indexed immediately. In other words, eZ Publish will not index the content during the publishing process.

Instead, the indexing cronjob will take care of this in the background and thus publishing will go a bit faster (since you don't have to wait for the content to be indexed). In order for this to work, the "indexcontent.php" cronjob must be executed periodically in the background (otherwise the content will be published but not indexed).

Please note that you do not need to run this script unless you have "DelayedIndexing=enabled" in the "[SearchSettings]" section of the "site.ini" configuration file (or an override).

Cleaning up old/unneeded drafts

Regular drafts (status "0")

The purpose of the "old_drafts_cleanup.php" cronjob script is to remove old drafts from the database. If enabled, this script will remove drafts that have been in the system for over 90 days. To set the number of days, hours, minutes and seconds before a draft is considered old and can be removed, specify the desired values in the "DraftsDuration[]" configuration array located in the "[VersionManagement]" block of an override for "content.ini". The maximal number of drafts to remove at one call of the script (100 by default) is controlled by the "DraftsCleanUpLimit" setting located at the same place.

Internal/untouched drafts (status "5")

The purpose of the "internal_drafts_cleanup.php" cronjob script is to remove drafts that probably will never be published. If a version of a content object is created but not modified (for example, if someone clicked an "Add comments" button but didn't actually post anything), the status of the version will be "5". The "internal_drafts_cleanup.php" script will remove status "5" drafts that have been in the system for over 24 hours (1 day). To set the number of days, hours, minutes and seconds before an internal draft is considered old and can be removed, specify the desired values in the "InternalDraftsDuration[]" configuration array located in the "[VersionManagement]" block of an override for "content.ini". The maximal number of internal drafts to remove at one call of the script (100 by default) is controlled by the "InternalDraftsCleanUpLimit" setting located at the same place.

Synchronizing user data with LDAP server

If the users are authenticated through an [LDAP](#) server, eZ Publish will fetch user account information from the external source and store it in the database. What happens is that it creates local accounts when the users are logging in.

The "ldapusermanage.php" cronjob script can be used to synchronize local user account information with the external source in the background. It is recommended to run this script periodically when the site is connected to an LDAP server. The script will take care of typical maintenance tasks. For example, if a user is deleted from the LDAP directory, it will disable (but not remove) the local account.

Please note that the script will only update the eZ Publish database. Modification of external data (stored on an LDAP server) is not supported.

Checking URLs

In eZ Publish, every address that is input as a link into an attribute using the "XML block" or the "URL" datatype is stored in the URL table (page 138) and thus the published URLs can be inspected and edited without having to interact with the content objects. This means that you don't have to edit and re-publish your content if you just want to change/update a link.

The URL table contains all the necessary information about each address including its status (valid or invalid) and the time it was last checked (when the system attempted to validate the URL). By default, all URLs are valid. The "linkcheck.php" cronjob script is intended to check all the addresses stored in the URL table by accessing the links one by one. If a broken link is found, its status will be set to "invalid". The last checked field will always be updated.

You will have to specify your site URLs using the "SiteURL" configuration directive located in the "[linkCheckSettings]" section of the "cronjob.ini.append.php" file. This will make sure that the "linkcheck.php" cronjob handles relative URLs (internal links) properly.

Please note that the link check script must be able to contact the outside world through port 80. In other words, the firewall must be opened for outgoing HTTP traffic from the web server that is running eZ Publish. From 3.9, it is possible to fetch data using a HTTP proxy specified in the "[ProxySettings]" section of "site.ini" (requires [CURL](#) support in PHP).

Sending notifications

eZ Publish has a built-in notification system (page 509) that allows users to receive information about miscellaneous happenings. It is possible to be notified by email when objects are updated or published, when work-flows are executed and so on. If you are going to use notifications on your site, you will have to run the "notification.php" cronjob script periodically. It will take care of sending notifications to subscribed users (this is done by launching the main notification processing script "kernel/classes/notification/eznotificationeventfilter.php").

If you are using the notification system, it is recommended to run this cronjob script every 15-30 minutes.

RSS import

The RSS import functionality makes it possible to receive feeds from various sites, for example, the latest community news from [www.ez.no](http://ez.no) (<http://ez.no/rss/feed/communitynews>). You will have to configure this using the "Setup - RSS" part of the administration interface and run the "rssimport.php" cronjob periodically. This script will get new items for all your active RSS imports and publish them on your site (if an item already exists, it will be skipped).

Please note that the RSS import script must be able to contact the outside world through port 80. In other words, the firewall must be opened for outgoing HTTP traffic from the web server that

is running eZ Publish. From 3.9, it is possible to fetch data using a HTTP proxy specified in the "[ProxySettings]" section of "site.ini" (requires [CURL](#) support in PHP).

Clearing expired template block caches

If you are using the "cache-block" template function with the "subtree_expiry" parameter to cache the contents of a template block, this cache block will only expire if an object is published below the given subtree (instead of the entire content node tree). The "DelayedCacheBlockCleanup" setting located in the "[TemplateSettings]" section of the "site.ini" configuration file controls whether expired cache blocks with the "subtree_expiry" parameter will be removed immediately or not. If this setting is enabled, the expired cache blocks must be removed manually or using the "subtreeexpirycleanup.php" cronjob script.

Removing objects at specific times

The "unpublish.php" script makes it possible to remove content objects when a specified date is reached. For example, you may wish to delete some articles (move them to the trash) within a couple of days/weeks/months. The following list reveals how this can be done.

1. Add a new attribute of the "Date and time" datatype to your article class using the "Setup - Classes" part of the administration interface. Specify "unpublish_date" as the attribute's identifier; a new field will become available when the objects (in this case articles) are edited. This field can be used to specify when the cronjob should remove the object. If the attribute is left blank, the object will not be affected by the unpublish cronjob.
2. Configure the "unpublish.php" cronjob by adding the following lines to your "content.ini.append.php" configuration file:

```
[UnpublishSettings]
RootNodeList []=2
ClassList []=2
```

You should specify the ID number of your article class in the "ClassList" configuration array and put the ID number of the parent node for your articles to the "RootNodeList" setting.

Analyzing the Apache log files

It is possible to have the view statistics for your site pages stored in the eZ Publish database. To do this, you will have to run the "updateviewcount.php" cronjob script periodically. The script will update the view counters of the nodes by analyzing the Apache log file (the view counters are stored in a database table called "ezview_counter"). When executed, the script will update a log file called "updateview.log" located in the "var/example/log/" directory (where "example" is usually the name of the siteaccess that is being used - it is set by the "VarDir" directive in "site.ini" or an override). This file contains information about which line in the Apache log file the script should start from the next time it is run.

You will also have to create an override for the "logfile.ini" configuration file and add the following lines there:

```
[AccessLogFileSettings]
StorageDir=/var/log/httpd/
LogFileName=access_log
SitePrefix[]=example
SitePrefix[]=example_admin
```

Replace "/var/log/httpd" with the full path of the directory where the Apache log file is stored, specify the actual name of this file instead of "access_log", replace "example" and "example_admin" with the names of your siteaccesses (if you have more than two siteaccesses, list all of them).

Once the correct settings are specified and the "updateviewcount.php" cronjob script is run periodically, you will be able to fetch the most popular (most viewed) nodes using the "view_top_list" template fetch function and/or check how many times a node has been viewed (as described in this example).

Processing workflows

In order to use workflows (page 165), you will have to run the "workflow.php" cronjob script periodically. The script will take care of processing the workflows. For example, let's say you are using the collaboration system and all the changes made in the "Standard" section (page 136) can not be published without your approval. (This can be done by creating a new "Approve" event within a new workflow initiated by the "content-publish-before" trigger function.) If somebody (except the administrator) changes article "A", the system will generate a new collaboration message "article A awaits your approval" for you and another collaboration message "article A awaits approval by editor" for the user who changed it. (Run "notification.php" periodically in order to make it possible for users to be notified by E-mail about new collaboration messages.) You will be able to view your collaboration messages and review/approve/reject the changes using the "My Account - Collaboration" part of the administration interface. However, the changes will not be applied to article A immediately after getting your approval. This will be done next time the "workflow.php" cronjob script is executed.

Clearing expired binary items from DB-based cluster installations (eZDB and eZDFS)

This is done by means of a cronjob and should be used for regular maintenance. The necessary cronjob is located here:

```
cronjobs/clusterbinarypurge.php
```

This cronjob is by default part of a cronjob named cluster_maintenance. Run it with the following script from the root of the eZ Publish directory:

```
$php runcronjobs.php cluster_maintenance -s ezwebin_site
```

Note that "\$php" should be replaced by the path to your php executable. Also how to run a script can differ depending on your system. Please consult your systems manual for more information.

It is recommended to run this cronjob weekly, although it can be done more frequently for websites where binary files are removed frequently.

For more information please visit our [Issue Tracker: issue 015793](#).

4.8.2 Configuring cronjobs

You can configure which cronjob that will be enabled (can be executed by the "runcronjobs.php" script) from within an override for the "settings/cronjob.ini" configuration file. The following list reveals which settings that can be specified in the "[CronjobSettings]" section of this file.

- The ScriptDirectories configuration array specifies the directories where eZ Publish will search for built-in cronjob scripts (the "cronjobs" directory is used by default).
- The ExtensionDirectories directive specifies the extension directories where eZ Publish will search for additional/custom cronjob scripts. By default, eZ Publish will search in the "cronjobs" sub-directory inside your extension(s).
- The Scripts array contains a list of cronjob scripts that will be run when the main "run-cronjobs.php" script is executed without specifying the "group_of_tasks" option. The tasks specified in this configuration array is called the *main set of cronjobs*.

Cronjob parts

Some cronjobs must be executed more frequently than others. It is possible to configure *additional sets of cronjobs* by adding specific sections (cronjob parts) to an override for "cronjob.ini".

The next examples demonstrate how the cronjobs can be configured.

Example 1 (default settings)

The following settings are specified in the "[CronjobSettings]" section of "cronjob.ini" by default:

```
[CronjobSettings]
ScriptDirectories []=cronjobs
Scripts []=unpublish.php
Scripts []=rssimport.php
Scripts []=indexcontent.php
Scripts []=hide.php
Scripts []=subtreeexpirycleanup.php
Scripts []=internal_drafts_cleanup.php
ExtensionDirectories []
```

This means that the main set of cronjobs contains the following six tasks:

- unpublish.php
- rssimport.php
- indexcontent.php
- hide.php

- subtreeexpirycleanup.php
- internal_drafts_cleanup.php

These scripts will be run each time the "runcronjobs.php" script is executed without the "group_of_tasks" option. The system will expect these scripts to be located in the "cronjobs" directory.

The following configuration blocks (cronjob parts) located in the "settings/cronjob.ini" configuration file specify two additional sets of cronjobs called "infrequent" and "frequent":

```
[CronjobPart-infrequent]
Scripts[]=basket_cleanup.php
Scripts[]=linkcheck.php

[CronjobPart-frequent]
Scripts[]=notification.php
Scripts[]=workflow.php
```

The settings located in the "[CronjobPart-infrequent]" section instruct the system to run the "basket_cleanup.php" and "linkcheck.php" scripts when the "runcronjobs.php" script is executed in the following way:

```
php runcronjobs.php infrequent
```

The "frequent" set of tasks only includes the notification and work-flow cronjobs. These scripts will be run when the "runcronjobs.php" script is executed in the following way:

```
php runcronjobs.php frequent
```

With this configuration, it is possible to run each set of cronjobs separately, e.g.:

- Workflow.php and notification.php - every 15 minutes.
- The basket cleanup and link check cronjobs - once a week.
- The main set of cronjobs - once a day.

Example 2

If you wish to run the "old_drafts_cleanup.php" cronjob once a month, you can add the following settings to the "cronjob.ini.append.php" file located in the "settings/siteaccess/example" directory (replace "example" by the actual name of the siteaccess):

```
[CronjobPart-monthly]
Scripts[]=old_drafts_cleanup.php
```

The settings located in the "[CronjobPart-monthly]" section will instruct the system to run the "old_drafts_cleanup.php" cronjob script when the "runcronjobs.php" script is executed in the following way:

```
php runcronjobs.php monthly -s example
```

Example 3

It is possible to extend the system by creating custom cronjob scripts. For example, if you have an extension "nExt" that includes a cronjob script "myjob.php", you'll need to put the following lines into an override for the "cronjob.ini" configuration file:

```
[CronjobSettings]
ExtensionDirectories[]=nExt
Scripts[]=myjob.php
```

or

```
[CronjobSettings]
ScriptDirectories[]=extension/nExt/cronjobs
Scripts[]=myjob.php
```

These settings will make eZ Publish expect the additional cronjob script to be located at "extension/nExt/cronjobs/myjob.php". This script will be added to the main set of cronjobs and thus it will be run each time the "runcronjobs.php" script is executed without the "group_of_tasks" option.

4.8.3 Running cronjobs

The "runcronjobs.php" script located in the root of the eZ Publish directory takes care of processing your cronjobs in the background. This script should be executed periodically. The most common practice is to instruct the operating system (or some application) to automatically run the script at regular intervals. On UNIX/Linux systems, this can be done by making use of "cron". On Windows, the script can be run by the "Scheduled Tasks" service. The following text describe how this script can be executed.

Running cronjobs from the shell

It is possible to execute the "runcronjobs.php" script manually from within a system shell:

1. Navigate into the eZ Publish directory.
2. Run the script (replace "example" with the actual name of the siteaccess):

```
php runcronjobs.php -s example group_of_tasks
```

The "group_of_tasks" option indicates that only scripts listed in the "[CronjobPart-group_of_tasks]" section of the "cronjob.ini" configuration file (or its override) will be executed. This parameter is optional. If omitted, the list of scripts will be taken from the "[CronjobSettings]" section of the "cronjob.ini" configuration file. (Please refer to the "Configuring cronjobs (page 411)" section for more information.)

The "-s example" indicates which siteaccess configuration the script should use. If you do not specify a siteaccess when running the script, then the default siteaccess will be used.

It is also possible to use the "-d" parameter that instructs the script to display the debug output at the end of execution, e.g.:

```
php runcronjobs.php -d -s example group_of_tasks
```

You can use this parameter with the "all" option to get more detailed information:

```
php runcronjobs.php -dall -s example group_of_tasks
```

The following options are also available: "accumulator", "debug", "error", "include", "notice", "timing", "warning". Please note when provided, they must be separated using commas. The instructions given in the following example will tell the script to display debug notices and produce a list of includes:

```
php runcronjobs.php -dinclude,notice -s example group_of_tasks
```

The script will not make any changes to log files by default (the ones located in the "var/log" directory of your eZ Publish installation). If you need this functionality, you'll have to run the script using both "-d" and "--logfiles" parameters:

```
php runcronjobs.php -d -s example --logfiles group_of_tasks
```

Cronjobs on UNIX/Linux

"Cron" is the name of a utility that allows the automatic execution of tasks in the background. It is typically used for periodic system administration and maintenance tasks (for example, creating a weekly backup). A program often referred to as the "cron daemon" is running silently in the background, spending its time waiting and executing cronjobs. A "cronjob" is a script or a command that is run at specified intervals by the daemon. The cronjobs must be set up in a [crontab](#). A crontab is a text file that contains information about the intervals and the tasks that should be executed. The crontab files are not intended to be edited directly. The following table reveals which shell commands that can be used for maintaining crontabs:

Shell command	Description
<code>crontab /var/www/ezpublish/ezpublish.cron</code>	Install a new crontab from the "ezpublish.cron" file (replace "/var/www/ezpublish" by the actual path to your eZ Publish directory).
<code>crontab -l</code>	Display the current crontab.
<code>crontab -e</code>	Edit the current crontab. The modified crontab will be installed automatically.
<code>crontab -r</code>	Remove the current crontab.

The following example shows how a cronjob for eZ Publish can be set up in the crontab. It assumes that eZ Publish is located in "/var/www/ezpublish/", that the PHP command line interface program is located at "/usr/local/bin/php" and that the name of the target siteaccess is "example".

```
# The path to the eZ Publish directory.
EZPUBLISH=/var/www/ezpublish

# Location of the PHP command line interface binary.
PHPCLI=/usr/local/bin/php

# Instruct cron to run the main set of cronjobs
# at 6:35am every day
35 6 * * * cd $EZPUBLISH && $PHPCLI runcronjobs.php -q -s example 2>&1

# Instruct cron to run the "infrequent" set of cronjobs
# at 5:20am every Monday
20 5 * * 1 cd $EZPUBLISH && $PHPCLI runcronjobs.php infrequent -q -s example
```



```
2>&1

# Instruct cron to run the "frequent" set of cronjobs
# every 15 minutes
0,15,30,45 * * * * cd $EZPUBLISH && $PHPCLI runcronjobs.php frequent -q -s
example 2>&1

# Instruct cron to run the "monthly" set of cronjobs
# at 4:10am the first day of every month
10 4 1 * * cd $EZPUBLISH && $PHPCLI runcronjobs.php monthly -q -s example 2>&1
```

When added to the crontab, the cron daemon will run the "runcronjobs.php" script using the PHP command line interface binary at the specified time. With this configuration, the main set of cronjobs will be run at 6:35am every day. This means that all the scripts listed in the "[CronjobSettings]" section of the "cronjob.ini" configuration file (or its override) will be executed once a day.

The "infrequent" set of cronjobs will be run at 5:20am every Monday, i.e. the scripts listed in the "[CronjobPart-infrequent]" section of "cronjob.ini" (or its override) will be executed once a week.

The "frequent" set of cronjobs will be run every 15 minutes. Only the scripts that are listed in the "[CronjobPart-frequent]" section of "cronjob.ini" (or its override) will be executed.

The "monthly" set of cronjobs will be run at 4:10am the first day of every month, i.e. the script(s) listed in the "[CronjobPart-monthly]" section of "cronjob.ini" (or its override) will be executed once a week.

The "-q" parameter instructs the script to run in quiet/silent mode (suppressing unnecessary output). The "-s example" indicates which siteaccess configuration the script should use. The "2>&1" notation instructs the system to combine standard output and error messages into one stream.

Scheduled tasks on Windows

Unlike UNIX/Linux systems, Windows does not provide access to cron. Instead, Windows has its own solution called "Scheduled Tasks". A scheduled task can be set up by selecting "Scheduled Tasks" from the Control Panel. This will bring up a wizard that asks what should be executed, when and so on. It should be configured to run a batch (.bat) file at regular intervals. The batch file should navigate into the eZ Publish directory and run the "runcronjobs.php" script.

4.9 Advanced redirection after login

In eZ Publish 3.8 you can configure where to redirect a user when he/she logs in to the system. To enable this possibility for users, do the following:

1. Add an attribute of the "Text line" datatype to your user class. If you have several user classes and wish to enable advanced redirection for all of them then you should add this attribute to each of your user classes (make sure you enter the same attribute identifier for all of them).
2. Specify the identifier of the newly added attribute in the "LoginRedirectionUriAttribute" setting located in the "[UserSettings]" section of the "settings/siteaccess/example/site.ini.append.php" configuration file (replace "example" with the actual name of your siteaccess) like this:

```
LoginRedirectionUriAttribute[key]=attribute_id
```

key

There are two keys that can be used: "user" for user class(es) or "group" for user group class(es).	
attribute_id	The identifier of the newly added attribute (not ID number of the attribute).

Now you can specify the redirection URI in the text line field when creating/editing a user.

This possibility can be also enabled for user groups in the same way as for users. This means that you should add an attribute of the "Text line" datatype to your user group class(es) and specify its identifier in the "LoginRedirectionUriAttribute" setting using "group" as a key.

Example 1

Let's say that a user must be redirected to the "News" folder after log-in. The following list reveals how this could be done:

1. Access the "Setup" tab in the administration interface, click "Classes" on the left and select the "Users" class group. You should see the list of classes assigned to this group. Find your user class there and click the "Edit" button located in the same line of the list. You will be taken to the class edit interface. Select the "Text line" datatype from the drop-down list located in the bottom, click the "Add attribute" button and edit the newly added attribute as shown below. Click "OK" to save your changes.

(see figure 4.69)

2. Specify the identifier of the newly added attribute in the "LoginRedirectionUriAttribute" setting located under the "[UserSettings]" section of an override for the "settings/site.ini" configuration file. To do this, you should add the following line:

Figure 4.69:

```
LoginRedirectionUriAttribute[user]=redirection_uri
```

where "redirection_uri" is the attribute identifier.

3. Access the "User accounts" tab in the administration interface, use the "Sub items" list to find the user and click the "Edit" button located in the same line of the list. You will be taken to the user edit interface. Specify "/news" in the text field called "Redirection URI" (the name of the newly added class attribute) as shown below. Click the "Send for publishing" button to save your changes and the user "John" will be always redirected to the "News" folder after log-in.

(see figure 4.70)

Figure 4.70:

Example 2

Let's say that you wish to redirect all users that belong to the "Guest accounts" group to the "News" folder after log-in. The following list reveals how this could be done:

1. Edit your user group class and add an attribute of the "Text line" datatype as shown below:
(see figure 4.71)
2. Add the following line into the "[UserSettings]" section of an override for the "settings/site.ini" configuration file:

Figure 4.71:

```
LoginRedirectionUriAttribute[group]=start_page
```

where "start_page" is the attribute identifier.

3. Edit the "Guest accounts" user group and specify "/news" in the text field called "Start page" (the name of the newly added class attribute) as shown below.

(see figure 4.72)

Figure 4.72:

Click the "Send for publishing" button to save your changes and all the users that belong to the "Guest accounts" group will be always redirected to the "News" folder after log-in.

Important notes

If a user is a member of several groups (a child of several "User group" nodes), the system will use the redirection URI that is specified for the "main" group (main parent node). The following screenshot shows the user view interface for the user John that belongs to both "Guest

accounts” and ”Editors” user groups. The ”Locations” list located under user preview allows to view and manage locations for the user object that is currently being viewed. The main location is displayed in a bold type (”Users / Guest accounts / John Doe” in our example).

(see figure 4.73)

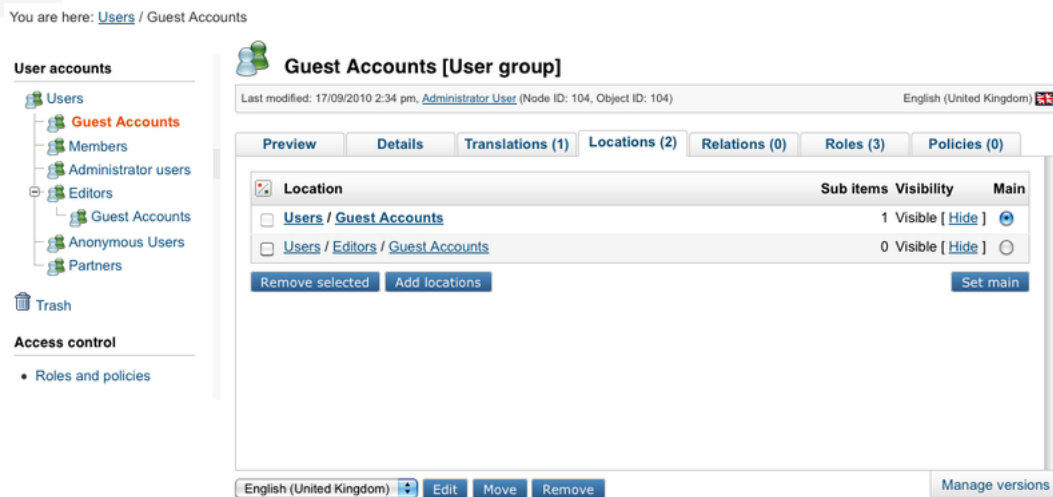


Figure 4.73:

Please note that the advanced redirection feature will get disabled if the redirection URI is already specified (e.g. via the ”LastAccessesURI” session variable. Let’s say that you have specified ”/news” as the redirection URI for user John (see Example 1). If John launches a browser and goes directly to for example ”http://yoursite.com/media_files” then he will not be redirected to ”http://yoursite.com/news”.

4.10 VAT charging system

Charging the value added taxes in your web-shop system is based on the VAT types. A *VAT type* consists of a name and a fixed rate, for example: "Std, 0%". The administration interface makes it possible to add, remove and modify VAT types as described in the "Managing VAT types" section. Although the quantity of the VAT types is not limited, there must be at least one VAT type in your web-shop system. The only purpose of these VAT types is to store some fixed rates of VAT in percent and thus you can call them "static VAT types" or "fixed VAT types".

If you assign a static VAT type to a product then the system will always charge the fixed rate of VAT specified by this VAT type for this product. (This is how the "VAT per product" approach works.)

Price inc. VAT / Price ex. VAT

There are two ways in which the assigned VAT type can be used. This configuration depends on how the product prices are entered when the objects are created. The "Price inc. VAT" alternative is to be used if the prices that are entered already include the value added tax. The "Price ex. VAT" alternative should be used if the prices that are entered do not contain the value added tax. When the first alternative is used and the product is viewed, the price that was entered will be shown. When the second alternative is used and the product is viewed, the price will be the price that was entered plus the amount of VAT.

Dynamic VAT type

The dynamic VAT type does not store any fixed rate of VAT and is not configurable from the administration interface. This VAT type is represented by an additional alternative that is displayed in the list of VAT types when you edit your products. This alternative is called "Determined by VAT charging rules" by default. (The name is specified by the "DynamicVatTypeName" INI setting described in the "VAT settings" section). Choosing this alternative (assigning the dynamic VAT type to a product) will tell the system that no fixed VAT percentage is assigned to this product and thus the amount of VAT should be determined dynamically using some complex VAT charging logic. For example, the amount of VAT can be changed dynamically depending on where the customer lives.

The dynamic VAT type is incompatible with the "Price inc. VAT" configuration. You should set the "Price ex. VAT" configuration for your products and specify prices that do not contain the value added tax. Please note that this VAT type is connected with the VAT handlers mechanism and is disabled if no handler is used.

VAT handlers

If you wish to use some complex VAT charging logic, it must be implemented in a *VAT handler* i.e. PHP class providing a mechanism that determines the rate of VAT for a product dynamically in accordance with the implemented logic. You can either use the built-in default VAT handler that

supports the "Country dependent VAT" approach or extend the system by creating your own VAT handler (the "Extended VAT" approach). Using two or more VAT handlers at the same time is not supported.

The VAT handler to use must be specified in the "Handler" INI setting described in the "VAT settings" section. To enable the built-in default VAT handler, you will have to add the following line to the "[VATSettings]" section in an override for the "settings/shop.ini" configuration file:

```
Handler=ezdefault
```

The system will start to use the default VAT handler and add the dynamic VAT type to the list of VAT types that is displayed when you create/edit a product or product class.

Please note that dynamic VAT type is a kind of virtual structure that comes into being after enabling a VAT handler. If no VAT handler is enabled, the dynamic VAT type is not displayed and can not be used.

4.10.1 Assigning VAT types to products

To assign a VAT type to a product, edit this product and select the desired VAT type from the drop-down list called "VAT type" as shown in the following screenshot. This can be done for both simple price and multi-price products (supported by both price and multi-price datatypes).

(see figure 4.74)

The screenshot shows a web interface for editing a product named "Persian" (a Carpet). The interface includes a language selector set to "English (United Kingdom)". The "Name" field is "Persian". The "Price" field is set to "1,250.00 (+ Std, 0%)". The "VAT" configuration is set to "Price inc. VAT". The "VAT type" dropdown menu is open, showing the following options: "Std, 0%", "Norway general, 25%", "Norway reduced, 11%", "Norway low, 7%", and "Determined by VAT charging rules".

Figure 4.74: Setting the VAT type on the object level.

The screenshot above shows a part of the object edit interface for a simple price product called "Persian". Since the price value is set to \$1,250 and the "Price inc. VAT" configuration is selected, the actual product price displayed to a customer will be \$1,250. If you assign the 25% static VAT to this product then the amount of VAT will be \$250.

If you then select the "Price ex. VAT" configuration then the amount of VAT will be calculated like this:

$$1,250.00 * 25 / 100 = 312.50$$

The actual product price for customers will be calculated like this:

$$1,250.00 + 312.50 = 1,562.50$$

If you select the last item called "Determined by VAT charging rules" then the dynamic VAT type will be assigned to this product. This VAT type is only compatible with the "Price ex. VAT" configuration.

Default VAT type for a product class

It is possible to choose *the default VAT type* on the class level (when you create a new product class or edit an existing one). This VAT type will be used by default when a new object of this class is created. However, it will be still possible to choose another VAT type for each individual product.

The following list reveals how you can set the default VAT type for a product class.

1. Access the "Setup" tab in the administration interface, click "Classes" on the left and select the "Content" class group. You should see the list of classes assigned to this group. Find your product class there and click the "Edit" button located in the same line of the list. You will be taken to the class edit interface.
2. Find the class attribute of the price or multi-price datatype. You should see a drop-down list called "Default VAT type" there. Select the desired VAT type from this list as shown in the following screenshot and click the "OK" button to save your changes.

(see figure 4.75)

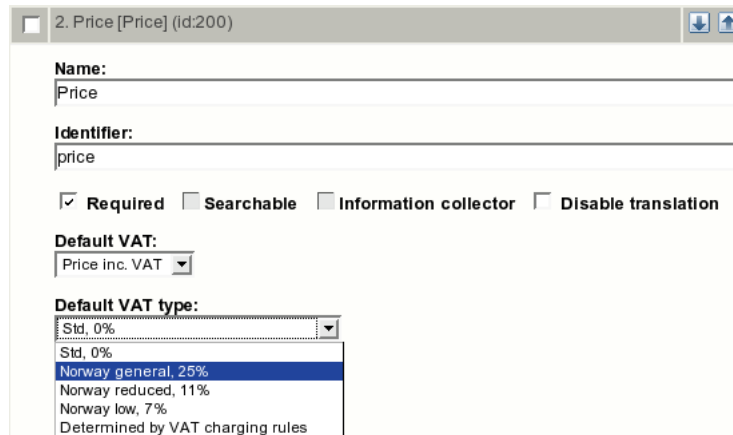


Figure 4.75: Setting the default VAT type on the class level.

The screenshot above shows a part of the class edit interface for a simple price product class called "Carpet". Here you can select "Price inc. VAT" or "Price ex. VAT" as default configuration for all newly created carpets and choose for example "Norway general, 25%" as the default VAT type for them. Please note that you will be able to change these default settings on the object level (for each particular carpet). If you select the last item called "Determined by VAT charging rules" then the dynamic VAT type will be assigned by default to all newly created carpets. This VAT type can only be used with the "Price ex. VAT" configuration.

4.10.2 Three approaches to VAT charging

The webshop system supports the following three approaches to VAT charging:

1. VAT per product
2. Country dependent VAT
3. Extended VAT

The next subsections explain the difference between these approaches.

VAT per product

The primitive "VAT per product" approach allows to choose one of the predefined static VAT types when you create a new product or edit an existing one. Thus you can specify a fixed VAT rate for each product. It is also possible to specify the default VAT type for a product class so that this VAT type will be used by default when a new object of this class is created.

Example

Let's say that you sell various products to Norwegian customers and need to charge the following rates of VAT depending on the type of goods:

- A general rate, 25% for most of the products.
- A reduced rate, 11% for food.
- A low rate, 7% for personal transport.

You will have to create these three VAT types (as described in the "Creating a VAT type" section) so that you can assign (page 423) an appropriate VAT type to each product.

It is also possible to create several product classes and specify different default VAT types for them. The value specified for a product class will be selected by default for a new object of this class.

For example, let's say that you have created the following three product classes:

- "Carpet" with default VAT type set to 25%
- "Food product" with default VAT type set to 11%
- "Motorcycle" with default VAT type set to 7%

In this case, the system will assign the 25% fixed VAT type to each newly created carpet, the 11% fixed VAT type to each newly created food product, and the 7% fixed VAT type to each newly created motorcycle.

Country dependent VAT

In most cases, the amount of the VAT depends on where the customer lives. The "Country dependent VAT" approach allows to charge different VAT percentage depending on the product category (if specified) and the country the customer is from. This can be done by using the dynamic VAT type and the built-in default VAT handler. This handler uses the VAT charging rules to determine the appropriate VAT percentage for a product.

VAT charging rules

A VAT charging rule consists of the following components:

- User country (page 432)
- Product category (page 430) (one or more)
- VAT type

and determines which static VAT type to use in case when the customer is from the specified country and the product belongs to one of specified categories. The administration interface makes it possible to add, remove and modify VAT charging rules as described in the "Managing VAT rules (page 443)" section. The *default VAT rule* specified for "Any" country and "Any" category determines which rate of VAT to use in case if all other VAT rules do not match.

The more exact match a rule provides for given "country-category" pair, the higher priority it has. In other words, the default VAT handler tries to choose the best matching VAT percentage. To understand how this VAT choosing algorithm works, look at possible match cases and their priorities described in the following table:

Country	Category	Example	Priority
exact match	exact match	Norway-Food	4
exact match	weak match	Norway-Any	3
weak match	exact match	Any-Food	2
weak match	weak match	Any-Any	1

If there is no match on country and/or no match on category then the lowest (zero) priority will be used.

Setting up country dependent VATs

If you sell for example carpets and need to levy the 16% VAT on purchases made by German customers and 25% on purchases made from Norway then the "VAT per product" approach is not applicable. The following text explains how the "country dependent VAT" approach can be used in this particular case.

1. Enable the built-in default VAT handler as described in the "VAT handlers" section.

2. Create the following two VAT types as described in the "Creating a VAT type" section:
 - Norway general, 25%
 - Germany general, 16%
3. Add an attribute of the country datatype to your user class and specify its identifier in the "UserCountryAttribute" INI setting as described in the "Adding a country attribute to a user class" section.
4. Create the following two VAT rules as described in the "Creating a VAT rule" section:

User country	Product category	VAT type
Norway	Any	Norway general, 25%
Germany	Any	Germany general, 16%

The system will also ask you to create the default VAT rule that will be applied to customers from all other countries.

Since you sell only one type of goods, there is no need to create product categories. The VAT rules specified for "Any" product category will be applied to all your products.

5. To make your products affected by the VAT charging rules, you should assign the dynamic VAT type to them as described in the "Assigning VAT types to products (page 423)" section.

Setting up country and category dependent VATs

If your webshop sells various types of products with different rates of VAT then the rate of VAT will depend on both user country and product category. This means that you will have to create product categories, assign them to your products and specify VAT rules for these product categories (not for "Any" category as described in the previous section).

For example, let's say that you sell various products to Norwegian and German customers and need to charge the following rates of VAT depending on the type of goods:

- Germany
 - A general rate, 16% for most of the products.
 - A reduced rate, 7% for food.
- Norway
 - A general rate, 25% for most of the products.
 - A reduced rate, 11% for food.
 - A low rate, 7% for personal transport.

The following text explains how the "country dependent VAT" approach can be used in this particular case.

1. Enable the built-in default VAT handler as described in the "VAT handlers" section.

2. Create the following four VAT types as described in the "Creating a VAT type" section:
 - Norway general, 25%
 - Germany general, 16%
 - Norway reduced, 11%
 - Norway low, Germany reduced, 7%
3. Add an attribute of the country datatype to your user class and specify its identifier in the "UserCountryAttribute" INI setting as described in the "Adding a country attribute to a user class" section.
4. Add an attribute of the product category datatype to your product class and specify its identifier in the "ProductCategoryAttribute" INI setting as described in the "Adding a product category attribute to a product class" section.
5. Create the following two product categories as described in the "Creating a product category" section:
 - Food
 - Personal transport
6. Create the following five VAT rules as described in the "Creating a VAT rule" section:

User country	Product category	VAT type
Germany	Food	Norway low, Germany reduced, 7%
Germany	Any	Germany general, 16%
Norway	Personal transport	Norway low, Germany reduced, 7%
Norway	Food	Norway reduced, 11%
Norway	Any	Norway general, 25%

The system will also ask you to create the default VAT rule for any category and any country (this VAT rule will be used in case if none of the other VAT rules is applicable).

7. Assign the dynamic VAT type to your products (as described in the "Assigning VAT types to products (page 423)" section) and assign the appropriate product category to each of them (as described in the "Assigning a category to a product" section).

Extended VAT

If you need more complicated VAT charging logic for your webshop, you can extend the system by creating your own VAT handler for special needs. This approach is incompatible with the previous one because using two or more VAT handlers at the same time is not supported. The "Handler" INI setting described in the "VAT settings (page 446)" section determines the VAT handler to use.

The VAT charging logic implemented by your handler will be applied to all products that have the dynamic VAT type assigned. Keep in mind that dynamic VAT type does not work with "Price inc. VAT" configuration.

Please refer to the "Creating new VAT handlers (page [448](#))" section for more information.

Share your information

4.10.3 Product category

The "Country dependent VAT" approach supposes that each of your products can be assigned a product category. The next subsections reveal how this can be achieved. The administration interface makes it possible to add, remove and rename product categories as described in the "Managing product categories (page 440)" section.

Adding new attribute to a product class

It is necessary to add an attribute of the product category datatype to your product class otherwise it will be impossible to assign a category to a product. The following text reveals how this can be done.

1. Access the "Setup" tab in the administration interface, click "Classes" on the left and select the "Content" class group. You should see the list of classes assigned to this group. Find your product class there and click the "Edit" button located in the same line of the list. You will be taken to the class edit interface.
2. Select the "Product category" datatype from the drop-down list located in the bottom, click the "Add attribute" button and edit the newly added attribute as shown below. The following screenshot shows the fragment of the class edit interface with newly added attribute of the product category datatype.

(see figure 4.76)

Figure 4.76: Class attribute edit interface for the "Product category" datatype.

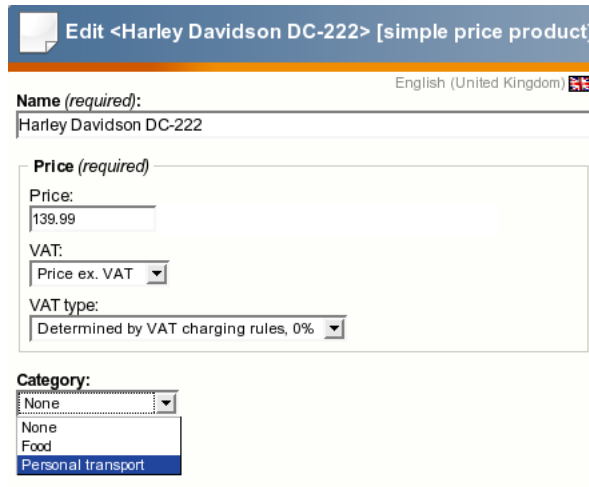
The system will add a drop-down list called "Category" (the name of the newly added attribute) in the object edit interface for products. You can assign a category to the product that is being edited by selecting the desired category from this list.

3. Specify the identifier of the newly added attribute in the "ProductCategoryAttribute" setting located under the "[VATSettings]" section of an override for the "settings/shop.ini" configuration file.

Assigning a category to a product

If your product class contains an attribute of the product category datatype then you can assign a category to a product when editing this product. To do this, edit this product and select the desired category from the drop-down list called "Category" as shown in the following screenshot.

(see figure 4.77)



The screenshot shows a web form for editing a product. The title bar reads "Edit <Harley Davidson DC-222> [simple price product]". The language is set to "English (United Kingdom)". The form contains the following fields:

- Name (required):** A text input field containing "Harley Davidson DC-222".
- Price (required):** A section containing:
 - Price:** A text input field containing "139.99".
 - VAT:** A dropdown menu with "Price ex. VAT" selected.
 - VAT type:** A dropdown menu with "Determined by VAT charging rules, 0%" selected.
- Category:** A dropdown menu with "None" selected. The list of options is: None, Food, Personal transport.

Figure 4.77: A fragment of the product edit interface.

Please note that product categories are always used together with the dynamic VAT type (as you can see in the screenshot above the last item called "Determined by VAT charging rules" is selected). There is no point to select a category for a product with fixed VAT rate.

4.10.4 User country

The "Country dependent VAT" approach supposes that each of your users can be assigned a country. The next subsections reveal how this can be achieved. Note that the list of available countries and their properties (calling codes, etc.) can be configured from within an override for the "country.ini" configuration file.

Adding new attribute to a user class

It is necessary to add an attribute of the country datatype to your user class otherwise it will be impossible to assign a country to a user. The following text reveals how this can be done.

1. Access the "Setup" tab in the administration interface, click "Classes" on the left and select the "Users" class group. You should see the list of classes assigned to this group. Find your user class there and click the "Edit" button located in the same line of the list. You will be taken to the class edit interface.
2. Select the "Country" datatype from the drop-down list located in the bottom, click the "Add attribute" button and edit the newly added attribute as shown below. The following screenshot shows the fragment of the class edit interface with newly added attribute of the country datatype.

(see figure 4.78)

Figure 4.78: Class attribute edit interface for the "Country" datatype.

The system will add a drop-down list called "Country" (the name of the newly added attribute) in the object edit interface for users. You can assign a country to the user that is being edited by selecting the desired country from this list.

- Specify the identifier of the newly added attribute in the "UserCountryAttribute" setting located under the "[VATSettings]" section of an override for the "settings/shop.ini" configuration file.

Assigning a country to a user

If your user class contains an attribute of the country datatype then a country can be assigned to a user in one of the following ways:

- A new user will be asked to specify his/her country when filling in the registration form.
- A site administrator will be able to assign a country to a user when editing a user's details (as described in the "Managing users" section of the "User manual").
- A site administrator can add a toolbar that allows a user to change his country "on-the-fly".

Adding a toolbar for customers

It is recommended that you add a possibility for a user to choose his/her country "on-the-fly" using the "User country" toolbar. To do this, add the following line into the "[Toolbar_right]" section of the "settings/siteaccess/example/toolbar.ini.append.php" file where "example" is your siteaccess name:

```
Tool []=user_country
```

This setting instructs the system to display the country selection toolbar on the right. When a user selects a country, the system will immediately update product prices according to the VAT rules specified for the selected country.

In order to avoid problems with content caching, you will have to specify "user_preferred_country" in the "CachedViewPreferences[full]" setting for all siteaccesses. To do this, open the "site.ini.append.php" configuration file located in the "settings/siteaccess/example" directory (replace "example" with the actual name of the siteaccess) and edit it. If the "[ContentSettings]" section of the configuration file already contains something like

```
CachedViewPreferences[full]=<list_of_user_preferences>
```

then you will have to append a semicolon and "user_preferred_country" at the end of the line, for example:

```
CachedViewPreferences[full]=admin_navigation_content=0;
admin_navigation_details=0;<...>;admin_bookmarkmenu=1;
admin_left_menu_width=13;user_preferred_country=''
```

Note that this configuration line tends to be very long. It is simplified in the example above (a lot of settings were replaced with <...> in order to keep things short).

If the "[ContentSettings]" section does not contain a line that starts from "CachedViewPreferences[full]", create it:

```
CachedViewPreferences[full]=user_preferred_country=''
```

If this setting is not specified, your customers will have problems when changing the country (the interface will not be updated because of the cache problem).

Using alternative country datatypes

There is an additional possibility to use an alternative country datatype instead of the built-in country datatype. This means that you can integrate an alternative datatype to the system so that the user's country will be stored in the same way by the datatype, by the VAT rules management interface and by the shop user registration module (shop/userregister). The following list reveals how this can be achieved.

1. Make sure your datatype's content is either a hash having "value" key or an object capable of getting and setting "value" attribute (like eZPersistentObject). It doesn't matter how the content is actually stored to database, but objectAttributeContent() method must return an array/object. The returned value (usually a country code) is then compared to VAT rules' countries.
2. Override the "view.tpl" and "edit.tpl" templates located in the "templates/shop/country" directory of the standard design in your datatype extension so that countries can be displayed and edited in the VAT rules management interface and the shop user registration module.

4.10.5 Displaying VATs on the actual site

The next subsections explain the way in which the VATs are displayed on the actual site in case of using the "VAT per product" and/or "Country dependent VAT" approach.

VAT per product

Let's say that a fixed rate of VAT is assigned to each product. When a user is viewing a product page, the system displays price including VAT for this product. A user can add products to his shopping basket where the VAT percentage will be displayed for each product. A user should click the "Checkout" button to make an order.

After clicking the "Checkout" button a user will be asked to specify his/her name, email, country and other details needed for customer account and this particular order. This information is handled by the shop account handler that is specified under the "[AccountSettings]" section of the "settings/shopaccount.ini" configuration file. Choosing a country is usually required (this behavior does not depend on the "RequireUserCountry" INI setting described in the "VAT settings (page 446)" section).

Information about the specified country will be stored in the system together with the customer's order. A customer is determined by his unique email, i.e. orders with different emails will belong to different customers. The customer's account contains information about the country that was specified in the first order made by this customer. Please note that a customer account is a special data structure that is used in the webshop system and is not connected with the actual user object.

If you have added an attribute of the country datatype to your user class then it is possible to assign a country to a user. This can be done automatically when the user makes his first order if the attribute identifier is specified in the "UserCountryAttribute" INI setting. Please note that this functionality is generally unneeded for "VAT per product" approach and thus it is disabled if no VAT handler is enabled. Moreover, the system will not automatically change the user's country if it is already set (the user's country can be set when filling in the user registration form, editing the user object or choosing a country from the toolbar).

Country dependent VAT

Let's say that you use the "country dependent VAT" approach, your user class contains an attribute of the country datatype and its identifier is specified in the "UserCountryAttribute" INI setting.

When a user is viewing a product page, the system will display price including VAT for this product. A user can add products to his shopping basket where the VAT percentage will be displayed for each product. After clicking the "Checkout" button a user will be asked to specify his/her name, email, country and other details needed for customer account and this particular order. The system will automatically re-calculate VAT's using the best matching VAT rules for the specified country and include these VAT's into the final product prices that are displayed on the "Confirm order" page.

If a country is assigned to a user, the system will calculate the amount of VAT for a product being

viewed using the best matching VAT rule for user country and product category. If a user selects another country after clicking the "Checkout" button (not the country assigned to this user), this country will not be assigned to the user but will be used only for this particular order.

If no country is assigned to a user then the amount of VAT for a product being viewed will be calculated using the default VAT rule. The country specified by user after clicking the "Checkout" button will be automatically assigned to this user.

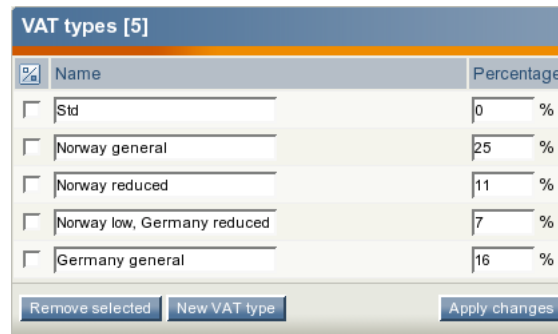
4.10.6 Managing VAT types

This section describes how you can add, remove and modify static VAT types (page 421) using the administration interface.

Creating a VAT type

It is necessary to create the VAT types in order to use value added taxes in your webshop system. The following list reveals how this can be done.

1. Click the "Webshop" tab in the administration interface and select the "VAT types" link on the left. You will be taken to the interface displaying the list of existing VAT types as shown in the following screenshot. This interface can also be accessed by requesting "shop/vattype" in the URL.
(see figure 4.79)



VAT types [5]		
<input type="checkbox"/>	Name	Percentage
<input type="checkbox"/>	Std	0 %
<input type="checkbox"/>	Norway general	25 %
<input type="checkbox"/>	Norway reduced	11 %
<input type="checkbox"/>	Norway low, Germany reduced	7 %
<input type="checkbox"/>	Germany general	16 %

Remove selected New VAT type Apply changes

Figure 4.79: The list of VAT types.

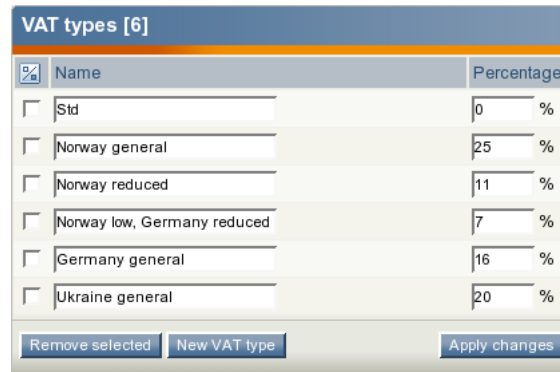
Click the "New VAT type" button. The system will add a new list item called "VAT type 1" with default percentage 0%.

2. Specify the desired name and percentage for this VAT type (see the next screenshot).
(see figure 4.80)
3. Click the "Apply changes" button to save your changes or click the "New VAT type" button to continue adding new VAT types.

Editing a VAT type

If you wish to modify one of your VAT types, do the following:

1. Open the list of VAT types by clicking the "Webshop" tab in the administration interface and selecting the "VAT types" link on the left.



VAT types [6]		
<input type="checkbox"/>	Name	Percentage
<input type="checkbox"/>	Std	0 %
<input type="checkbox"/>	Norway general	25 %
<input type="checkbox"/>	Norway reduced	11 %
<input type="checkbox"/>	Norway low, Germany reduced	7 %
<input type="checkbox"/>	Germany general	16 %
<input type="checkbox"/>	Ukraine general	20 %

Remove selected New VAT type Apply changes

Figure 4.80: The newly added VAT type in the list of VAT types.

2. Specify the desired percentage and/or name for the VAT type you wish to modify (this can be done for several VAT types at the same time).
3. Click the "Apply changes" button to save your changes.

Removing a VAT type

You are not allowed to remove a VAT type that is used as "default VAT type" for your product class. Removing a VAT type that is assigned to your products and/or used by your VAT rules is possible but not recommended. (These VAT rules will be removed and the default VAT type will be assigned to the products.) In most cases, you should change the name and/or percentage of the VAT type instead of removing it from the system.

Please note that you are not allowed to remove all the VAT types. If you do not wish to charge any VAT for your products then just leave one VAT type and set its percentage to zero.

The following text reveals how to remove one or more VAT types from the webshop system.

1. Open the list of VAT types by clicking the "Webshop" tab in the administration interface and selecting the "VAT types" link on the left.
2. Use the checkboxes to select the VAT types that you wish to remove. Do not select all the VAT types.
3. Click the "Remove selected" button.
4. If some of your products use this VAT type and/or some of your VAT rules are based on this VAT type, the system will display a confirmation dialog as shown in the following screenshot.

(see figure 4.81)

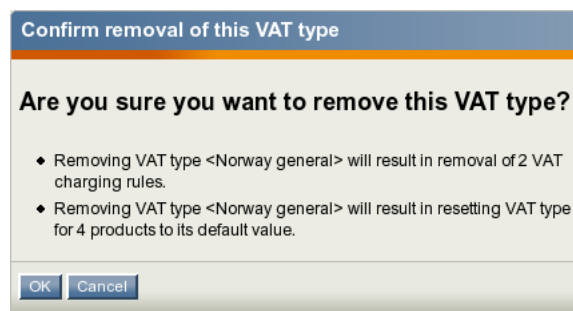


Figure 4.81: *The confirmation dialog.*

4.10.7 Managing product categories

This section describes how you can add, remove and modify product categories (page 430) using the administration interface.

Creating a product category

The administration interface allows you to add new product categories to the webshop system. The following text reveals how this can be done.

1. Click the "Webshop" tab in the administration interface and select the "Product categories" link on the left. You will be taken to the interface displaying the list of existing product categories as shown in the following screenshot. This interface can also be accessed by requesting "/shop/productcategories" in the URL.
(see figure 4.82)

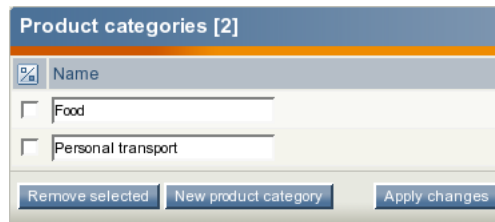


Figure 4.82: The list of product categories.

Click the "New product category" button. The system will add a new list item called "Product category 1".

2. Specify the desired name for this category (see the next screenshot).
(see figure 4.83)

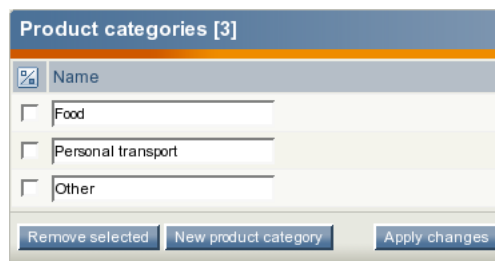


Figure 4.83: The newly added category in the list of product categories.

3. Click the "Apply changes" button to save your changes or click the "New product category" button to continue adding new categories.

Editing a product category

If you wish to change the name of a product category, do the following:

1. Open the list of categories by clicking the "Webshop" tab in the administration interface and selecting the "Product categories" link on the left.
2. Edit the name of the product category (this can be done for several categories at the same time).
3. Click the "Apply changes" button to save your changes.

Removing a product category

Removing a category that is assigned to your products and/or used by your VAT rules is possible but not recommended. In most cases, you should change the name of the category or VAT rules specified for this category instead of removing it from the system. Removing a category will not result in removing all products that belong to this category. The system will unset the "category" attribute for these products and modify or remove the VAT rules specified for this category.

Example

Let's say that you have the following VAT rules (see the next table).

User country	Product category	VAT type
Any	Chocolate, Coffee, Juice	Norway reduced, 7%
Any	Shampoo	Norway general, 25%

If you remove the "Shampoo" category, the last VAT rule will be removed. If you remove the "Chocolate" category, the first VAT rule will be modified as shown in the following table:

User country	Product category	VAT type
Any	Coffee, Juice	Norway reduced, 7%

The following text reveals how you can remove the product category.

1. Open the list of categories by clicking the "Webshop" tab in the administration interface and selecting the "Product categories" link on the left.
2. Use the checkboxes to select the categories that you wish to remove.
3. Click the "Remove selected" button. If there are products and/or VAT rules assigned to this category, the system will display the removal confirmation dialog as shown in the following screenshot. Click "OK" to confirm the removal.
(see figure 4.84)



Figure 4.84: *The confirmation dialog.*

4.10.8 Managing VAT rules

The next subsections describe how you can add, remove and modify the VAT charging rules from the administration interface.

Creating a VAT rule

The administration interface allows you to add new VAT charging rules to the webshop system. The following text reveals how this can be done.

Please note that it is recommended (but not required) to create static VAT types (page 421) and product categories (page 430) (if needed) before creating your VAT charging rules.

1. Click the "Webshop" tab in the administration interface and select the "VAT rules" link on the left. You will be taken to the interface displaying the list of existing VAT rules as shown in the following screenshot. This interface can also be accessed by requesting "/shop/vatrules" in the URL.

(see figure 4.85)



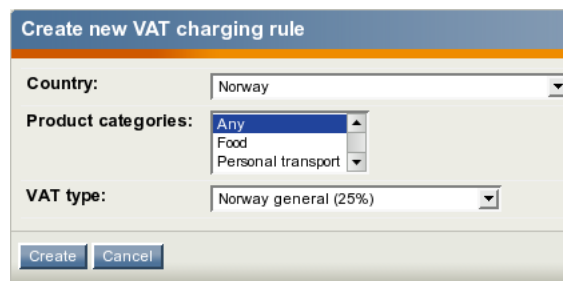
VAT charging rules [4]			
<input type="checkbox"/>	Country	Product categories	VAT type
<input type="checkbox"/>	Germany	Food	Norway low, Germany reduced (7%)
<input type="checkbox"/>	Germany	Any	Germany general (16%)
<input type="checkbox"/>	Norway	Food	Norway reduced (11%)
<input type="checkbox"/>	Norway	Personal transport	Norway low, Germany reduced (7%)

Remove selected New rule

Figure 4.85: The list of VAT charging rules.

Click the "New rule" button. The system will bring up the VAT rule edit interface (look at the next screenshot).

(see figure 4.86)



Create new VAT charging rule

Country: Norway

Product categories: Any
Food
Personal transport

VAT type: Norway general (25%)

Create Cancel

Figure 4.86: The VAT charging rule edit interface.

2. Specify the desired parameters in the following way:

- Choose the desired country from the drop-down list. The VAT rule will be used for customers from this country.
- Select one or more product categories that will be affected by the VAT rule.
- Choose one of the static VAT types from the drop-down list in the bottom. The selected VAT type determines the actual VAT percentage that will be used.
- Click the "Create" button.

The new VAT rule will appear in the list as shown in the screenshot below.

(see figure 4.87)



VAT charging rules [5]			
<input type="checkbox"/>	Country	Product categories	VAT type
<input type="checkbox"/>	Germany	Food	Norway low, Germany reduced (7%) 
<input type="checkbox"/>	Germany	Any	Germany general (16%) 
<input type="checkbox"/>	Norway	Food	Norway reduced (11%) 
<input type="checkbox"/>	Norway	Personal transport	Norway low, Germany reduced (7%) 
<input type="checkbox"/>	Norway	Any	Norway general (25%) 

Remove selected New rule

Figure 4.87: The newly created VAT rule in the list of VAT charging rules.

Editing a VAT rule

The following text reveals how you can edit a VAT charging rule.

1. Open the list of VAT charging rules by clicking the "Webshop" tab in the administration interface and selecting the "VAT rules" link on the left.
2. Find the desired VAT rule and click the "Edit rule" button located in the same table row.
3. The system will bring up the VAT rule edit interface. Specify the desired parameters and click the "Store changes" button.

Removing a VAT rule

The following text reveals how you can remove a VAT rule.

1. Open the list of VAT rules by clicking the "Webshop" tab in the administration interface and selecting the "VAT rules" link on the left.

2. Use the checkboxes to select the VAT rules that you wish to remove.
3. Click the "Remove selected" button.

Share your information

4.10.9 VAT settings

The "[VATSettings]" section of the "settings/shop.ini" configuration file defines the VAT handler that will be used for assigning the value added taxes to your products. Under this section, the following settings can be specified:

- The "Handler" setting specifies the VAT handler that will be used.
- The "RepositoryDirectories[]" array specifies the directories where eZ Publish will search for built-in VAT handlers.
- The "ExtensionDirectories[]" array specifies the extension directories where eZ Publish will search for additional VAT handlers. By default eZ publish will search in the "vathandlers" subdirectory inside your extension.
- The "UserCountryAttribute" setting specifies the identifier of the user country content attribute.
- The "ProductCategoryAttribute" setting specifies the identifier of the product category content attribute.
- The "RequireUserCountry" setting is set to true by default so that the system will always require a user country. If set to false, no error messages will be displayed in case if user country is not specified.
- The "DynamicVatTypeName" setting specifies how the system will display the name of the dynamic VAT type. This alternative is called "Determined by VAT charging rules" by default. You can specify for example "Dynamic VAT", "Country dependent VAT", "Extended VAT" or "My own VAT" in this setting. The system will use this name for the last item in the drop-down list of VAT types located in the class/object view and edit interfaces.

Example 1

The following lines can be specified under the "[VATSettings]" section of the "shop.ini" configuration file:

```
[VATSettings]
Handler=ezdefault
RepositoryDirectories[]=kernel/classes/vathandlers
```

These settings will instruct eZ publish to use the built-in update handler located at "kernel/classes/vathandlers/ezdefaultvathandler.php".

Example 2

You can extend the system by creating custom VAT handlers for special needs. For example, if you have an extension "myextension" that includes a VAT handler "myrule", you can put the following lines into an override for the "shop.ini" configuration file:

```
[VATSettings]
Handler=myrule
ExtensionDirectories[]=myextension
```

or

```
[VATSettings]
Handler=myrule
RepositoryDirectories[]=extension/myextension/vathandlers
```

These settings will instruct eZ Publish to use the VAT handler located at "extension/myextension/vathandlers/myrulevathandler.php".

4.10.10 Creating new VAT handlers

This section reveals some helpful tips for those developers who want to create a new VAT handler (only for people who are familiar with PHP). Please note that it is not recommended to modify the eZ Publish kernel and thus you should implement it as an extension.

Handler interface

This section describes some implementation details that can be useful for PHP developers.

A VAT handler is a file that contains a class implementing the following method:

```
/**
 *
 * \public
 * \static
 * \param $object      The product content object.
 * \param $country     Country the buyer is from, or false if not specified.
 * \return             VAT percent (integer), or null in case of an error.
 */
mixed function getVatPercent( eZContentObject $object, mixed $country );
```

A handler is not called directly but via eZVATManager class. Method getVAT() of that class returns the VAT percentage that should be charged for a given product:

```
$vatPercent = eZVATManager::getVAT( $object, $country );
```

All that getVAT() method does is invoking getVatPercent() method of the handler specified in the "Handler" INI setting.

The next subsection explains how you can implement your own VAT handler.

Creating your own handler

Let's say that you need to determine the VAT percentage for a product depending on the section this product belongs to. You can create your own VAT handler called "mysectionbased" as described below.

1. Create the following sub-directories in the "extension" directory of your eZ Publish installation:
 - myextension
 - myextension/settings
 - myextension/vathandlers

2. Create a file called "mysectionbasedvathandler.php" in the "myextension/vathandlers/" directory (this file must contain a PHP class called "MySectionBasedVATHandler") and add the following lines into it:

```
<?php
class MySectionBasedVATHandler
{
    /**
     * \public
     * \static
     */
    function getVatPercent( $object, $country )
    {
        $section = $object->attribute( 'section_id' );
        if ( $section == 1 )
            $percentage = 10;
        else
            $percentage = 20;
        return $percentage;
    }
}
?>
```

3. Create a file called "shop.ini.append.php" in the "myextension/settings" directory and add the following lines into it:

```
[VATSettings]
ExtensionDirectories[]=myextension
Handler=mysectionbased
RequireUserCountry=false
DynamicVatTypeName=Section based VAT
```

This will instruct eZ Publish to use the VAT handler located at "extension/myextension/vathandlers/mysectionbasedvathandler.php". Since the VAT percentage determined by this handler does not depend on user country, the "RequireUserCountry" setting must be set to false. Since this handler does not use the VAT rules, it is reasonable to display the name of the dynamic VAT type as "Section based VAT" (not "Determined by VAT charging rules"). This is done by using the "DynamicVatTypeName" setting.

4. To activate your extension in eZ Publish, log in to your eZ Publish administration interface, click on the "Setup" tab, and then click "Extensions" on the left. You will see the list of available extensions. Select the "myextension" item and click the "Apply changes" button.

4.11 Improved shipping handling

eZ Publish 3.8 makes it possible to define custom shipping options for your webshop (e.g. the cost of shipping may depend on the product properties). This could be done by implementing a *shipping handler* i.e. PHP class providing a mechanism that keeps shipping information for a product collection (basket or order) and calculates the cost of shipping for it. eZ Publish does not include any built-in shipping handlers so you will need to extend the system by creating your own shipping handler in order to add shipping options for your webshop. Using two or more shipping handlers at the same time is not supported (within one siteaccess). The shipping handler to use must be specified in the "Handler" INI setting described in the "INI settings" subsection.

When a user is viewing a product page, no shipping cost will be displayed and included into product price. After adding some products to the basket the system will calculate their shipping cost that will be shown under the list of items and included into order total. A shipping handler returns not only shipping cost but also shipping options summary and a link to the shipping management interface where shipping options can be modified. This information will be displayed in the basket together with the shipping cost. The system will also show shipping cost and shipping options summary when asking a user to confirm his/her order and in the order view interface for site administrators.

INI settings

The "[ShippingSettings]" section of the "settings/shop.ini" configuration file defines the shipping handler that will be used for calculating the cost of shipping for your products. Under this section, the following settings can be specified:

- The "Handler" setting specifies the shipping handler that will be used.
- The "RepositoryDirectories[]" array specifies the directories where eZ publish will search for built-in shipping handlers.
- The "ExtensionDirectories[]" array specifies the extension directories where eZ publish will search for additional shipping handlers. By default eZ publish will search in the "shipping-handlers" subdirectory inside your extension.

Example 1

The following lines can be specified under the "[ShippingSettings]" section of the "shop.ini" configuration file:

```
[ShippingSettings]
Handler=ezcustom
RepositoryDirectories[]=kernel/classes/shippinghandlers
```

These settings will instruct eZ Publish to use the shipping handler located at "kernel/classes/shippinghandlers/ezcustomshippinghandler.php".

Example 2

If you have an extension "myextension" that includes a shipping handler "mycost", you can put the following lines into an override for the "shop.ini" configuration file:

```
[ShippingSettings]
Handler=mycost
ExtensionDirectories[]=myextension
```

or

```
[ShippingSettings]
Handler=mycost
RepositoryDirectories[]=extension/myextension/shippinghandlers
```

These settings will instruct eZ Publish to use the VAT handler located at "extension/myextension/shippinghandlers/mycostshippinghandler.php".

Creating new shipping handlers

This section reveals some helpful tips for those developers who want to create a new shipping handler (only for people who are familiar with PHP). Please note that it is not recommended to modify the eZ Publish kernel and thus you should implement it as an extension.

Implementation details

A shipping handler is a file that contains a class implementing the following methods:

```
/**
 * Invoked to get shipping information for given product collection.
 * \public
 * \static
 */
function getShippingInfo( $productCollectionID );

/*
 * Invoked when shopping basket contents is changed
 * to update shipping info/cost appropriately.
 * \public
 * \static
 */
function updateShippingInfo( $productCollectionID );

/**
```

```

* Invoked when the associated product collection is removed
* to clean up shipping information.
* \public
* \static
*/
function purgeShippingInfo( $productCollectionID );

```

A handler is called via eZShippingManager class that has the same methods.

```
$shippingInfo = eZShippingManager::getShippingInfo( $productCollection );
```

All that getShippingInfo() method does is invoking getShippingInfo() method of the shipping handler that is specified in the "Handler" INI setting. This method returns shipping information for a given product collection as a hash containing the following elements:

Name	Type	Description
description	string	Shipping options summary.
cost	integer	Shipping cost for given set of products.
management_link	string	Link to the shipping management interface where shipping options can be modified.

The next subsection explains how you can implement your own shipping handler.

Creating your own handler

The following text describes the implementation of a trivial shipping handler for demonstration purposes.

1. Create the following subdirectories in the "extension" directory of your eZ Publish installation:
 - myextension
 - myextension/settings
 - myextension/shippinghandlers
2. Create a file called "mycostshippinghandler.php" in the "myextension/shippinghandlers/" directory (this file must contain a PHP class called "MyCostShippingHandler") and add the following lines into it:

```

<?php
class MyCostShippingHandler
{

```

```
function getShippingInfo( $productCollectionID )
{
    return array(
        'description' => 'Manual',
        'cost' => 10,
        'management_link' => '/shop/basket/' // dummy
    );
}
function purgeShippingInfo( $productCollectionID )
{
    // nothing to purge
}
function updateShippingInfo( $productCollectionID )
{
    // nothing to update
}
}
?>
```

3. Create a file called "shop.ini.append.php" in the "myextension/settings" directory and add the following lines into it:

```
[ShippingSettings]
Handler=mycost
ExtensionDirectories[]=myextension
```

4. To activate your extension in eZ Publish, log in to your eZ Publish administration interface, click on the "Setup" tab, and then click "Extensions" on the left. You will see the list of available extensions. Select the "myextension" item and click the "Apply changes" button.

This will make the system add the fixed cost of shipping to any set of products being purchased from your site.

If you need more complicated shipping options, you can try to use the advanced example from http://ez.no/community/contribs/examples/sample_shipping_handler or develop your own shipping handler.

4.12 LDAP Login Handler

The eZ LDAP login handler enables you to use an LDAP server as a repository for user and group information. Upon login, it will verify the user name and password against the LDAP server. If successful, it will create a copy of the LDAP user object in eZ Publish and depending on settings it may also create groups. The password however is not stored in the local copy, so password verification is always against the LDAP server.

It can be used in combination with other login handlers.

The login process

The following is a step by step walk-through of how the LDAP login process works, with information about the settings that are necessary for each step. This guide requires a basic understanding of LDAP terminology and configuration. Please visit the `ldap.ini` documentation page for information regarding each setting.

Log In

As the user submits his username and password in the login form, the user/login module view runs. It checks the `site.ini` `[UserSettings]` `LoginHandler[]`. The 'standard' handler verifies the username and password against the user data stored in eZ Publish. The 'LDAP' handler on the other hand verifies it against an LDAP server.

You can use multiple login handlers, which will be executed in sequence until one of them returns a valid login. The 'standard' handler should be set first in `site.ini`:

```
[UserSettings]
LoginHandler []
LoginHandler [] =standard
LoginHandler [] =LDAP
```

While the LDAP login handler executes, it checks the settings in `ldap.ini`. The `LDAPEnabled` setting must be set to true, otherwise the handler will exit. The brand, version and setup of your LDAP server dictates most of the first settings. These include `LDAPFollowReferrals`, `LDAPServer`, `LDAPPort` and `Utf8Encoding`, and `LDAPVersion` which should be 2 or 3.

Please visit the `ldap.ini` documentation page for information regarding each setting.

The handler now performs the first of two binds against the server ('bind' is an LDAP term for authenticating against a server). If `LDAPDebugTrace` is enabled, this step will be shown as step 1/5. If your LDAP server is configured to not accept anonymous binds (binds without username and password) you must set bind credentials in the `LDAPBindUser` and `LDAPBindPassword` settings.

Note that these credentials have nothing to do with the credentials of the user logging in. Instead, they are the credentials that must be supplied in order to read the LDAP user object we want to authenticate against.

Search for user object

Next, the handler performs a search for the user object given by the login and password provided by the user. If `LDAPDebugTrace` is enabled, this step will be shown as step 2/5. The starting position for the search is given by the `LDAPBaseDn` setting. The more specific your base DN is, the less data the handler has to search through.

Please note that all DN's and filters in `ldap.ini` must be entered without the equal sign. (This is due to a limitation in the ini file format.) Instead, you must use the string given in `LDAPEqualSign`, which defaults to `'--'`.

There are three kinds of searches, these are configured by the `LDAPSearchScope` setting, which can be `'sub'`, `'one'` or `'base'`.

The default is `'sub'`, which searches the whole subtree below the base DN. `'one'` searches one level only - the level immediately below the base DN. And `'base'` searches for only one object - meaning that the base DN must equal the DN of the object you want. (It can only be used in the very special case where there is only one LDAP user, presumably a system user of some kind, and all other users come from other sources.)

In case you want to limit the set of data to search through, you can configure a filter in the `LDAPSearchFilters` setting. You can search for a particular class using e.g. `'objectClass--inetOrgPerson'`, or a particular attribute using `'myattribute--myvalue'` (see note about `LDAPEqualSign`, above). You can configure multiple filters, they will be combined and used together. One filter is added automatically by the login handler - the filter for the user who is logging in (usually something like `'uid--johndoe'`). The user ID attribute is set in the `LDAPLoginAttribute` setting.

Authentication

If the search returns a valid user object, the handler performs the second bind, this time using the credentials of the user logging in. If `LDAPDebugTrace` is enabled, this step will be shown as step 3/5. If this bind succeeds, the user is authenticated.

Group Assignment

Next, the login handler tries to assign the user to one or more groups, based on the groups he belongs to in LDAP. If `LDAPDebugTrace` is enabled, this step will be shown as step 4/5. The location of groups created by the login handler is set in `LDAPGroupRootNodeId`. This must be the node ID of the group node that should be the parent of all created groups. By default, if the user has been moved to a different group on the LDAP server or in eZ Publish since the last login, the group assignment will be corrected in eZ Publish on the next login. If you rather want the eZ Publish group assignments to stay the same, set `KeepGroupAssignment` to enabled. The mapping between users and groups can be done in three different ways, depending on your LDAP server configuration, the setting is `LDAPGroupMappingType`, and the available modes are `'UseGroupAttribute'`, `'SimpleMapping'` and `'GetGroupsTree'`.

Storing user object

Finally, when the group assignment is complete, the user object is stored in eZ Publish. If LDAPDebugTrace is enabled, this step will be shown as step 5/5.

You must set LDAPFirstNameAttribute, LDAPLastNameAttribute and LDAPEmailAttribute. They should be set to the attributes holding the LDAP users first name, last name and email, respectively.

If the first name is not stored directly in LDAP, but can be extracted from the cn (common name), then set LDAPFirstNameAttribute=cn and LDAPFirstNameIsCommonName=true. The first name will then be set to common name minus last name. (This feature was added in version 4.1 beta 1.)

If email is not stored directly in LDAP, but can be composed by adding the login name and a static suffix, you can leave LDAPEmailAttribute empty, and set the suffix in LDAPEmailEmptyAttributeSuffix. If you set '@example.com' here, then a user with user ID 'johndoe' will get the email 'johndoe@example.com'. The user ID attribute is set in the LDAPLoginAttribute setting.

This concludes the login process.

4.12.1 LDAP Group Mapping Type

UseGroupAttribute

This mode requires that group membership is specified in the LDAP user object, i.e. the user object contains an attribute specifying the groups. When you use this, you must set LDAPUserGroupAttribute to the LDAP attribute that holds the group information. Also, LDAPUserGroupAttributeType must be either 'id' or 'name'.

If 'id' is used, then the LDAPUserGroupAttribute attribute must contain an ID (example: 42) that will match an existing eZ Publish group with the name 'LDAP 42'. If 'name' is used, then the LDAPUserGroupAttribute attribute must contain the name of an existing eZ Publish group. Starting with eZ Publish 4.3, there is a third option 'dn', meaning that the LDAPUserGroupAttribute attribute must contain a DN referring to the group the user belongs to. (Note that in LDAP, some attributes may be set multiple times, as opposed to having the same attribute containing multiple values.)

Some examples:

ldap.ini settings	LDAPGroupMappingType=UseGroupAttribute LDAPUserGroupAttributeType=id LDAPUserGroupAttribute=employeetype
LDAP user objects	uid: janedoe employeetype: 22 uid: johndoe employeetype: 22 employeetype: 42
Maps users to eZPublish groups	janedoe: 'LDAP 22' johndoe: 'LDAP 22', 'LDAP 42'

ldap.ini settings	LDAPGroupMappingType=UseGroupAttribute LDAPUserGroupAttributeType=name LDAPUserGroupAttribute=employeetype
LDAP user objects	uid: janedoe employeetype: Editors uid: johndoe employeetype: Editors employeetype: Technical writers
Maps users to eZPublish groups	janedoe: 'Editors' johndoe: 'Editors', 'Technical writers'

ldap.ini settings	LDAPGroupMappingType=UseGroupAttribute LDAPUserGroupAttributeType=dn LDAPUserGroupAttribute=employeetype
LDAP user objects	uid: janedoe employeetype:cn=Editors,dc=Groups,dc=example,dc=co uid: johndoe employeetype:cn=Editors,dc=Groups,dc=example,dc=co

	employeetype:cn="Technical writers",dc=Groups,dc=example,dc=com
Maps users to eZ	janedoe: 'Editors'
Publish groups	johndoe: 'Editors', 'Technical writers'

In eZ Publish 4.2 and older, all three LDAPUserGroupAttributeType alternatives require that the groups exist in eZ Publish, they will not be created by the login handler. However, starting with eZ Publish 4.3, if LDAPCreateMissingGroups is enabled, it will create groups as needed. This setting is disabled by default, for backwards compatibility. When not creating groups, the handler will fall back to using the default group.

SimpleMapping

This mode requires that group membership is specified in the LDAP group object, i.e. the group object contains an attribute specifying the users that are members. When you use this, you must set LDAPGroupClass, LDAPGroupNameAttribute, LDAPGroupMemberAttribute, and LDAPUserGroupMap.

LDAPGroupClass must be set to the class of LDAP group objects. LDAPGroupNameAttribute must be set to the attribute of the LDAP group that contains the name of the group. LDAPGroupMemberAttribute must be set to the attribute of the LDAP group that contains the names of the users that are members. LDAPUserGroupMap is a simple mapping from LDAP group names to eZ Publish group names. Example: 'LDAPUserGroupMap[myldapgroup]=myezgroup'. You may also set LDAPGroupBaseDN and LDAPGroupDescriptionAttribute. LDAPGroupBaseDN may be set to the base DN of your groups tree, to limit the amount of data to search through. LDAPGroupDescriptionAttribute may be set to the attribute of the LDAP group that contains the description of the group, if any. This will set the same description in the eZ Publish group. This mode requires that the groups exist in eZ Publish, because they will not be created automatically. This allows the eZ Publish groups to have different names than the corresponding LDAP groups.

The LDAPCreateMissingGroups setting does not apply in this mode.

An example:

ldap.ini settings	LDAPGroupMappingType=SimpleMapping LDAPUserGroupClass=organizationalUnit LDAPUserGroupAttribute=cn LDAPGroupMemberAttribute=members LDAPUserGroupMap[] LDAPUserGroupMap[editor]=Editor LDAPUserGroupMap[techwriter]=Technical writer
LDAP group objects	cn: editor members: janedoe members: johndoe cn: techwriter members: johndoe
Maps users to eZ Publish groups	janedoe: 'Editors'

johndoe: 'Editors', 'Technical writers'

GetGroupsTree

This mode requires that group membership is specified in the LDAP group object, i.e. the group object contains an attribute specifying the users that are members. When you use this, you must set LDAPGroupClass, LDAPGroupNameAttribute and LDAPGroupMemberAttribute.

LDAPGroupClass must be set to the class of LDAP group objects. LDAPGroupNameAttribute must be set to the attribute of the LDAP group that contains the name of the group. LDAPGroupMemberAttribute must be set to the attribute of the LDAP group that contains the names of the users that are members. You may also set LDAPGroupBaseDN and LDAPGroupDescriptionAttribute. LDAPGroupBaseDN may be set to the base DN of your groups tree, to limit the amount of data to search through. LDAPGroupDescriptionAttribute may be set to the attribute of the LDAP group that contains the description of the group, if any. This will set the same description in the eZ Publish group. This mode will create groups in eZ Publish automatically. The groups will be given the same name in eZ Publish as they have in LDAP.

An example:

ldap.ini settings	LDAPGroupMappingType=GetGroupsTree LDAPUserGroupClass=organizationalUnit LDAPUserGroupAttribute=cn LDAPGroupMemberAttribute=members
LDAP group objects	cn: editor members: janedoe members: johndoe cn: techwriter members: johndoe
Maps users to eZ Publish groups	janedoe: 'editors' johndoe: 'editors', 'techwriter'

Default group

The default LDAP group is not created automatically, you should create one and call it e.g. 'LDAP Users'. If the group mapping fails, the user will instead be placed in the default group, as specified in the LDAPUserGroupType and LDAPUserGroup[] settings.

LDAPUserGroupType can be either 'id' or 'name'. If 'id' is used, LDAPUserGroup[] must contain the content object ID of the default group. If 'name' is used, LDAPUserGroup[] must contain the name of the default group. The default group is not used when group mapping succeeds.

4.12.2 Roles and Settings

Roles

The login handler handles users and groups, it will not create or assign any roles. Since there is no standard way to automate this function, this is the responsibility of the eZ Publish system administrator. Users and groups created by the login handler will only have the roles they inherit from their parent groups. If the roles they inherit are too restrictive, or they don't inherit any, such users may not be able to log in even when the LDAP authentication succeeded. Therefore it is recommended to create and assign a basic role with login rights to the LDAP root group (see `LDAPGroupRootNodeId`). This also applies to the default group (see `LDAPUserGroup[]`) if you want users to be able to login even when the group assignment failed.

In addition to this, it is common to make additional roles for each of the sub groups of the root group, granting the necessary permissions for each group.

Settings

LDAP login is configured in `ldap.ini`, for more information see descriptions in the settings file itself and the documentation regarding configuration files. In addition to this, LDAP must be enabled in the `LoginHandler` setting in `site.ini` (see examples in chapter [LDAPGroupMappingType](#)).

4.12.3 Enhancements

The LDAP user manager cronjob

The cronjobs/ldapusermanage.php script synchronizes data from the LDAP server to eZ Publish. But this is no longer needed, as [issue #15530](#) has been fixed and data is now synchronized with every login. The cronjob is only needed if you don't have the fix for #15530 yet, and user data has been changed in LDAP or eZ Publish. The script is likely to be deprecated and removed soon, and therefore it is recommended to avoid using it.

An alternative way (though not recommended) of synchronizing user objects is to delete them, since they will be recreated on the next login with synchronized data. Please note that this solution can be destructive - any object relations, object ownership information, and sub nodes of the user node will not be recreated.

LDAP login handler improvements ([Issue #15490](#))

As mentioned before (see LDAP Group Mapping Type: UseGroupAttribute), it will be possible to set the LDAPUserGroupAttributeType to "dn" in eZ Publish 4.3. When LDAPUserGroupAttributeType is set to "dn", the LDAPUserGroupAttribute should be set to an LDAP attribute that holds the DN of the group(s) to which the user belongs. If the user belongs to multiple groups, then this attribute should be set multiple times in the LDAP user object - it should not contain multiple DNs (This is how LDAP attributes are normally used). The 'dn' value comes in addition to the existing allowed values 'name' and 'id', which are not changed.

Also UseGroupAttribute mode can now create groups. Previously when LDAPGroupMappingType was set to "UseGroupAttribute", no groups would be created. If the indicated group(s) were not found, the user(s) would be placed in the default group. With the addition of the LDAPCreateMissingGroups setting, the creation of groups is now supported. This setting is disabled by default, for backwards compatibility. When it is enabled, missing groups will be created. This enhancement will be implemented in eZ Publish 4.3.0

Bug fixes

[#15530](#): Existing eZ Publish users are not synchronized with LDAP users

When using the LDAP login handler, the following problems affect LDAP users that exist in eZ Publish:

- If the user data is changed on the LDAP server, this change is not reflected in the eZ Publish user object
- If the user data of the eZ Publish user object is changed, this change remains after next login.

In other words, existing eZ Publish users are not synchronized with data from LDAP. This should occur with every login for the user that logs in.

Fixed in: 4.3.0, 4.2.1, 4.1.5, 4.0.8

#15485: Deleted LDAP user nodes are not recreated

When a LDAP user has multiple nodes and one of them is removed, it will not be recreated on the next login. On the other hand, when all the nodes are removed (meaning that the object is also gone) then all nodes are correctly recreated.

Fixed in: 4.3.0, 4.2.1, 4.1.5, 4.0.8

#14389: In ezldapuser.php, LDAPLoginAttribute and LDAPGroupNameAttribute ini variables with upper case characters unread

When LDAPLoginAttribute and LDAPGroupNameAttribute contain upper case characters the login fails.

Fixed in: 4.3.0, 4.2.1, 4.1.5, 4.0.8

Troubleshooting

The LDAPDebugTrace setting was added in version 4.1 beta 1. When this is enabled, it will write LDAP login data to notice.log. This can help you figure out a configuration problem.

4.13 Multi-currency

The purpose of this section is to introduce and describe the multi-currency feature available in eZ Publish 3.8. People previously unfamiliar with eZ Publish webshop subsystem should read the "Webshop" section of the "Concepts and basics" chapter first. The next sections will help you to understand the following issues:

- The concept of custom prices and auto prices
- What the base custom price is
- How to specify your own price rounding criteria for auto prices
- How the currency rates can be used (auto rates and custom rates)
- What the base currency is
- How to manage your currencies
- What the preferred currency is
- How to use additional view templates for multi-price products
- The purpose of the "Products overview" interface
- How to use the default exchange rates update handler
- How to create your own handler for rates updating
- How to convert all your products to multi-price format

4.13.1 Custom prices and auto prices

The multi-price datatype allows you to set prices in multiple currencies for each product. If you use for example five currencies then a product will always have five prices. However, you don't have to enter all these prices manually although it is possible. It is required that you specify at least one price per product which is called *base custom price* (*base price* for short). The system will automatically convert it using the appropriate rates (page 469) in order to calculate prices in other currencies (the rest four prices in our example). These are called *auto prices*. In contrast to these, prices that are specified manually are called *custom prices*.

Custom prices are fully independent of the currency rates (page 469). By saying "base custom price" we mean a special custom price that is used for calculating the auto prices. All other custom prices will be called *non-base*. Please note that non-base custom prices are independent of the base price. If you change the base price, the system will automatically update all the auto prices for this product but not the custom prices. If you change a non-base custom price, no automatic updates will be done.

The auto prices are usually marked as "(Auto)" in the object edit interface while the custom prices aren't. There is no special mark for a base custom price because this value usually comes right after the auto prices and right before 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.

If you remove a non-base custom price, the system will create an auto price in this currency.

If you remove the base price, the system will do the following:

- Set the earliest non-base custom price as a new base price.
- Remove the old base price and create a new auto price instead.
- Update all the auto prices in accordance with the new base price.

It is recommended that each product has at least one custom price. If you remove all the custom prices for a product, the system will use zero auto prices in all currencies for this product.

Example

Let's say that you use three currencies with the rates indicated in the following table.

Currency code	Currency rate
NOK	1.32015
EUR	0.16380
USD	0.19500

If you specify for example \$50 as the base price, the system will automatically calculate two auto prices for this product (look at the next screenshot).

(see figure 4.88)

Price (required):

Currency	Value
<input type="checkbox"/> EUR	42.00(Auto)
<input type="checkbox"/> NOK	338.50(Auto)
<input type="checkbox"/> USD	<input type="text" value="50.00"/>

Remove selected

NOK Set custom price

Figure 4.88: The base price in USD and two auto prices.

These auto prices are calculated by converting \$50 to Euro and Norwegian krone:

- EUR/USD cross rate = $0.16380 / 0.19500 = 0.84000$
- price in EUR = $50 * 0.84 = 42.00$
- NOK/USD cross rate = $1.32015 / 0.19500 = 6.77000$
- price in NOK = $50 * 6.77 = 338.50$

If you think that this product costs much more in some particular countries, for example in Norway, you can set the desired price by creating a custom price in NOK. The result is shown in the following screenshot.

(see figure 4.89)

Price (required):

Currency	Value
<input type="checkbox"/> EUR	42.00(Auto)
<input type="checkbox"/> USD	<input type="text" value="50.00"/>
<input type="checkbox"/> NOK	<input type="text" value="600.00"/>

Remove selected

Figure 4.89: The base price in USD, non-base custom price in NOK and auto price in EUR.

As you can see from the screenshot above, there are two custom prices (base \$50 and non-base 600nok) and one auto price (42). Please note that you can always remove non-base custom prices, so that the system will automatically set auto prices instead.

If you remove the base price (\$50), the non-base custom price (600 NOK) will become new base price so the system will automatically update the auto prices as shown in the screenshot below.

(see figure 4.90)

These auto prices are calculated by converting 600 nok to Euro and US Dollars:

- EUR/NOK cross rate = $0.16380 / 1.32015 = 0.12408$
- price in EUR = $600 * 0.12408 = 74.45$

Price (required):

Currency	Value
<input type="checkbox"/> EUR	74.45(Auto)
<input type="checkbox"/> USD	88.63(Auto)
<input type="checkbox"/> NOK	<input type="text" value="600.00"/>

Remove selected

Figure 4.90: The results of removing the base custom price.

- USD/NOK cross rate = $0.19500 / 1.32015 = 0.15$
- price in NOK = $600 * 0.15 = 88.63$

4.13.2 Rounding auto prices

The price rounding process is affected by settings which are specified in the "[MathSettings]" section of the "settings/shop.ini" configuration file such as:

- RoundingPrecision
- RoundingType
- RoundingTarget

You can specify your own price rounding criteria for auto prices calculation by creating an override for this configuration file.

RoundingPrecision

This setting specifies how many significant digits after the decimal point should be kept while rounding. By default, the precision is set to 2. Normally, there is no need to set for example "RoundingPrecision=3" because only two decimal digits are stored in the database for each price.

RoundingType

This setting defines which rounding method should be used. The possible values are described in the following table.

Setting	Description	Actual value	Rounded value
RoundingType=round	returns the closest value	0.124	0.12
RoundingType=round	returns the closest value	0.125	0.13
RoundingType=ceil	returns the next highest value by rounding up	0.121	0.13
RoundingType=floor	returns the next lowest value by rounding down	0.129	0.12
RoundingType=none	rounding is not used	1/3	*0.333333...

* as long as only two decimal digits are stored in the database for each price, the result will be 0.33

Please note that the examples above are calculated supposing that RoundingPrecision is set to 2. The default value of the "RoundingType" setting is "round".

RoundingTarget

This setting allows you to force rounding to the specified target. For example, if you prefer "retail" prices like \$2.49 instead of \$2.50, you can instruct the system to use 9 as the end digit for all your auto prices.

The default value of the "RoundingTarget" is "false".

Please refer to the following table for examples of usage.

Setting	Actual value	Rounded value
RoundingTarget=false	89.468543	89.47
RoundingTarget=5	89.468543	89.45
RoundingTarget=99	89.468543	89.99

Please note that the examples above are calculated supposing that RoundingPrecision is set to 2 and RoundingType is set to "round".

Don't forget to update auto prices for existing products after changing the rounding settings.

4.13.3 Currency rates

There are two types of currency rates:

- Auto rates
- Custom rates

Auto rates

Auto rates are retrieved via automatic update of the exchange rates from the external source. It is possible to get these rates from the website of the European Central Bank using the built-in "eZECB" handler or to extend the system by creating your own update handler. Please note that you should specify the desired handler in the "ExchangeRatesUpdateHandler" INI setting described in the "Exchange rates update handlers (page 488)" section, otherwise the system will not be able to update auto rates.

An auto rate of the currency is nothing more than the amount of this currency that must be given up in order to obtain one unit of the *base currency*. The base currency is determined by the "BaseCurrency" INI setting described in the "Exchange rates update handlers (page 488)" section. It is recommended (but not required) that you specify one of the existing currencies in this setting.

The website of the European Central Bank allows to get the currency exchange rates relative to EUR. If you use the "eZECB" handler and set for example USD as the base currency, the system will retrieve the exchange rates relative to EUR and then calculate the auto rates relative to USD. Please note that you will get an error if the exchange rate for USD was not retrieved.

Example

Let's say that the following rates relative to EUR are indicated on the website of the European Central Bank:

Currency	Rate
NOK	7.85620
USD	1.20940
UAH	not available

If you specify EUR as the base currency, the system will set the following auto rates:

Currency	Auto rate
EUR	1.00000
NOK	7.85620
USD	1.20940
UAH	N/A

If you do not specify the base currency, the result will be the same.

If you specify USD as the base currency, the system will calculate the auto rates relative to USD as shown in the following table:

Currency	Auto rate
EUR	0.82685 (1 / 1.20940)
NOK	6.49594 (7.85620 / 1.20940)
USD	1.00000
UAH	N/A

If you specify UAH as the base currency, the system will display the following error message: "Unable to calculate cross-rate for currency-pair EUR/UAH" when trying to update the auto rates. Since the exchange rate for UAH was not retrieved from the website of the European Central Bank, the system will not be able to calculate the rates relative to UAH.

Custom rates

You can specify a fixed *custom rate* for a currency so that this value will be used instead of the auto rate value (the auto rate will not be used and thus it will be displayed in gray color). Please note that the custom rates must be relative to the same base currency as the auto rates. If you have for example five currencies with auto rates relative to EUR and you wish to specify a custom rate for one of them, make sure this rate is also relative to EUR.

If you are going to specify custom rates for all your currencies without exception, you may use some other currency as the base one. It is recommended (but not required) that you use one of the existing currencies as the base currency.

Example

Let's say that you have four currencies: USD, EUR, NOK and UAH. You can consider USD as the base currency and specify the custom rates relative to this currency, for example:

- 1 for USD
- 0.84 for EUR
- 6.52 for NOK
- 5.05 for UAH

If you then remove the "USD" currency, the custom rates for EUR, NOK and UAH will not be changed.

4.13.4 Creating a new currency

The administration interface allows you to add new currencies to the webshop system. Let's say that you already have three currencies (USD, UAH and NOK) and you wish to add another one (EUR). The following example demonstrates how to add EUR when you already have USD, UAH, NOK.

1. Click the "Webshop" tab in the administration interface, select the "Currencies" link on the left and click the "New currency" button located under the list of existing currencies. (This interface can also be accessed by requesting "/shop/currencylist" in the URL.) (see figure 4.91)

Name	Code	Symbol	Locale	Status	Auto rate	Custom rate	Factor	Rate
Norwegian Krone	NOK	kr	nor-NO	Active	N/A	6.77000	1.00000	6.77000
Ukrainian hryvnia	UAH	грн	ukr-UA	Active	N/A	5.05000	1.00000	5.05000
U.S. dollar	USD	\$	eng-US	Active	N/A	1.00000	1.00000	1.00000

Figure 4.91: The list of available currencies.

The system will bring up the currency edit interface where you can specify the desired properties for a new currency (look at the next screenshot). (see figure 4.92)

Create currency

Currency code: (Use three capital letters)

Currency symbol:

Formatting locale:

Custom rate:

Rate factor:

Figure 4.92: The currency edit interface.

2. Specify the currency attributes (these are described below) and click the "Create" button. The system will add a new currency as shown in the screenshot below. (see figure 4.93)

Please note that after creating a new currency the system will automatically create zero auto prices in this currency for all your products. It is recommended to click the "Update autoprices" button when you have finished managing your currencies. This will instruct the system to update auto prices for all products.

Available currencies									
Name	Code	Symbol	Locale	Status	Auto rate	Custom rate	Factor	Rate	
<input type="checkbox"/> European euro	EUR	€	eng-GB@euro	Active	N/A	0.84000	1.00000	0.84000	
<input type="checkbox"/> Norwegian Krone	NOK	kr	nor-NO	Active	N/A	6.77000	1.00000	6.77000	
<input type="checkbox"/> Ukrainian hryvnia	UAH	грн	ukr-UA	Active	N/A	5.05000	1.00000	5.05000	
<input type="checkbox"/> U.S. dollar	USD	\$	eng-US	Active	N/A	1.00000	1.00000	1.00000	

Figure 4.93: The list of available currencies.

Currency code

The three-character currency code which is generally used to represent this currency ("USD", "EUR" and so on). This parameter is required. This code can be thought of as an unique identifier of the currency. You can not use two currencies with the same codes. The currency code consists of three English capital letters and often (but not always) corresponds to the ISO 4217 standard.

Once the currency code is specified, the system will be able to display the currency name ("European euro", "U.S. dollar" and so on). These currency names can be changed by providing a custom version of the "currencynames.tpl" template which is located in the "templates/shop/" directory of the standard design. This template does not have any effect on the shop functionality available for site visitors. The currency names are displayed only in the administration interface. If you have created a new currency with unknown code for example "ABC" which is not listed in the "currencynames.tpl" template, the system will display the currency name as "Unknown currency name".

Currency symbol

A *currency symbol* is a string that will be displayed near the numerical price value ("\$", "" and so on). Currency symbols are used in everyday life to denote that a number is a monetary value. This parameter is not required. If the currency symbol is not defined, the visitors who prefer using this currency will see the numerical price values without any additional symbols. Please note that if you are not able to type in the desired symbol then you can copy and paste it from your browser or text editor.

Formatting locale

A *formatting locale* is a locale which is used for price formatting. This parameter is required. You can choose the desired locale from the drop-down list of available locales. By default, the current system locale is selected (this locale is determined by the "Locale" setting located in the "[RegionalSettings]" section of the "settings/site.ini" configuration file or its override). The

available locales and their settings are defined by the locale INI files located in the "share/locale" directory of your eZ Publish installation.

Once the formatting locale is specified, the system will automatically format the prices using the "DecimalSymbol", "ThousandsSeparator", "FractDigits" and "PositiveFormat" settings specified in the "[Currency]" section of the locale's INI file. Please note that the "Symbol", "Name" and "ShortName" settings defined in the same section will not have any effect in this case.

Example

Let's create a new currency "ABC" and specify its properties as shown in the following screenshot.

(see figure 4.94)

Figure 4.94: *The currency edit interface.*

Since the "ABC" code is not listed in the "currencynames.tpl" template, the system will display the currency name as "Unknown currency name" (look at the next screenshot).

(see figure 4.95)

Name	Code	Symbol	Locale	Status	Auto rate	Custom rate	Factor	Rate	
Unknown currency name	ABC	abc	eng-US	Active	N/A	1.11000	2.00000	2.22000	
European euro	EUR	€	eng-GB@euro	Active	N/A	0.84000	1.00000	0.84000	
Norwegian Krone	NOK	kr	nor-NO	Active	N/A	6.77000	1.00000	6.77000	
Ukrainian hryvnia	UAH	грн	ukr-UA	Active	N/A	5.05000	1.00000	5.05000	
U.S. dollar	USD	\$	eng-US	Active	N/A	1.00000	1.00000	1.00000	

Figure 4.95: *Unknown currency name in the list of currencies.*

This problem can be solved by creating a custom version of the "currencynames.tpl" which is

located in the "templates/shop/" directory of the standard design. To do this, copy the "currency_names.tpl" template into the "templates/shop/" directory of the admin design and edit it. Add a new key/value pair to the list of pairs that are passed to the "hash" template operator which creates the "set_currency_names" associative array as shown below:

```
{set currency_names = hash( 'ABC', 'AB-Currency',
'AUD', 'Australian dollar',
...
'USD', 'U.S.dollar' ) }
```

After clearing the eZ Publish caches, the system will display the currency name as "AB-Currency".

The "eng-US.ini" configuration file located in the "share/locale" directory contains the following section:

```
[Currency]
Symbol=$
Name=US Dollar
ShortName=USD
DecimalSymbol=.
ThousandsSeparator=,
FractDigits=2
PositiveSymbol=
NegativeSymbol=-
PositiveFormat=%c%p%q
NegativeFormat=%c%p%q
```

Since the "eng-US" locale is selected for the "ABC" currency, the system will use "." as a decimal symbol and "," as thousands separator, with 2 digits after decimal point and the currency symbol placed before the numeric value as specified in the "DecimalSymbol", "ThousandsSeparator", "FractDigits" and "PositiveFormat" settings. (The "Symbol", "Name" and "ShortName" settings will not be used.)

Let's say that some product costs for example 550 units in this currency. In this case, the visitors who prefer using this currency will see the price of this product like this:

```
abc550.00
```

Custom rate

This required parameter tells the system about which rate to use for calculating auto prices (page [464](#)) for/in this currency. By default, the custom rate is set to 0 so the system will use auto rate for this currency. However, it is possible to specify a non-zero fixed custom rate value that will be used for calculating auto prices in this currency.

Rate factor

This required parameter is intended for supporting a kind of virtual rate that can be used for calculating auto prices (page 464) in this currency. If a non-zero custom rate is specified, the system will multiply it by rate factor in order to calculate the final rate, otherwise the system will multiply the auto rate by this factor. The default value of the rate factor is 1. The following table reveals how the final rate is calculated according to the auto rate, custom rate and rate factor values.

Custom rate	Rate factor	Auto rate	Final rate
0	1	0.85	0.85
0	1.4	0.85	$0.85 \times 1.4 = 1.19$
0.75	1	0.85	0.75
0.75	1.4	0.85	$0.75 \times 1.4 = 1.05$

Status

The status of the currency can be either "active or "inactive". When you create a new currency, the status will be automatically set to "Active". Inactive currencies will be invisible for the site visitors. In other words, you can hide a currency from your customers if you don't wish them to use this currency.

Inactive currencies are displayed in red color in the list of currencies as shown in the following screenshot.

(see figure 4.96)

Name	Code	Symbol	Locale	Status	Auto rate	Custom rate	Factor	Rate
European euro	EUR	€	eng-GB@euro	Active	N/A	0.84000	1.00000	0.84000
Norwegian Krone	NOK	kr	nor-NO	Inactive	N/A	6.77000	1.00000	6.77000
U.S.dollar	USD	\$	eng-US	Active	N/A	1.00000	1.00000	1.00000

Buttons: Remove selected, New currency, Update auto rates, Update autoprices, Apply changes

Figure 4.96: Displaying inactive currency in the list of currencies.

4.13.5 Editing a currency

The administration interface allows you to edit currencies. The following text reveals how this can be done.

1. Open the list of currencies by clicking the "Webshop" tab in the administration interface and selecting the "Currencies" link on the left. Find the target currency in the list and click the "Edit" button for this currency. You will be taken to the currency edit interface that allows to modify the following attributes (these are described in the previous section):
 - Currency code
 - Currency symbol
 - Formatting locale
 - Custom rate
 - Rate factor
2. Specify the desired currency attributes.
3. Click the "Store changes" button.

Changing the currency status

The currency edit interface does not support changing the statuses of the currencies. The following text reveals how you can change the status for one or more currencies.

1. Open the list of currencies by clicking the "Webshop" tab in the admin interface and selecting the "Currencies" link on the left.
2. Choose the desired status value from the drop-down list in the "Status" column (this can be done for several currencies).
3. Click the "Apply changes" button to save your changes.

Changing rates for multiple currencies

You can update currency rates and/or rate factors for several currencies at the same time using the list of currencies.

Updating auto rates

To update auto rates, do the following:

1. Open the list of currencies by clicking the "Webshop" tab in the administration interface and selecting the "Currencies" link on the left.

- If the "Update auto rates" button is inactive, this means that the system can not update auto rates because no update handler is specified. The following screenshot shows the situation when the custom rates are specified for all currencies and therefore the system will not use the auto rates. In this case, the auto rates are displayed in gray color. Since no updates were performed, the "N/A" marks are displayed in the "Auto rate" column. (see figure 4.97)

Name	Code	Symbol	Locale	Status	Auto rate	Custom rate	Factor	Rate
European euro	EUR	€	eng-GB@euro	Active	N/A	0.84000	1.00000	0.84000
Norwegian Krone	NOK	kr	nor-NO	Active	N/A	6.77000	1.00000	6.77000
U.S.dollar	USD	\$	eng-US	Active	N/A	1.00000	1.00000	1.00000

Figure 4.97: The list of currencies with disabled possibility to update auto rates.

- Since no update handler is specified, the "Update auto rates" button is inactive. To activate this button, you should choose the desired handler as described in the "Exchange rates update handlers (page 488)" section and clear the eZ Publish caches. The "Update auto rates" button will become active.
- Click the "Update auto rates" button. The auto rates will be automatically updated. (see figure 4.98)

Name	Code	Symbol	Locale	Status	Auto rate	Custom rate	Factor	Rate
European euro	EUR	€	eng-GB@euro	Active	1.00000	0.84000	1.00000	0.84000
Norwegian Krone	NOK	kr	nor-NO	Active	7.97700	6.77000	1.00000	6.77000
U.S.dollar	USD	\$	eng-US	Active	1.20960	1.00000	1.00000	1.00000

Figure 4.98: The list of currencies with updated auto rates.

This screenshot shows the situation when the auto rates are updated using EUR as the base currency (the EUR auto rate value is 1.00000). As you can see, the retrieved auto rates are still displayed in gray color because the custom rates are available. However, you can remove the custom rates in order to enable the auto rates.

Enabling auto rates

An auto rate for a currency will not be used if a non-zero custom rate is specified. If you remove the custom rate or set it to 0, the auto rate will be enabled. For example, if you remove all the values from the "Custom rate" column and click the "Apply changes" button, the system will start to use auto rates (look at the next screenshot).

(see figure 4.99)

Available currencies									
Name	Code	Symbol	Locale	Status	Auto rate	Custom rate	Factor	Rate	
<input type="checkbox"/> European euro	EUR	€	eng-GB@euro	Active	1.00000		1.00000	1.00000	
<input type="checkbox"/> Norwegian Krone	NOK	kr	nor-NO	Active	7.97700		1.00000	7.97700	
<input type="checkbox"/> U.S.dollar	USD	\$	eng-US	Active	1.20960		1.00000	1.20960	

Figure 4.99: The list of currencies with removed custom rates.

Please note that changing the rates will not result in updating auto prices for your products. It is recommended to click the "Update auto-prices" button when you have finished managing your currencies. This will instruct the system to update auto prices for all products.

Changing custom rates and/or rate factors

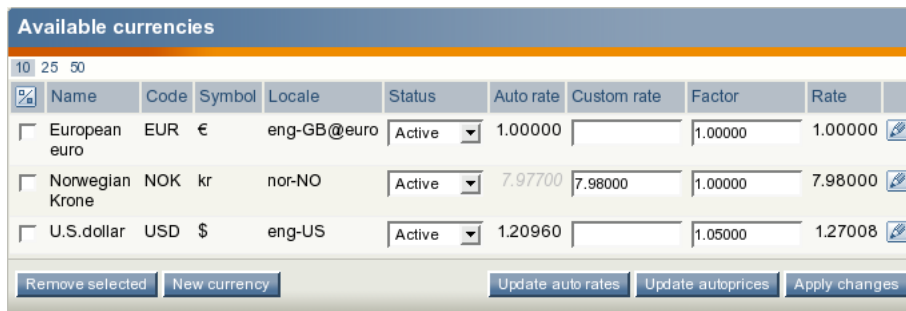
It is possible to change custom rates and/or rate factors for several currencies at the same time. The following text reveals how this can be done.

1. Open the list of currencies by clicking the "Webshop" tab in the administration interface and selecting the "Currencies" link on the left.
2. Set the desired custom rate in the "Custom rate" column and/or specify the desired rate factor value in the "Factor" column (this can be done for several currencies).
3. Click the "Apply changes" button to save your changes. The system will automatically re-calculate the final rates indicated in the "Rate" column.

(see figure 4.100)

The screenshot above shows the list of currencies after changing the NOK custom rate and the USD rate factor.

Please note that clicking the "Apply changes" button will not result in updating auto prices for your products. It is recommended to click the "Update auto-prices" button when you have finished managing your currencies. This will instruct the system to update auto prices for all products.



	Name	Code	Symbol	Locale	Status	Auto rate	Custom rate	Factor	Rate	
<input type="checkbox"/>	European euro	EUR	€	eng-GB@euro	Active	1.00000		1.00000	1.00000	
<input type="checkbox"/>	Norwegian Krone	NOK	kr	nor-NO	Active	7.97700	7.98000	1.00000	7.98000	
<input type="checkbox"/>	U.S.dollar	USD	\$	eng-US	Active	1.20960		1.05000	1.27008	

Remove selected New currency Update auto rates Update autoprices Apply changes

Figure 4.100: *The list of currencies with one custom rate.*

4.13.6 Removing a currency

It is possible (but not recommended) to remove currencies from the webshop system. If you need to hide some currency from your customers, you should set its status to "inactive" instead of removing it from the system.

Please note that removing a currency will result in removing prices in this currency for all products. This may cause problems if some of your products have a base price in this currency.

Example

Let's say that some of your products have base prices in USD as shown in the following table.

	Product 1	Product 2		
USD	50.00	Base custom price	50.00	Base custom price
NOK	338.50	Auto price	600.00	Non-base custom price
EUR	42.00	Auto price	42.00	Auto price

These base prices will be deleted if you remove the USD currency (see the next table).

	Product 1	Product 2		
NOK	0.00	Auto price	600.00	Base custom price
EUR	0.00	Auto price	74.45	Auto price

As you can see from the table above, removing the base price may result in unwanted behavior like setting all the product prices to zero. That is why removing currencies is not recommended.

The following text reveals how you can remove one or more currencies from the webshop system.

1. Open the list of currencies by clicking the "Webshop" tab in the administration interface and selecting the "Currencies" link on the left.
2. Use the check-boxes to select the currencies that you wish to remove.
3. Click the "Remove selected" button.

4.13.7 Preferred currency

A user can select one of the active currencies as "preferred currency". The system will then use this currency for the user. This can be done by requesting "shop/preferredcurrency" in the URL, choosing the desired currency from drop-down list and clicking the "Set" button. There is an additional possibility to set preferred currency by requesting "shop/setpreferredcurrency/(currency)/NOK" in the URL (you should replace "NOK" with the desired currency code). You can either create links for different currencies on your site or add a special toolbar as described below.

If the preferred currency is not specified, the system will use the default value specified by the "PreferredCurrency" setting in the "[CurrencySettings]" section of the "settings/shop.ini" configuration file. It is strongly recommended that you specify one of the active currencies in this setting.

It is possible to display only price in preferred currency to your customers when they are viewing multi-price products (please refer to the "Templates for viewing multi-price products" section for more information). Note that if you do not specify one of the existing currencies in the "PreferredCurrency" INI setting, the system will display zero prices to the first-time visitors of your site.

Example

Let's say that you have two currencies: EUR, NOK and the "settings/shop.ini" configuration file (or an override configuration file) contains the following lines:

```
[CurrencySettings]
PreferredCurrency=USD
```

If a user visits your site for the first time, the system knows nothing about his preferred currency and thus it will try to use the default value. However, the "USD" currency is not defined in your webshop system so there are no prices in this currency. The system will display zero prices using the currency symbol taken from your locale settings.

Adding a toolbar for customers

You can add a possibility for site visitors to change their preferred currency "on-the-fly" using the "Preferred currency" toolbar. To do this, add the following line into the "[Toolbar_right]" section of the "settings/siteaccess/example/toolbar.ini.append.php" file where "example" is your siteaccess name:

```
Tool []=preferred_currency
```

This setting instructs the system to display the toolbar which is determined by the "preferred_currency.tpl" template located in the "templates/toolbar/full" directory of the standard design.

Preferred currency for site administrators

A site administrator can choose the preferred currency by requesting "shop/preferredcurrency" in the URL, choosing the desired currency from drop-down list and clicking the "Set" button. This interface can also be accessed by clicking the "Webshop" tab and selecting the "Preferred currency" link on the left. The selected currency will be used for displaying prices in the products overview (page [487](#)) interface.

4.13.8 Multi-price products

An actual product is represented by a content object (with at least one node assignment) that contains information about the product itself along with a price. The price can be represented by an attribute that makes use of the built-in price or multi-price datatype. These are special datatypes which plug more deeply into the system and connect content objects with the webshop system. The main difference is that the price datatype allows to specify only one price value for each object (simple price product) whereas the multi-price datatype makes it possible to specify several price values in different currencies for each object (multi-price product). Please note that simple price products are incompatible with multi-currency feature.

A content class can only contain one price attribute or one multi-price attribute. There is no way to have a simple price product and a multi-price one in the shopping basket at the same time and it is not recommended to use both price and multi-price datatype on your site.

If you are going to use multi-price products, you should create at least one content class containing an attribute of the multi-price datatype as described in the next subsections. Instances/objects of this class will be treated as multi-price products. If you already have simple price products, you can automatically convert these to multi-price products as described in the "Upgrading your webshop" section.

Creating a product class

Access the "Setup" tab in the administration interface, click "Classes" on the left, select the "Content" class group and click the "New class" button located in the bottom of the list. You will be taken to the class edit interface as shown in the screenshot below.

(see figure 4.101)

Specify name, identifier, object name pattern and container flag for the newly created class and add the desired attributes using the drop-down list located in the bottom of the class edit interface.

Multi-price attribute

To add an attribute of the multi-price datatype, select the desired datatype from this list, click the "Add attribute" button and edit the newly added attribute (see the screenshot below).

(see figure 4.102)

It is recommended to specify "price" as the identifier (this value is used in the additional view templates). You have to select one of the predefined currencies as "default currency". This currency will be used for custom prices by default.

Example

Let's say that there are four predefined currencies: NOK, EUR, USD, UAH, and you are creating a class called "Products" with a multi-price attribute. If you set EUR as "default currency", the

Figure 4.101: *The class edit interface for a product class.*

Figure 4.102: *Class attribute edit interface for the "Multi-price" datatype.*

system will create a base price in EUR and auto prices in NOK, USD, UAH for each new object of this class. When a new product is created, the system will set this base price to 0.00 but you can specify the desired value instead (for example 50). It is also possible to remove this price and create a new base price in some other currency (for example \$60).

After adding the attributes, click "OK" to save the class.

Please note that if you need several different structures for storing info about your products, you can create several multi-product classes. If you sell for example computer hardware, you may need several content classes called "Monitors", "Printers", "Scanners" and so on. In this case, you will be able to filter products by class name in the products overview interface.

Creating a product

If you have a content class with multi-price datatype, you can create objects of this class i.e. your multi-price products. Please refer to the "Adding content" chapter of the "User manual" for more information about adding content objects.

Templates for viewing multi-price products

By default, the system will display prices in all currencies to a user. This is determined by the default "ezmultiprice.tpl" template located in the "templates/content/datatype/view/" directory of the standard design.

If you wish to display only price in the preferred currency, you can use the "multiprice.tpl" template located in the "override/templates/datatype/" directory of the base design. To do this, add the following lines to the "override.ini.append.php" file located in the "settings/siteaccess/example" directory where "example" is the name of your siteaccess (actual site but not the administration interface):

```
[multiprice]
Source=content/datatype/view/ezmultiprice.tpl
MatchFile=datatype/multiprice.tpl
Subdir=templates
```

It is also recommended to add a possibility for site visitors to change their preferred currency "on-the-fly" as described in the "Adding a toolbar for customers" section.

The following templates for viewing multi-price products are also available:

- design/base/override/templates/full/multiprice_product.tpl
- design/base/override/templates/line/multiprice_product.tpl
- design/base/override/templates/embed/multiprice_product.tpl
- design/base/override/templates/listitem/multiprice_product.tpl

To use these templates, add the following lines to the "override.ini.append.php" file:

```
[multiprice_product_full]
Source=node/view/full.tpl
MatchFile=full/multiprice_product.tpl
Subdir=templates
Match[class_identifier]=myproduct

[multiprice_product_line]
Source=node/view/line.tpl
MatchFile=line/multiprice_product.tpl
```

```

Subdir=templates
Match[class_identifier]=myproduct

[multiprice_product_embed]
Source=content/view/embed.tpl
MatchFile=embed/multiprice_product.tpl
Subdir=templates
Match[class_identifier]=myproduct

[multiprice_product_listitem]
Source=node/view/listitem.tpl
MatchFile=listitem/multiprice_product.tpl
Subdir=templates
Match[class_identifier]=myproduct

```

and replace "myproduct" with the actual class identifier of your multi-price products. (To check the class identifier, access the "Setup" tab in the administration interface, click "Classes" on the left, select the "Content" class group and find your multi-price product class.)

If you are going to use these templates, you will have to specify "user_preferred_currency" in the "CachedViewPreferences[full]" setting for all siteaccesses. To do this, open the "site.ini.append.php" configuration file located in the "settings/siteaccess/example" directory (replace "example" with the actual name of the siteaccess) and edit it. If the "[ContentSettings]" section of the configuration file already contains something like

```
CachedViewPreferences[full]=<list_of_user_preferences>
```

then you will have to append a semicolon and "user_preferred_currency" at the end of the line, for example:

```
CachedViewPreferences[full]=admin_navigation_content=0;
admin_navigation_details=0;<...>;admin_bookmarkmenu=1;
admin_left_menu_width=13;user_preferred_currency=''
```

Note that this configuration line tends to be very long. It is simplified in the example above (a lot of settings were replaced with <...> in order to keep things short).

If the "[ContentSettings]" section does not contain a line that starts from "CachedViewPreferences[full]", create it:

```
CachedViewPreferences[full]=user_preferred_currency=''
```

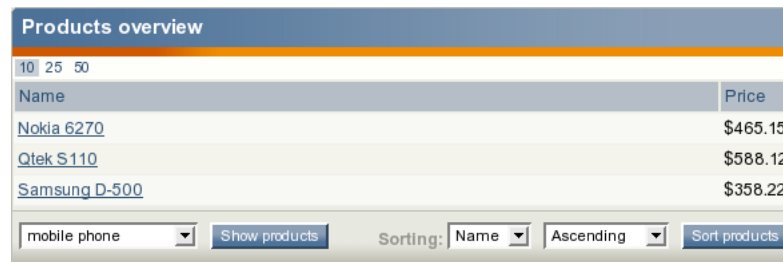
If this setting is not specified, your customers will have problems when changing the preferred currency setting (the interface will not be updated because of the cache problem).

4.13.9 Products overview

A user/administrator can view all products grouped by class and sorted either by price or name by requesting "shop/productsoverview" in the URL. Site administrators can also access this interface by clicking the "Webshop" tab and selecting the "Products overview" link on the left.

The following screenshot shows how this interface looks like when a site administrator is viewing products. Please note that only prices in the admin's preferred currency (page 481) are displayed for multi-price products.

(see figure 4.103)



Products overview	
10 25 50	
Name	Price
Nokia 6270	\$465.15
Qtek S110	\$588.12
Samsung D-500	\$358.22

mobile phone Sorting: Name Ascending

Figure 4.103: *The products overview interface.*

Filtering by class

The screenshot above shows the situation when the products from the "mobile phone" class are displayed. If you wish to view products from another product class, select the desired class name from the drop-down list and click the "Show products" button.

Choosing the sorting order

The screenshot above shows the situation when the products are sorted alphabetically by name. If you wish to sort products in some other order (for example by price), choose the desired sorting parameters and click the "Sort products" button.

4.13.10 Exchange rates update handlers

The exchange rates update handlers make it possible to retrieve the latest exchange rates from external sources and thus update the auto rates. You should specify the desired handler in the "ExchangeRatesUpdateHandler" setting described in the next subsection, otherwise the system will not be able to update the auto rates. You can either use the built-in "eZECB" handler for getting the exchange rates from the website of the European Central Bank or extend the system by creating your own update handler.

Settings

The "[ExchangeRatesSettings]" section of the "settings/shop.ini" configuration file defines the update handler that will be used for updating auto rates. Under this section, the following settings can be specified:

- The "RepositoryDirectories[]" array specifies the directories where eZ Publish will search for built-in update handlers. The exact location of the handler in the directory is specified using the "ExchangeRatesUpdateHandler" setting.
- The "ExtensionDirectories[]" array specifies the extension directories where eZ Publish will search for additional update handlers. By default eZ publish will search in the "exchangeratehandlers" subdirectory inside your extension. The exact location of the handler in the directory is specified using the "ExchangeRatesUpdateHandler" setting.
- The "ExchangeRatesUpdateHandler" setting specifies the update handler that will be used.
- The "BaseCurrency" setting specifies the base currency for auto rates. The default value of this setting is "EUR". It is recommended (but not required) that you specify one of the existing currencies in this setting.

The "[ECBExchangeRatesSettings]" section of the "settings/shop.ini" configuration file defines the specific settings for the "eZECB" update handler. The combination of the "ServerName", "ServerPort" and "RatesURI" settings allows to specify the exact address of the XML file containing the currency rates.

Example 1

The following lines can be specified in the "[ExchangeRatesSettings]" section of the "shop.ini" configuration file:

```
ExchangeRatesUpdateHandler=eZECB
RepositoryDirectories[]=kernel/shop/classes/exchangeratehandlers
ExtensionDirectories[]
BaseCurrency=EUR
```

These settings will instruct eZ Publish to use the built-in update handler located at "kernel/shop/classes/exchangeratehandlers/ezeceb/ezecebhandler.php" and use EUR as the base currency for auto rates.

Example 2

You can extend the system by creating custom update handlers for special needs. For example, if you have an extension "myshop" that includes an update handler "mybank", you can put the following lines into an override for the "shop.ini" configuration file:

```
[ExchangeRatesSettings]
ExchangeRatesUpdateHandler=mybank
ExtensionDirectories[]=myshop/classes
```

or

```
[ExchangeRatesSettings]
ExchangeRatesUpdateHandler=mybank
RepositoryDirectories[]=extension/myshop/classes/exchangeratehandlers/
```

These settings will instruct eZ Publish to use the update handler located at "extension/myshop/classes/exchangeratehandlers/mybank/mybankhandler.php"

Example 3

The following lines can be specified in the "[ECBExchangeRatesSettings]" section of the "shop.ini" configuration file:

```
ServerName=http://www.ecb.int
ServerPort=80
RatesURI=stats/eurofxref/eurofxref-daily.xml
```

These settings will instruct the eZECEB handler to import the currency exchange rates from http://www.ecb.int:80/stats/eurofxref/eurofxref-daily.xml.

Creating new handlers

This section reveals some helpful tips for those developers who want to create a new exchange rates update handler (only for people who are familiar with PHP). Please note that it is not recommended to modify the eZ publish kernel and thus you implement it as an extension. The following list reveals how you can implement your own handler for rates updating.

1. Create the following subdirectories in the "extension" directory of your eZ publish installation:

- myextension
- myextension/settings
- myextension/exchangeratehandlers
- myextension/exchangeratehandlers/mybank

2. Create a file called "mybankhandler.php" in the "myextension/exchangeratehandlers/mybank" directory. This file must contain a PHP class called "MyBankHandler". You should extend your class from the "eZExchangeRatesUpdateHandler" class (kernel/shop/classes/exchangeratehandlers/ezexchangeratesupdatehandler.php) and reimplement the "initialize" and "requestRates" functions. The "initialize" function is called while creating a handler-object. It allows to initialize your handler with some preset values (for example from INI file). The "requestRates" function performs the actual update of the rates. This function will assign an array of the retrieved rate values to the "\$RateList" member variable in the following format:

```
$RateList = array( 'currencyCode1' => 'rateValue1',  
.....  
'currencyCodeN' => 'rateValueN' );
```

3. Create a file called "shop.ini.append.php" in the "myextension/settings" directory and add the following lines into it:

```
[ExchangeRatesSettings]  
ExchangeRatesUpdateHandler=mybank  
ExtensionDirectories[]=myextension
```

This will instruct eZ Publish to use the update handler located at "extension/myextension/exchangeratehandlers/mybank/mybankhandler.php".

4. To activate your extension in eZ Publish, log in to your eZ Publish administration interface, click on the "Setup" tab, and then click "Extensions" on the left. You will see the list of available extensions. Select the "myextension" item and click the "Apply changes" button.

4.13.11 Upgrading your webshop

If you have just upgraded your eZ Publish installation in order to start using multiple currencies, you will need to create currencies (page 471) and convert all your products to multi-price format using the "convertprice2multiprice.php" script located in the "bin/php/" directory. Please note that you should launch this script from the root of your eZ Publish installation.

The script will go through all classes and objects containing an attribute of the price datatype and create an attribute of the multi-price datatype instead without changing the identifiers of the attributes, objects and classes. This means that your overrides will still be applied to converted objects (please refer to the "Templates for viewing multiple products" section if you wish to display only price in the preferred currency to a customer).

It is recommended to create all the currencies (including your locale currency) and check their exchange rates before launching the script.

Example

If your locale currency is USD, one of your simple price products costs for example \$100 and you wish to start using the multi-currency feature, you can create for example the following three currencies:

Currency code	Currency rate
EUR	1
NOK	7.9675
USD	1.2104

These rates can be for example auto rates retrieved from the website of the European central bank.

After successful execution of the "convertprice2multiprice.php" script the product with \$100 simple price will be automatically converted to a multi-price product with one custom price (base price) and two auto prices as shown in the following screenshot.

(see figure 4.104)

Price (required):

	Currency	Value
<input type="checkbox"/>	EUR	82.62(Auto)
<input type="checkbox"/>	NOK	658.25(Auto)
<input type="checkbox"/>	USD	<input type="text" value="100.00"/>

Figure 4.104: The resulting prices after product upgrading.

As you can see, the script converts the base price from USD to EUR and NOK using the existing currency exchange rates.

If you have forgotten to create your locale currency, the script will automatically create it with custom rate set to 1. The table of currencies will then look like this:

Currency code	Currency rate
EUR	1
NOK	7.9675
USD	1

These rates are incorrect (1 USD does not cost 1 EUR). The auto prices for all converted products will be also incorrect since generated according to incorrect currency rates.

4.14 View caching

Caching is a widely used technique supposing that frequently used information is retained in a temporary storage area for rapid access. It is extremely effective when the original data is expensive (usually in terms of access time) to fetch or compute/generate relative to reading the cache. Once the data is stored in the cache, future use can be made by accessing the cached copy rather than re-fetching or re-computing the original data, so that the average access time is lower.

eZ Publish includes a powerful caching mechanism that allows to improve system performance. This chapter describes a fundamental part of the cache systems in eZ Publish, called *content view caching* ("view caching" for short). This mechanism only works for "view" and "pdf" views of the "content" module.

Node view cache

This subsection describes how the view caches are generated when the nodes are accessed.

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 in turn returns it to the rest of the system. eZ Publish automatically generates an array called "module_result" containing information about which module that was run, which view that was called, the output that was produced and so on.

The actual output of the view (i.e. the XHTML code generated using one of the node templates (page 176)) is put to "\$module_result.content" and is included in the page layout (page 180) by accessing this template variable:

```
{$module_result.content}
```

When the page layout is rendered, the {`$module_result.content`} part will be replaced with the actual output. If view caching is enabled, the entire result of the module will be cached. This means that the contents of the "module_result" variable will be put into a cache file located in the "var/example/cache/content" directory (where "example" is usually the name of the siteaccess that is being used - it is set by the "VarDir" directive in "site.ini" or an override).

Please note that eZ Publish creates multiple view caches for the nodes based on roles and user preferences. This means that for example different users (who are logged in, with different permissions/preferences) will be served different caches while anonymous users and users with the same type of permissions/preferences will be served the same file. In other words, when view caching is on, 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 page layout. Please note that the page layout itself is not cached by default.

Another thing worth noticing is that the view caches are also depending on some other parameters. For example:

- View mode
- Language
 - View parameters in the URL
- Layout (for example print will be in different cache files)
- etc.

Example

Let's say that node 46 is the company about page and the custom template "aboutpage.tpl" overrides the default "node/view/full.tpl" template for this particular node.

Both the virtual URL "http://www.mysite.com/company/about" and the system URL "http://www.mysite.com/content/view/full/46" point to this page. When one of these URLs is requested, the system will execute the "view" view of the "content" module using "46" as the ID number of the node and "full" as the view mode (page 176). The resulting XHTML code containing the company about information will be generated using the "aboutpage.tpl" template. The output will be put to "\$module_result.content" and then will be included in the page layout.

If view caching is enabled for the "full" view mode, the entire result of the module will be cached in a file. The cache files get long names like for example "46-122bc591bf62e87a4e9ddcb5ba352bc4.cache". Next time the company about page is being accessed, the system will not go through the burden of executing the "view" view of the "content" module (generating the result, etc.) but load the cache file instead.

The \$node variable

In eZ Publish versions prior to 3.9, the "\$node" variable might be present in the page layout when view caching is disabled. From 3.9, this variable is not available in the page layout, regardless of whether view caching is enabled or not. It is recommended to use \$module_result for fetching the necessary information (for example, "\$module_result.node_id" outputs the ID number of the node that is being viewed).

PDF cache

The following text explains how the view cache is generated when a PDF version of a site page (a content node) is being accessed. Note that the PDF export mechanism has been deprecated in eZ Publish 4.0 and will be removed in future releases.

Whenever eZ Publish is requested to generate a PDF version of a node, it executes the program code that is associated with the "pdf" view of the "content" module. Instead of inserting the output into the page layout via \$module_result.content, the system will fetch the actual page content (i.e. the attributes of the object that is encapsulated by the specified node) using the

"pdf.tpl" template and generate a PDF file using "execute_pdf.tpl". These default templates are located in the "templates/node/view" directory of the standard design.

If view caching is enabled for the "pdf" view, the resulting PDF file will be cached. This means that the system will save a copy of the actual PDF document into a cache file located in the "var/example/cache/content" directory (where "example" is usually the name of the siteaccess that is being accessed - it is controlled by the "VarDir" directive in "site.ini" or an override).

Example

If node 46 is the company about page, then accessing the URL "http://www.mysite.com/content/pdf/46" will lead to executing the "pdf" view of the "content" module. The system will generate a PDF version of the company about page and display it to the user.

If view caching is enabled for the "pdf" view, the resulting PDF document will be cached in a file, for example called "46-3579d18de31e99fc84d2d9a5f113c3be.cache". Please note that this file can be opened using a PDF reader (in some cases it would have to be renamed to .pdf).

4.14.1 Configuring the view cache

The view caching mechanism is enabled by default. However, you probably want to turn it off during site development (otherwise any changes being made in your custom node templates (page 176) will not be visible on the site until you clear the caches). This can be done by adding the following line under the "[ContentSettings]" section in the "site.ini.append.php" file of your siteaccess:

```
ViewCaching=disabled
```

Note that it is strongly recommended to re-enable the view caching when development has finished. This can be done by changing it from "disabled" to "enabled":

```
ViewCaching=enabled
```

The CachedViewModes setting located in the "[ContentSettings]" section of the "site.ini" configuration file (or an override) controls which view modes the caching will be enabled for. The default value of this setting defines that view cache should be stored for "full", "sitemap" view modes and for the "pdf" view:

```
CachedViewModes=full;sitemap;pdf
```

However, note that the "pdf" view of the content module is deprecated.

If you need to disable view caching for a specific page, add the following line in the beginning of the template that is used:

```
{set-block scope=global variable=cache_ttl}0{/set-block}
```

This will set the global variable "cache_ttl" to zero for the current template. The "cache_ttl" variable contains the TTL (Time To Live) value as seconds. A value of 0 means that the result should not be changed. A value of -1 means that the view cache should never expire, see the example below.

```
{set-block scope=global variable=cache_ttl}-1{/set-block}
```

Roles

The cache files are different for dissimilar role combinations. This means that the templates can have conditions based on roles (page 161) even when view caching is on.

User preferences

The following text describes handling of user preferences and the way the preferences of the current user are taken into account when generating content view cache.

For example, whenever the user performs the following actions using the administration interface:

- Enabling or disabling the bookmark menu (+/-)
- Adjusting the horizontal size of the content structure menu (small / medium / large)
- Choosing the view mode (list / thumbnail / detailed) and "items per page" limitation (10 / 25 / 50) for the sub items window
- Changing the visibility of different windows (Preview / Details / Translations / Locations / Relations)

...or sets any other user preference, the system executes the "preferences" view of the "user" module in order to save the selected value. The information about which preference has been changed is passed using view parameters:

```
.../user/preferences/set/<name_of_preference>/<value>
```

for example

```
http://my.com/myadmin/user/preferences/set/admin_left_menu_width/medium
```

After saving the selected value, the "preference" view of the "user" module will redirect the user back to the last accessed page.

If your templates of cached view modes have conditions based on user preferences, you should specify which preferences that are used together with the different view modes - this can be done using the `CachedViewPreferences` setting located under the "[ContentSettings]" section of the "site.ini.append.php" configuration file.

Example

Let's say that you are using prices in several currencies for your products and the "node/view/full.tpl" template is overridden for products in order to display prices in the user's preferred currency (page 481). If view caching is enabled for the "full" view mode, the system will store view cache files for the product view pages. If the cache is generated regardless of the preferred currency, when another preferred currency is chosen, the same product page that is being viewed will be returned (in other words, it will not change).

In order to avoid this, you need to specify the "user_preferred_currency" preference in the "CachedViewPreferences[]" array using "full" as an array key so that the cache will be stored for each possible preferred currency on your site. To do this, open the "site.ini.append.php" configuration file located in the "settings/siteaccess/example" directory (replace "example" with the actual name of the siteaccess) and edit it. If the "[ContentSettings]" section of the configuration file already contains something like

```
CachedViewPreferences[full]=<list_of_user_preferences>
```

then you will have to append a semicolon and "user_preferred_currency" at the end of the line, for example:

```
CachedViewPreferences[full]=admin_navigation_content=0;  
admin_navigation_details=0;<...>;admin_bookmarkmenu=1;  
admin_left_menu_width=13;user_preferred_currency=''
```

Note that this configuration line tends to be very long. It is simplified in the example above (a lot of settings were replaced with <...> in order to keep things short).

If the "[ContentSettings]" section does not contain a line that starts from "CachedViewPreferences[full]", create it:

```
CachedViewPreferences[full]=user_preferred_currency=''
```

Related siteaccesses

The RelatedSiteAccessList setting located under the "[SiteAccessSettings]" section of the "site.ini.append.php" configuration file controls which other siteaccesses the view cache should be cleared for when it is cleared for the current siteaccess. (This requires the VarDir configuration setting to be the same for all related siteaccesses.) If the RelatedSiteAccessList setting is not specified, the cache subsystem will use the AvailableSiteAccessList setting instead.

Note that the AvailableSiteDesignList setting located under the "[VersionView]" section of the "content.ini" configuration file is no longer (from eZ Publish 3.8) used by the cache system. In 3.7 and earlier versions, it could contain an array of designs that would be touched when the caches were cleared.

4.14.2 Clearing the view cache

When a new version (also the first version) of an object is published, the system will automatically clear the view cache for the following items:

- All published nodes of the object
- The parent nodes
- Nodes of other objects that have the same keyword as the object (if the "Keywords" datatype is used by at least one of the attributes).

In addition, the view caches of the following nodes will also be cleared:

- Nodes of related and reverse related objects that have relations of the "common" type
- Nodes of reverse related objects that have relations of the "XML embedded" type

This is controlled by the "ClearRelationTypes" configuration setting.

This default behavior can be extended by configuring the smart viewcache cleaning system (page 502).

Note that clearing the view caches for a set of nodes within a certain siteaccess means that caches for the same nodes will also be cleared on all the related siteaccesses.

Using the administration interface

The administration interface allows you to clear the view cache for a node that is being viewed:

1. Navigate to the node that you wish to clear the view cache for. In other words, make sure that the target node is being displayed.
2. In the title bar of the preview window, click on an icon that indicates the node type and select the "Delete the view cache" item from the pop-up menu (shown in the following screenshot):

(see figure 4.105)

You can also clear the view cache for the entire subtree (together with the node itself) by selecting the "Delete the view cache from here" item from the pop-up menu.

Using the script

It is possible to clear the view cache for a specific node or subtree using the "ezcontentcache.php" script located in the "bin/php" directory of the eZ Publish installation. The following examples demonstrate how this can be done.

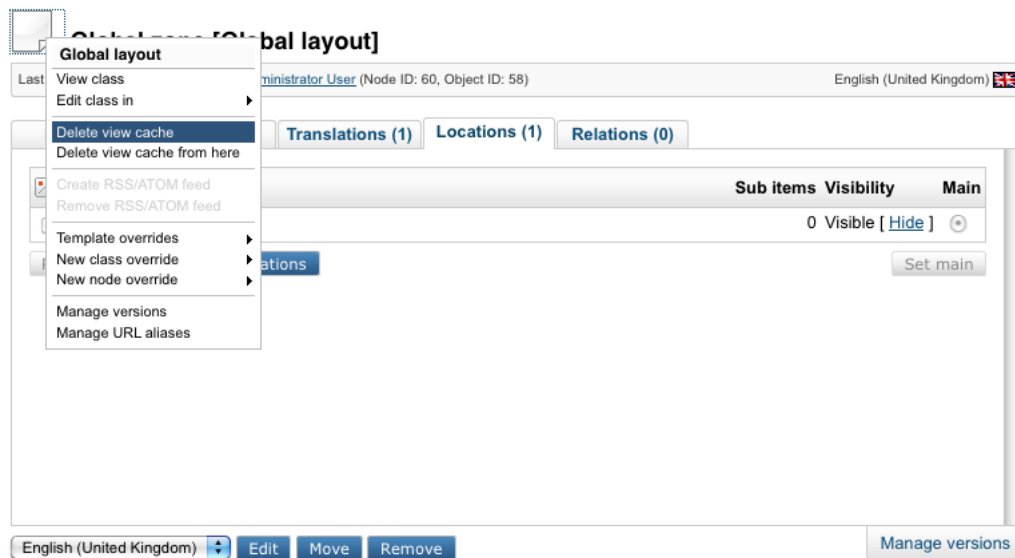


Figure 4.105:

Example 1

Let's say that node 46 is the company about page (<http://www.mysite.com/company/about>) and you have made some changes to a custom template that is used for this node. If view caching is enabled, your changes will not be seen until you clear the view cache for node 46:

1. Navigate into the eZ Publish directory.
2. Run the script using the following command:

```
./bin/php/ezcontentcache.php --clear-node=46
```

or

```
./bin/php/ezcontentcache.php --clear-node=/company/about
```

This will clear the view cache for node 46 (and for other locations of the same object, if any), their parent nodes, nodes of objects containing the same keywords, nodes of related and reverse related objects that have relations of the "common" type and nodes of reverse related objects that have relations of the "XML embedded" type. If you need to clear view caches for several nodes, specify their ID numbers (or nice URLs) separated by commas:

```
./bin/php/ezcontentcache.php --clear-node=46,59,63
```

The script will clear all content caches related to each of the given nodes.

Example 2

Let's say that node 72 is the company news folder (<http://www.mysite.com/company/news>) containing a lot of news articles. To clear the view caches for this folder and the nodes below it, do the following:

1. Navigate into the eZ Publish directory.
2. Run the script using the following command:

```
./bin/php/ezcontentcache.php --clear-subtree=72
```

or

```
./bin/php/ezcontentcache.php --clear-subtree=/company/news
```

If you need to clear view caches for several subtrees, specify the nice URLs (or ID numbers) of their root nodes separated by commas:

```
./bin/php/ezcontentcache.php --clear-subtree=/company/news,/partners
```

The script will then clear view caches for the specified subtrees.

4.14.3 Smart view cache cleaning

The *smart viewcache cleaning system* (referred to as "svcs" on this page) makes it possible to set up custom rules that control which nodes the view cache should be cleared for when a published object is changed. This feature is turned off by default and thus the system will only clear the view cache for the following nodes when a new version of an object (also the first version) is published:

- All published nodes of the object
- The parent nodes
- Nodes of other objects that have the same keyword as the object (if the "Keywords" datatype is used by at least one of the attributes).

In addition, the view caches of the following nodes will also be cleared:

- Nodes of related and reverse related objects that have relations of the "common" type
- Nodes of reverse related objects that have relations of the "XML embedded" type

This is controlled by the "ClearRelationTypes" configuration setting.

If you want to use the smart viewcache cleaning feature, make sure the "viewcache.ini.append.php" file located in the "settings/siteaccess/example_admin" directory (replace "example_admin" by the name of the siteaccess that is used for adding and editing content) contains the following lines:

```
[ViewCacheSettings]
SmartCacheClear=enabled
```

These lines will instruct the system to follow the rules specified in this configuration file in addition to the default behavior. The configuration file usually includes a single, common settings section called "[ViewCacheSettings]" and multiple specific sections that describe the rules determining which *additional nodes* the view cache should be cleared for. Note that the view cache of those additional nodes will also be cleared in accordance with the svcs rules (see example 5). These sections are named after the class identifiers.

When a published object is changed, svcs gets its identifier as an input parameter. It checks which class this object belongs to and looks for a section named after that class identifier in the "viewcache.ini" configuration file (and its overrides). The rules specified in this section will be applied to the parent nodes that are listed in the path_string attribute of the initially changed node. If the published object has several nodes/locations, svcs will sequentially handle their path strings. The following list reveals how svcs will handle each node of the published object:

1. Scan the parent nodes listed in the node's "path_string" attribute (the maximal quantity of nodes that will be scanned is controlled by the "MaxParents" setting).

2. Perform the following actions for each of the parent nodes:
 - Check which class the object encapsulated by that node belongs to.
 - If the identifier of that class is listed in the "DependentClassIdentifier[]" array, add the matching parent node to the list of additional nodes.
3. If the "ObjectFilter[]" setting is empty, clear the view caches for additional nodes. Otherwise, check the identifiers of the objects encapsulated by additional nodes and only clear caches for those that have their object identifiers listed in the "ObjectFilter[]" array. In both cases, the caches are cleared using the method(s) specified in the "ClearCacheMethod[]" setting.

From 3.9, it is also possible to clear the caches for a set of arbitrary objects when objects of specific types are changed. If the "AdditionalObjectIDs[]" setting contains a list of object identifiers, the system will clear the view cache for all the locations (nodes) of these objects, regardless of whether they are listed in "path_string" or not.

The following table gives detailed description for the configuration settings mentioned above.

Name	Type	Description
DependentClassIdentifier	An array of class identifiers (not ID numbers)	Specifies which content classes that will be considered as "dependent classes". If a node encapsulating an object of such a class is listed in "path_string", svcs will add it to the list of additional nodes. The view cache for additional nodes will be cleared using the method(s) specified in the next setting.
ClearCacheMethod	An array of strings	Sets which method(s) to use when clearing the view caches for additional nodes. This setting is an array of strings where only six pre-defined values can be used (see the next table). Name: object Description: Clear the view cache for all the locations (nodes) of the object. Name: parent Description: Clear the view cache for the parent node(s)

		<p>of the object.</p> <p>Name: relating Description: Clear the view cache for related and reverse related objects that have relations of the "common" type and reverse related objects that have relations of the "XML embedded" type (according to the "ClearRelationTypes" configuration setting).</p> <p>Name: keyword Description: Clear the view cache for the objects that have the same keyword as this object.</p> <p>Name: siblings Description: Clear the view cache for all the siblings of this node/object.</p> <p>Name: all Description: Clear the view cache for all the listed above.</p> <p>Name: children Description: Clear view cache on direct children of this node.</p>
ObjectFilter	An array of object ID numbers	If specified, the view caches will only be cleared for those additional nodes that encapsulate the objects with these identifiers. If not specified, all additional nodes will have their view cache cleared.
MaxParents	Integer	Sets how many parents in "path_string" will be checked. If not specified, svcs will scan all the parents listed in

		"path_string".
AdditionalObjectIDs	An array of object ID numbers	Makes it possible to clear the caches for a set of arbitrary objects regardless of whether their locations are listed in the node's "path_string" attribute or not.

Example 1

Let's say that both view caching and svcs are enabled with the following part of a content structure:

(see figure 4.106)

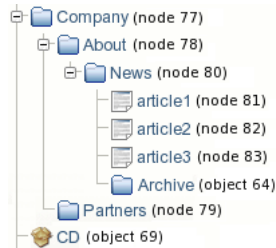


Figure 4.106: A part of the site content structure.

 If you do not specify any rules for svcs, changing an article will lead to clearing the view caches for all its published nodes, their parent nodes, nodes of objects containing the same keywords, nodes of related and reverse related objects that have relations of the "common" type and nodes of reverse related objects that have relations of the "XML embedded" type (this is the default behavior of svcs).

If the "article2" object has only one location, does not contain any keywords and is not related to any other objects, changing it will lead to clearing the view cache of the article itself and the "News" folder. The view cache of the "About" and "Company" nodes will not be cleared.

However, you can extend this default behavior by adding the following configuration group to the "viewcache.ini.append.php" configuration file of your (admin) siteaccess:

```

[article]
DependentClassIdentifier []
DependentClassIdentifier []=folder
ClearCacheMethod []
ClearCacheMethod []=object
  
```

Now, if an article is changed, the system will fetch all the parent nodes of this article sequentially according to its "path_string" attribute (the path string for "article2" ends with "/77/78/80/82/"),

check which of them are folder nodes and clear the view cache for those folders. This means that changing "article2" will lead to clearing the view cache of "article2", "News", "About", "Company" and all the parent folder nodes that are located above the "Company" node.

Example 2

It is possible to limit the depth of fetching node IDs from the "path_string" attribute like this:

```
[article]
DependentClassIdentifier []
DependentClassIdentifier []=folder
ClearCacheMethod []
ClearCacheMethod []=object
MaxParents=2
```

This will tell the system to take into account only two penultimate items from the node's path string (i.e. parent and grandparent of the node). This means that changing "article2" will only lead to clearing the view caches of "article2", "News" and "About". The view cache of the "Company" node and its parent folders will not be cleared.

Example 3

You can use the "ObjectFilter[]" configuration array so that a folder node listed in "path_string" will not be included in the list of additional nodes unless the object encapsulated by this node is explicitly specified in the following way:

```
ObjectFilter []
ObjectFilter []=<object_id1>
ObjectFilter []=<object_id2>
...
```

Assuming that the "Company" folder object in example 1 has ID number 74 (while its node ID is 77), you can specify the following settings in the "viewcache.ini.append.php" of your (admin) siteaccess:

```
[article]
DependentClassIdentifier []
DependentClassIdentifier []=folder
ClearCacheMethod []
ClearCacheMethod []=object
ObjectFilter []
ObjectFilter []=74
```

If "article2" is changed, svcs will check which of the folder nodes listed in the given path string (nodes 80, 78, 77, ...) have their object ID numbers matching one of the values in the "ObjectFilter[]" array and thus only the "Company" page will be included in the list of additional nodes.

The system will only clear view caches of "article2", "News" and "Company" (according to the default behavior and the given svcs rules). The view cache of the "About" page will not be cleared because the ID number of that object is not 74.

Example 4

If you wish to clear the caches for a set of arbitrary objects when objects of specific types are changed, you will have to specify a list of object ID numbers using the "AdditionalObjectIDs[]" configuration array. Assuming that the object ID numbers of the "Archive" folder and the "CD" product are 64 and 69 accordingly, you can specify the following settings in the "view-cache.ini.append.php" of your (admin) siteaccess:

```
[article]
AdditionalObjectIDs []
AdditionalObjectIDs []=64
AdditionalObjectIDs []=69
```

This will tell the system to always clear the view cache of the "Archive" folder and the "CD" product when an article (any article) is changed. This means that changing "article2" will lead to clearing the view caches of "article2", "News", "Archive" and "CD".

Example 5

Let's say that you have specified the following settings in the "viewcache.ini.append.php" of your (admin) siteaccess:

```
[article]
AdditionalObjectIDs []
AdditionalObjectIDs []=69

[product]
AdditionalObjectIDs []
AdditionalObjectIDs []=64
```

The settings located in the "[article]" configuration block will tell the system to always clear the view cache of the "CD" product when an article is changed. This means that changing "article2" will lead to clearing the view caches of "article2", "News" and "CD". When clearing the cache of the "CD" product, svcs will apply the rules specified in the "[product]" section, and thus the view cache of the "Archive" folder will also be cleared.

4.14.4 Pre-generation of view cache

The "cache on request" approach described in the previous sections supposes that the view cache for a page is created when this page is being accessed for the first time. The additional "cache on publishing" functionality makes it possible to generate view cache files when a node is being created or modified. This makes the publishing process a bit slower (not recommended for sites with lots of content editors), but reduces the access time when the pages are being requested.

The "cache on publishing" feature is disabled by default. This behavior is controlled by the `PreViewCache` setting located under the "[ContentSettings]" section of the "site.ini" configuration file. Please note that enabling this feature will only affect the view caches generated for the "full" view mode. Whenever an object is published, the system will generate the view cache for all the nodes/locations of this object and their parent nodes. The `PreCacheSiteaccessArray` setting located at the same place controls which siteaccess(es) the view cache should be generated for (usually public siteaccesses that are used for viewing content).

If you wish to create the view cache files on publishing, add the following lines to the "[ContentSettings]" section of "settings/siteaccess/example_admin/site.ini.append.php" configuration file (replace "example_admin" by the name of the siteaccess that is used for creating and editing content):

```
PreViewCache=enabled
PreCacheSiteaccessArray []
PreCacheSiteaccessArray []=example
```

This will enable the "cache on publishing" feature and tell the system that the view cache should be generated for the "example" siteaccess. If you have a news folder containing a lot of articles, editing one of them will lead to re-generating the view cache files of the article itself and its parent (the news folder). When a new article is published in the folder, the system will generate a view cache for the newly added article and regenerate the view cache of the news folder.

Please note that the view cache will only be generated for the Anonymous user by default. This behavior is controlled by the `PreviewCacheUsers` setting located in the "[ContentSettings]" section of the "site.ini" configuration file.

4.15 Notifications

eZ Publish has a built-in notification system that allows users to be informed about miscellaneous events that occur. It is possible to be notified when objects are updated or published, when workflows are executed and so on.

There are two built-in types of notifications:

- Subtree notifications
- Collaboration notifications

Subtree notifications

It is possible to subscribe for notifications about a subtree. For example, if you have a set of articles located under a folder called "Business", a user can subscribe for *subtree notifications* for this folder. The system will then send an E-mail to the user every time changes are made under the "Business" folder. The following changes will trigger a notification:

- When a new node is published within the subtree.
- When the contents of an existing node is changed.

A user can choose to receive notifications in the form of a single E-mail or as a digest of messages.

Collaboration notifications

The eZ Publish collaboration system allows you to work together with other people so that you can approve/reject any changes they made when it comes to content. For example, you can specify that all the changes made in the "Standard" section (page 136) can not be published without your approval. (This can be done by creating a new "Approve" event within a new workflow (page 165) initiated by the "content-publish-before" trigger function.) If somebody (except you, the administrator) edits content located under the "Standard" section, the system will generate new collaboration messages. For example, if somebody changes article "A", the system will generate a new collaboration message "article A awaits your approval" for you and another collaboration message "article A awaits approval by editor" for the user who changed it.

To view your collaboration messages, click the "My Account" tab in the administration interface and then access the "Collaboration" link on the left. You will be able to review/approve/reject the changes.

You can use collaboration notifications to be notified by E-mail about new collaboration messages. The system will send you an E-mail every time a new collaboration message is generated for you.

Processing notifications

In the root of the eZ Publish directory there is a file called "runcronjobs.php". It takes care of processing the workflows, notifications and other tasks that should be processed in the background. If you are going to use the notification system, "runcronjobs.php" must be executed periodically. The most common way to do this is to set up a scheduled job that runs every 30-60 minutes or so. Please refer to the "Configuring cronjobs (page 411)" and "Running cronjobs (page 414)" sections for more information.

In accordance with the instructions specified in the "cronjobs/notification.php" file, "runcronjobs.php" launches the main notification processing script "kernel/classes/notification/eznotificationeventfilter.php".

If you need to launch this script manually, add the "notification/runfilter" notation to the administration interface URL and then click the "Run notification filter" button there (see the next screenshot).

(see figure 4.107)

Notification

Run notification filter Spawn time event

Figure 4.107:

 Please note that processing notifications may cause a timeout error if there is a huge amount of notification events in the database. Because of this, the "runfilter" view of the "notification" module should only be used for testing and debugging.

4.15.1 Using the admin interface

Subtree notifications

Subscribing

You can easily subscribe for subtree notifications about an object using either the context menu or the notification settings interface.

Using the context menu

To subscribe for subtree notification for an object, you should do the following:

1. Log in to the administration interface. You should see the "Content structure" tree on the left where the top node is selected. The following screenshot shows how the system will display the contents of the selected node and the list of its sub-items.

(see figure 4.108)

2. Locate the desired node in the "Content structure" tree or the "Sub items" window, click on its icon and select "Add to my notifications" from the context menu - this is shown in the screenshot below.

(see figure 4.109)

3. The system will add a new subtree notification and show you a confirmation:
(see figure 4.110)

Using the notification settings interface

It is possible to subscribe for subtree notifications for an object by adding this object to the "My item notifications" list located towards the bottom of the notification settings interface. The following text reveals how this can be done:

1. Click the "Dashboard" tab in the administration interface and select the "My notification settings" link on the left. You will be taken to the notification settings interface as shown in the screenshot below.

(see figure 4.111)

At the bottom of this interface you will get a list of your notifications as shown in the screenshot below.

(see figure 4.112)

The "My notifications settings" interface can be also accessed by adding the "/notification/settings" notation to the site URL.

2. Look at the "My item notifications" list located towards the bottom of the notification settings interface. All the items that you have already subscribed for are listed here. Click the "Add items" button to add a new notification.

The screenshot displays the eZ CMS admin interface. At the top, there is a navigation bar with tabs for 'Dashboard', 'Content structure', 'Media library', 'User accounts', and 'Setup'. The 'Content structure' tab is active. Below the navigation bar, the breadcrumb path shows 'You are here: Home'. On the left, a 'Content structure' tree shows 'Home' as the root, with sub-nodes for 'Conference', 'Movies', 'News', and 'Live Video'. The main area shows the 'Home [Frontpage]' node, last modified on 28/11/2007 at 5:51 pm by Administrator User. The layout includes three blocks: 'Block: Main story (Manual)' with one item, 'Block: 2 items (Manual)' with two items, and 'Block: Multimedia carousel'. Below the blocks is a table of sub-items with columns for Name, Type, Modified, Published, and Priority.

Name	Type	Modified	Published	Priority
<input type="checkbox"/> Global zone	Global layout	23/11/2007 10:01 am	23/11/2007 9:51 am	0
<input type="checkbox"/> Conference	Frontpage	28/11/2007 11:09 am	22/11/2007 1:57 pm	10
<input type="checkbox"/> Conference Blog	Blog	22/11/2007 2:45 pm	22/11/2007 2:45 pm	20
<input type="checkbox"/> Live Video	Folder	23/11/2007 11:59 am	23/11/2007 11:59 am	30
<input type="checkbox"/> Discussion Forum	Forum	23/11/2007 11:17 am	23/11/2007 11:17 am	40

Figure 4.108:

3. The system will bring up the browse interface which will allow you to select the desired nodes:

(see figure 4.113)

Use the list to select the node which encapsulates the object that you wish to be notified about. Please note that it is possible to select multiple nodes/objects at the same time. You can navigate the list by clicking on the names of the nodes. If the desired node is located outside the "Content structure" tree then simply click the up arrow icon/button until it brings you to the root of the tree. This operation will allow you to for example switch to the "User accounts" tree and select user groups that are located there. The following illustration shows the up-arrow. (see figure 4.114)

It is possible to reconfigure how the list is displayed. For example, you can set the quantity of objects per page by clicking the "10" / "25" and "50" links. If you wish to browse image objects as thumbnails, simply click the "Thumbnail" button.

4. When you're finished selecting the desired object(s) (simply use the check-boxes to do this) click the "OK" button. The system will subscribe you for subtree notifications about

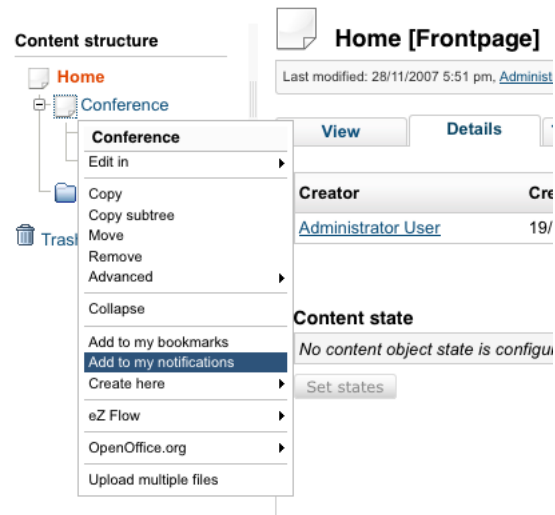


Figure 4.109:

Add to my notifications

Notification for node <Using an actual site> was added successfully.

OK

Figure 4.110:

My notification settings

Receive all messages combined in one digest

Receive digests

- Daily, at

 Once per week, on

 Once per month, on day number

If day number is larger than the number of days within the current month, the last day of the current month will be used.

Collaboration notification

Choose which collaboration items you want to get notifications for.

Approval

Apply changes

Figure 4.111:

these objects and add them to the "My item notifications" list.

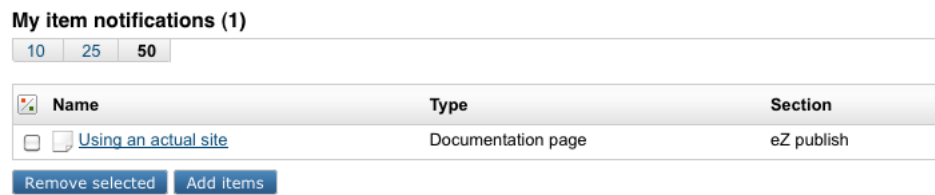


Figure 4.112:

Browse

To select objects, choose the appropriate radio button or checkbox(es), then click the "Select" button.

To select an object that is a child of one of the displayed objects, click the object name for a list of the children of the object.



Figure 4.113:



Figure 4.114: The "Up" button

Setting up the digest mode

If you wish to receive the subtree notifications as a daily/weekly/monthly digest, enable the digest mode as described below.

1. Access the notification settings interface either by adding the "/notification/settings" notation to the URL or selecting "Dashboard - My notification settings" in the administration interface.
2. The digest settings are located at the top of the notification settings interface. By default, the digest mode is disabled (as shown in the screenshot below).

(see figure 4.115)

To enable the digest mode, select the "Receive all messages combined in one digest" check-box and choose how often the digest should be sent to you.

- Once a day, at some fixed time (from 0:00 to 23:00).

My notification settings

Receive all messages combined in one digest

Receive digests

Daily, at

Once per week, on

Once per month, on day number

If day number is larger than the number of days within the current month, the last day of the current month will be used.

Collaboration notification

Choose which collaboration items you want to get notifications for.

Approval

[Apply changes](#)

Figure 4.115:

- Once a week, on some fixed day (from Sunday to Saturday).
 - Once a month, on some fixed day (from 1 to 31).
3. Click the "Apply changes" button to save your settings.

Unsubscribing

If you no longer wish to receive notifications about an object, use the following instructions to unsubscribe.

1. Access the notification settings interface either by adding the "/notification/settings" notation to the URL or selecting "Dashboard" and then "My notification settings" in the administration interface.
2. The "My item notifications" list located towards the bottom of the notification settings interface contains all the items that you have already subscribed for. Use check-boxes to select the item(s) that you no longer wish to be notified about (see the screenshot below).

(see figure 4.116)

3. Click the "Remove selected" button. The system will remove the selected item(s) from the list of notifications and thus you will no longer receive any messages about that/those object(s).

Name	Type	Section
<input type="checkbox"/> Using an actual site	Documentation page	eZ publish
<input checked="" type="checkbox"/> Customizing the E-mails	Documentation page	eZ publish

Figure 4.116:

Collaboration notifications

If you're using the collaboration system to work together with other people, you may wish to be notified by E-mail every time a new collaboration message is created for you. In this case, you should enable the collaboration notifications feature. The following text describes how to do this.

1. Access the notification settings interface either by adding the "/notification/settings" notation to the URL or selecting "dashboard" and then "My notification settings" in the administration interface.
2. Look at the "Collaboration notification" section located under the digest settings. By default, the collaboration notifications are disabled. If you wish to receive collaboration notifications, select the "Approval" check-box as shown in the following screenshot.

(see figure 4.117)

Figure 4.117:

3. Click the "Apply changes" button to save your settings. The system will then send you an E-mail every time a new collaboration message is generated for you.

Please note that collaboration notifications do not support digest mode.

4.15.2 Using an actual site

Subtree notifications are available for those users who are allowed to use the "notification" module by the role/policy settings. Please refer to "Granting access to notifications (page 522)" for more information about access rights. Collaboration notifications are only available when you're using the administration interface.

Subscribing for subtree notifications

A user can subscribe for subtree notifications about the item that is being viewed by clicking the "Keep me updated" button. The following screenshot shows a forum from one of the standard sites.

(see figure 4.118)

Small talk

This is a demo forum where you can discuss small talk or other programming languages ;)

New topic

Keep me updated

Topic	Replies	Author	Last reply
 How big can a colour be?	1	21/04/2004 11:36 am Administrator User	21/04/2004 11:37 am Administrator User Not TRUE!

Figure 4.118: The "keep me updated" button.

 After clicking this button, the system will add a new subtree notification and show a confirmation:

(see figure 4.119)

Notification was added successfully!

Add to my notifications

Notification for node <Small talk> was added successfully.

OK

Figure 4.119: The "notification added" confirmation for users.

 In the standard sites, the "Keep me updated" button is always displayed on the forum pages while other pages do not contain this button. The forum is controlled by the following templates:

- design/your_siteaccess/override/templates/full/forum.tpl

- ` design/your_siteaccess/override/templates/full/forum_topic.tpl`

Please refer to "Adding the "Keep me updated" button (page 520)" for more information about adding the update button to other templates/pages.

Setting the digest mode

The notification settings can be accessed regardless of the siteaccess/design that is used (as long as the permissions are ok). You can do the following to access the interface:

1. ` `After logging in to the system, add the `"/notification/settings"` notation to the site URL (`http://www.example.com/notification/settings`) in order to access the notification settings. You should see the digest settings and the "Node notification" list (as shown in the following screenshot).
` `(see figure 4.120)

Notification settings

Receive all messages combined in one digest

Time of day:
0:00

Daily

Weekly, day of week:
Sunday

Monthly, day of month:
1

If the day of month number you have chosen is larger than the number of days in the current month, then the last day of the current month will be used instead.

Node notification

Name	Class	Section	Select
News	Folder	1	<input type="checkbox"/>
Links	Folder	1	<input type="checkbox"/>

Remove

Store Cancel

Figure 4.120: Notification settings for users.

2. ` `By default, the digest mode is disabled. To enable the digest mode, select the "Receive all messages combined in one digest" check-box, and choose how often the digest should be sent to you.
` `

- ` `Once a day, at some fixed time (from 0:00 to 23:00).

- Once a week, on some fixed day (from Sunday to Saturday).
 - Once a month, on some fixed day (from 1 to 31).
3. Click the "Store" button to save your settings. (If you wish to discard changes, simply click the "Cancel" button.)

Unsubscribing

If you no longer wish to receive subtree notifications about an object, follow these instructions to unsubscribe:

1. Access your notification settings by adding the "/notification/settings" notation to the site URL (<http://www.example.com/notification/settings>).
2. The "Node notification" list located under the digest settings contains all the items that you have already subscribed for (look at the previous screenshot). Use check-boxes to select the item(s) that you no longer wish to be notified about.
3. Click the "Remove" button. The system will remove the selected item(s) from the list of notifications.

Please note that you can customize the notification settings template(s) by copying the default templates from either the standard or the admin design and changing them to suit your site.

4.15.3 Adding a "Keep me updated" button

A user can subscribe for subtree notifications for the page that is being viewed by making use of a "Keep me updated" button. Many of the default templates do not contain this button. The only exception is made for the forum pages where the button code is included into the following templates:

- design/your_siteaccess/override/templates/full/forum.tpl
- design/your_siteaccess/override/templates/full/forum_topic.tpl

Returning to the previous example, if you have a set of articles located under a folder called "Business", your users will not be able to subscribe for subtree notifications for this folder as long as there is no "Keep me updated" button there. Please note that the user must be logged in to make use of this feature.

You can easily add the "Keep me updated" button by inserting the following code into the override templates. For example, you can add this code to the "design/your_siteaccess/override/templates/full/folder.tpl" template:

```
<form method="post" action={'/content/action'|ezurl}>
<input type="hidden" name="ContentNodeID" value="{ $node.node_id}" />
<input type="submit" name="ActionAddToNotification" value="Keep me updated" />
</form>
```

After clearing the caches, the "Keep me updated" button will appear every time a user is viewing a folder. The same changes can be easily done for your articles and other content objects.

Please note that some of the default templates may already contain a "/content/action" form. In this case, make sure that all the variables listed in the above code fragment are present inside this form in the template. You can also have several forms posting data to "/content/action".

If you wish to have the button present in the page layout then you'll have to do it a bit differently. The reason for this is that the \$node variable is not present in the page layout.

```
{* Check if we have a node... *}
{if $module_result.node_id}

<form method="post" action={'/content/action'|ezurl}>

<input type="hidden" name="ContentNodeID" value="{ $module_result.node_id}" />
<input type="submit" name="ActionAddToNotification" value="Keep me updated" />

</form>

{/if}
```

4.15.4 Customizing the E-mails

It is possible to customize the notification e-mails by modifying templates. For example, the "plain.tpl" template located in the "templates/notification/handler/ezgeneraldigest/view/" directory of the standard design is the main notification template. It controls how the e-mails will be generated.

If you need to make changes to this template, you should copy it to your custom design and change it there. For example, you could copy it and add some additional/static text that will appear in all e-mails that are sent out. Remember to clear the caches before testing the changes. Please note that you should not change the default template but instead copy them to your own design.

4.15.5 Granting access to notifications

The built-in permission system controls whether users are allowed to use notifications or not. The following text explains how you can check and assign the necessary permissions.

Checking the access rights

The following text explains how you can view a user or a user group and check if the user or the group is allowed to access the "notification" module.

1. Log in to the administration interface and click the "User accounts" tab. You should see your users and groups on the left.
2. Select the target user/group using the tree or the "Sub items" window.

(see figure 4.121)

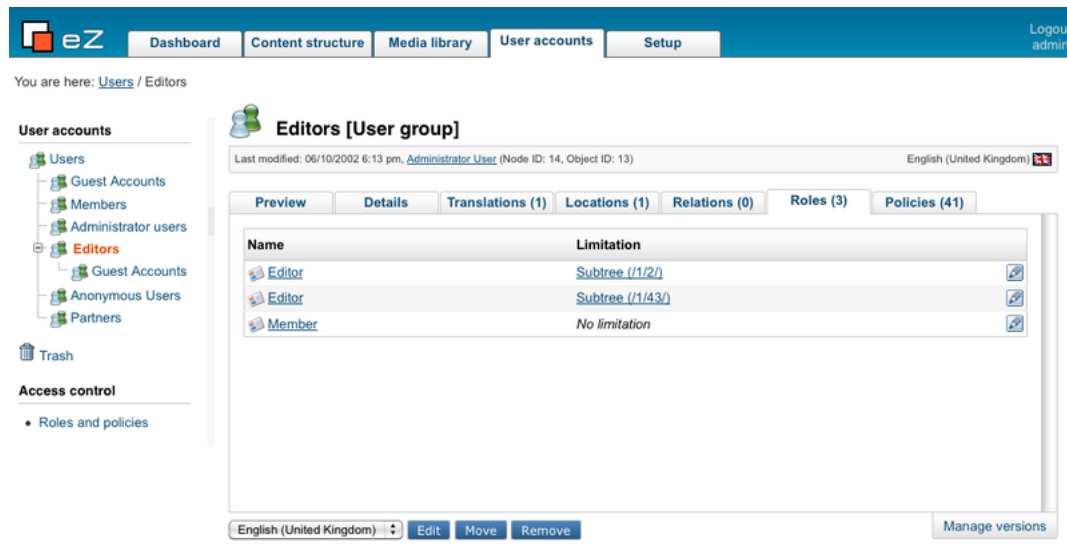


Figure 4.121:

The screenshot above shows a situation when the "Editors" user group is selected. You can bring up a list of roles and policies assigned to this group by enabling the "Roles" and "Policies" windows using the menu at the top.

3. Look at the "Module" column in the table of policies. As long as the "notification" module is not listed here, the selected user/group is not allowed to use notifications. Please refer to the next sections for information about how you can create a new role (that grants access to the module) and assign it to a user/group.

Creating a new role

The following text reveals how you can create a new role for granting access to notifications.

1. Click the "User accounts" tab in the administration interface and then access the "Roles and policies" link on the left. You should see the list of existing roles as shown in the screenshot below.

(see figure 4.122)



Figure 4.122:

2. Let's create a new role called for example "My notification role". Click the "New role" button under the list of roles. You will be taken to the role edit interface as shown in the following screenshot.

(see figure 4.123)



Figure 4.123:

3. Specify the name of the role and click the "New policy" button.
4. The wizard will help you to create a new policy.
(see figure 4.124)

Create a new policy for the <My notification role> role

Welcome to the policy wizard. This three-step wizard will help you create a new policy that will be added to the role that is currently being edited. The wizard can be aborted at any stage by using the "Cancel" button.

Step one: select module

Instructions:

1. Use the drop-down menu to select the module that you want to grant access to.
2. Click one of the "Grant.." buttons (explained below) in order to go to the next step.

The "Grant access to all functions" button will create a policy that grants unlimited access to all functions of the selected module. If you wish to limit the access method to a specific function, use the "Grant access to a function" button. Please note that function limitation is only supported by some modules (the next step will reveal if it works or not).

Module:

Function:


Grant access to all functions Grant access to one function **Grant full access** Grant limited access

OK Cancel

Figure 4.124:


Select the "notification" module from the "Module" drop-down list. Choose the "use" function from the "Function" drop-down list. Please note that you shouldn't choose the "administrate" function because it grants access to the "runfilter" view of the "notification" module.

5. Click the "Grant full access" button. (There is no point clicking the "Grant limited access" button because the functions of the "notification" module do not support limitations.)
6. The new policy will appear in the role edit interface as shown in the following screenshot. (see figure 4.125)

 **Edit <My notification role> [Role]**

Name:

Policies

<input type="checkbox"/>	Module	Function	Limitations
<input type="checkbox"/>	notification	use	No limitations 

Remove selected New policy

Save Cancel

Figure 4.125:

7. Click "Save" to save your changes and go back the role view interface. (see figure 4.126)
-

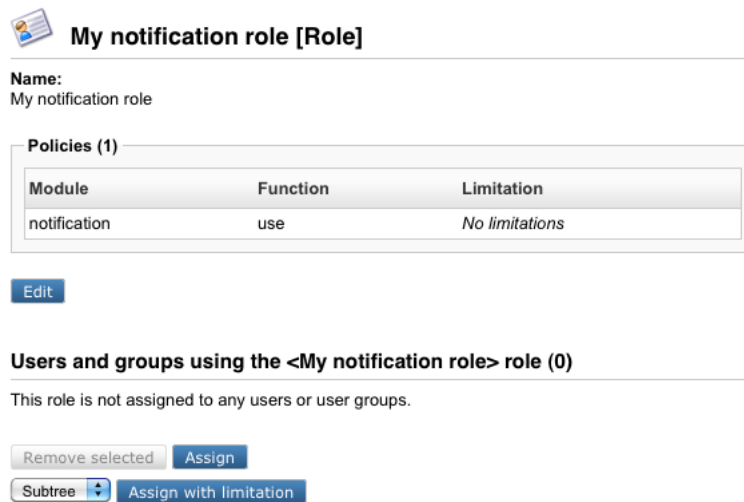


Figure 4.126:

The new policy will appear in the role view interface as shown in the screenshot above. You can now assign this role to any user or group (this is explained in the next section).

Assigning a role to a user and/or a user group

A role can be viewed by clicking on its name in the list of existing roles in the role interface (select "Roles and policies" from within the "User accounts" to bring up the role interface).

When you're looking at a role, there should be a list of users/groups towards the bottom of the page. This list reveals the users and groups that the role which is being viewed has been assigned to. The following text explains how to use this list in order to assign the role that is currently being viewed to the "Editors" user group.

1. Click the "Assign" button located under the list of users in the role view interface.
2. Select the "Editors" user group as shown in the following screenshot and click the "OK" button.
(see figure 4.127)
3. The "Editors" user group will appear in the list of users. The screenshot below shows the role view interface for "My notification role" that is assigned to the "Editors" user group (this means that all users that belong to this group are allowed to use notifications).
(see figure 4.128)

Please note that you can assign the role to a single user in the same way as to a user group.

Browse

To select objects, choose the appropriate radio button or checkbox(es), then click the "Select" button.

To select an object that is a child of one of the displayed objects, click the object name for a list of the children of the object.

^ **Users [6]**

10 25 50 List Thumbnail

<input type="checkbox"/>	Name	Type
<input type="checkbox"/>	Guest Accounts	User group
<input type="checkbox"/>	Members	User group
<input type="checkbox"/>	Administrator users	User group
<input checked="" type="checkbox"/>	Editors	User group
<input type="checkbox"/>	Anonymous Users	User group
<input type="checkbox"/>	Partners	User group

Select Cancel

Figure 4.127:

My notification role [Role]

Name:
My notification role

Policies (1)

Module	Function	Limitation
notification	use	No limitations

Edit

Users and groups using the <My notification role> role (2)

User/group	Limitation
<input type="checkbox"/> John Connor	No limitations
<input type="checkbox"/> Editors	No limitations

Remove selected Assign

Subtree Assign with limitation

Figure 4.128:

4.15.6 Notification events

The following three notification events are supported by default:

- Publish
- Collaboration
- Current time

Publish

Every time an object is published, a new "ezpublish" event is created.

Collaboration

Every time a collaboration message is generated, a new "ezcollaboration" event is created.

Current time

Every time the "runcronjobs.php" script is executed, a new "ezcurrenttime" event is created. This behavior is specified in the "cronjobs/notification.php" file. The system uses "ezcurrenttime" events for generating digest notifications.

If you need to generate this event manually, add the "notification/runfilter" notation to the URL of your site administration interface and then click the "Spawn time event" button. Please note that the "runfilter" view of the "notification" module should be used only for testing and debugging.

The built-in notification event types are stored in the "kernel/classes/notification/event/" directory. It is possible to extend the system by creating custom notification events for special needs.

Creation and storage

Let's say that you have an article on your site, and a user has subscribed for subtree notifications about this article. Every time a new comment is posted or an updated version of the article is published, the system will generate a new "ezpublish" event and store it in the database. This event can be processed by zero, one, or more notification handlers (page [530](#)).

Settings

The available notification event types are specified in the "[NotificationEventTypeSettings]" section of the "notification.ini" configuration file located in the "settings" directory. The following settings can be used under this section:

The "RepositoryDirectories[]" array specifies the directories where eZ Publish will search for built in notification event types. The exact location of the event in the directory is specified using the "AvailableNotificationEventTypes" setting.

The "ExtensionDirectories[]" array specifies the extension directories where eZ Publish will search for additional notification event types. By default eZ Publish will search in the "notificationtypes" subdirectory inside your extension. The exact location of the event in this subdirectory is specified with the "AvailableNotificationEventTypes" setting.

The "AvailableNotificationEventTypes[]" array contains a list of event types.

Example 1

The following lines can be specified in the "notification.ini" configuration file:

```
[NotificationEventTypeSettings]
RepositoryDirectories[]=kernel/classes/notification/event/
ExtensionDirectories[]
AvailableNotificationEventTypes[]=ezpublish
AvailableNotificationEventTypes[]=ezcurrenttime
AvailableNotificationEventTypes[]=ezcollaboration
```

These settings will make eZ Publish search for the following files for built in notification events:

- kernel/classes/notification/event/ezpublish/ezpublishtype.php
- kernel/classes/notification/event/ezcurrenttime/ezcurrenttimetype.php
- kernel/classes/notification/event/ezcollaboration/ezcollaborationtype.php

Example 2

You can extend the system by creating custom notification events. For example, if you have an extension "nExt" that includes a notification event "nev", put the following lines into an override for "notification.ini" configuration file:

```
[NotificationEventTypeSettings]
ExtensionDirectories[]=nExt
AvailableNotificationEventTypes[]=nev
```

or

```
[NotificationEventTypeSettings]
RepositoryDirectories[]=extension/nExt/notificationtypes/
AvailableNotificationEventTypes[]=nev
```

These settings will make eZ Publish expect the additional notification event to be located at "extension/nExt/notificationtypes/nev/nevtype.php"

Please note that you must always clear at least the ini cache in order to make the system re-read the changed configuration files.

4.15.7 Notification handlers

There are several handlers that process notification events. The following handlers are known to the eZ Publish system by default:

- Subtree notification
- General digest
- Collaboration notification

Subtree notification

The "ezsubtree" notification handler processes "ezpublish" events.

General digest

The "ezgeneraldigest" notification handler processes "ezcurrenttime" events.

Collaboration notification

The "ezcollaborationnotification" notification handler processes "ezcollaboration" events.

The built-in notification handlers are stored in the "kernel/classes/notification/handler/" directory. It is possible to extend the system by creating custom notification handlers for special needs.

Processing the notification events

Whenever the "eznotificationeventfilter.php" script is executed, the system will try to run every unhandled notification event (page [527](#)) with every available notification handler.

Please note that handling one event may result in sending/generating more than one notification. For example, if a new version of an article is published, all the subscribed users will be notified.

If every notification is sent successfully, the event will be deleted from the database. Otherwise, if some of the generated notifications must be delayed in order to form a daily/weekly/monthly digest, the system will add new notification items to the user collection. A notification item contains data about the notification event, its handler, subscriber e-mail and time when this notification must be sent.

The digest handler starts processing the "ezcurrenttime" event by accessing the collection of notification items. The time specified in the "ezcurrenttime" event will be compared with the time of each notification item in order to determine which items that must be handled at the moment. As long as each notification item contains data on the notification event and its handler, the system will process this event with the right handler. The resulting notifications will be collected into digest messages and sent to the subscribers.

If the notification item was handled successfully, this item will be removed from the collection. If none of the remaining notification items reference the handled notification event, this event will be deleted from the database.

The "ezcurrenttime" event will be deleted from the database when processing is completed. Please note that processing the "ezcurrenttime" event by the digest handler does not always result in sending/generating digest notifications (for example, if none of the subscribers has chosen the digest mode for notifications).

Settings

The "[NotificationEventHandlerSettings]" section of the "notification.ini" configuration file defines the event handlers that will respond to the notification event. Under this section, the following settings can be specified:

The "RepositoryDirectories[]" array specifies the directories where eZ Publish will search for built in notification handlers. The exact location of the handler in the directory is specified using the "AvailableNotificationEventTypes" setting.

The "ExtensionDirectories[]" array specifies the extension directories where eZ Publish will search for additional notification handlers. By default eZ Publish will search in the "notification/handler/" subdirectory inside your extension. The exact location of the handler in this subdirectory is specified using the "AvailableNotificationEventTypes" setting.

The "AvailableNotificationEventTypes[]" array contains a list of handlers.

Example 1

The following lines can be specified in the "notification.ini" configuration file:

```
[NotificationEventHandlerSettings]
RepositoryDirectories []=kernel/classes/notification/handler/
ExtensionDirectories []
AvailableNotificationEventTypes []=ezgeneraldigest
AvailableNotificationEventTypes []=ezcollaborationnotification
AvailableNotificationEventTypes []=ezsubtree
```

These settings will make eZ Publish search for the following files for built in notification handlers.

- kernel/classes/notification/handler/ezgeneraldigest/ezgeneraldigesthandler.php
- kernel/classes/notification/handler/ezcollaborationnotification/ezcollaborationnotificationhandler.php
- kernel/classes/notification/handler/ezsubtree/ezsubtreehandler.php

Example 2

You can extend the system by creating custom notification handlers. For example, if you have an extension "nExt" that includes a notification handler "nh" put the following lines into an override for the "notification.ini" configuration file:

```
[NotificationEventHandlerSettings]
ExtensionDirectories []=nExt
AvailableNotificationEventTypes []=nh
```

or

```
[NotificationEventHandlerSettings]
RepositoryDirectories []=extension/nExt/notification/handler/
AvailableNotificationEventTypes []=nh
```

These settings will make eZ Publish expect the additional notification handler to be located at "extension/nExt/notification/handler/nh/nhhandler.php"

4.15.8 Frequently Asked Questions

Q: Is there a standard user who automatically get notified about all the site changes (creation/modification of content objects)?

A: By default, none of the users is notified about all the site changes. If you want to be notified whenever content is changed or added, you can subscribe for subtree notifications (page 511) for the top node in the "Content structure" tree.

Q: Is it possible to be notified about new user registrations?

A: You can subscribe for subtree notifications (page 511) for the top node in the "User accounts" tree so that you will be notified every time a new user is created. To do this, click on the "User accounts" tab in the administration interface, locate the desired node (under which new users are created upon user registration) in the tree and select "Add to my notifications" using the context menu.

Q: Is it possible to receive E-mails whenever I need to approve an article? Is it possible that the writer of the article is notified whether or not the article was approved?

A: It is possible to get notifications when you need to approve an item (same for the author). This can be easily done by enabling the collaboration notifications. Currently there is no support for notifications to the author when the the article has been approved/rejected.

Q: I have subscribed for notifications but I do not receive any E-mails.

A: You might have forgotten about the "runcronjobs.php" script. If you wish to use the notification system, this script must be executed periodically.

Q: I use both subtree and collaboration notifications. The subtree notifications work well but I do not receive collaboration notifications.

A: The collaboration notifications are sent every time a new collaboration message is generated. Check your collaboration messages by clicking the "My Account" tab in the administration interface and select the "Collaboration" link on the left. If there are no collaboration messages there, check your collaboration settings by clicking the "Setup" tab and choosing the "Workflows" and/or "Triggers" link on the left. Refer to the "Workflows (page 165)" and "Approve" documentation chapters for more information about workflows, triggers and approval events.

Q: Notification settings are not available for one of my users.

A: A user must be logged in to access the notification settings. If the notification settings are still unavailable after logging in, check the role/policy settings specified for the user(s) as described in the "Granting access to notifications (page 522)" part of the documentation.

Q: Why do the users see the "access denied" page when they click the "Keep me updated" button?

A: Perhaps they are not allowed to use notifications. Check the role/policy settings specified for these users as described in the "Granting access to notifications (page 522)" chapter.

Q: I have the default/built-in forum on my site and I wish that every registered user should be able to subscribe/unsubscribe for subtree notifications about the forum/topic/reply. How can I do this?

A: By default, all the users that belong to the "Guest accounts" user group are allowed to use

notifications. This is specified in the default "Forum user" role that is assigned to the guest user group. It is possible to assign this role to other users (refer to "Assigning a role to a user / user group (page 522)" section of the "Granting access to notifications (page 522)" chapter for more information). There is no point to assign this role to the "Administrator users" group. The default "Administrator" role assigned to the "Administrator users" group allows these users to access all modules including the "notification" module.

Q: In the role/policy settings I can choose the "administrate" function when granting access to the "notification" module. Does it mean that it is possible to view/edit the notification settings of each subscribed user somewhere in the administration area?

A: Although letting administrators to view and/or edit notification settings for all users is probably good idea, it is not implemented yet. The only difference between "use" and "administrate" functions is that the latter grants access to the "runfilter" view of the "notification" module. Note that this view should only be used for testing and debugging.

Q: Is it possible to force digest mode for notifications so that the digest mode is set by default for all the subscribed users (with the preset time)?

A: This functionality is not implemented. By default, the digest mode is off and the database contains no records about this setting. If the user sets the digest mode, it will be recorded in the database.

Q: Is there any way to set "filters" for subtree notifications? I have a set of articles under a certain folder and the users are notified whenever a new article is created there. However, they also receive notifications when an existing article is edited or a new folder is created. I'd like to specify "only notify if a new object of type article is being created" or something similar.

A: This is not supported at the moment.

4.16 Search engine

The system comes with a built-in search engine which integrates tightly with the content structure. It is capable of indexing everything that is inputted through the native content model.

Note: Before you read on, please be aware that eZ Systems has developed a new powerful, enterprise search extension. eZ Find is a search extension for eZ Publish, providing more functionality and better results than the default search in eZ Publish. The main advantages of eZ Find over the default eZ Publish search mechanism are:

- relevancy ranking and keyword highlighting in the search results.
- search results are served from a copy of the search index.
- In addition, the eZ Find search engine remembers all caching structures from previous searches and, during indexing, updates these as well. Therefore, the more the search engine is used, the faster it becomes.

Find more information on eZ Find [here](#).

In eZ Publish, a content class describes the actual data structures (for example news articles, products, etc.). The classes are built up of attributes which are represented by datatypes. An attribute can be the title of an article, the price of a product and so on. It is possible to control which attributes that should be indexed by the search engine. This can be done by making use of the "Searchable" check-boxes while editing a class. Some datatypes (for example float, price, etc.) do not support indexing. Please refer to the datatype overview page to see which datatypes that can be indexed.

When an object is published, the attributes that are marked searchable will be indexed by the search engine. It will then be possible to use the search interface to find words or phrases that are a part of the published content. For example, if the user searches for "backpack", the system will return a list of all kinds of objects where the word "backpack" occurs. This is the default behavior. The following screenshot shows the standard search interface.

(see [figure 4.129](#))

Advanced search

The advanced search interface makes it possible to tweak and narrow the search. The following features are supported:

- Search for several words at the same time (for example "car bike train").
- Search for an exact phrase (for example "cheap cars in Scandinavia").
- Class level filtering (limit the search to a specific class).
- Attribute level filtering (search only a specific attribute).

Search

events

For more options try the [Advanced search](#).

Search for <events> returned 3 matches




Name	Type
 Events	Frontpage
 eZ Awards 2009 winners	Article
 Will you attend eZ events in 2011?	Poll

Figure 4.129:

- Tree level filtering (limit the search to a part of the node tree).
- Section filtering (limit the search to objects that belong to a certain section).
- Time filtering (yesterday, last week/month/3-months/year).

The following screenshot shows the advanced search interface.

(see figure 4.130)

Advanced search

Search for all of the following words:
events

Search for an exact phrase:

Class:

In: Published:

Search for <events> returned 3 matches




Name	Type
 Events	Frontpage
 eZ Awards 2009 winners	Article
 Will you attend eZ events in 2011?	Poll

Figure 4.130:

Wildcard searching

The default behavior of the search engine is that it only searches for complete words or phrases. If the user searches for "demo", the system will not return objects that contain words like "demolition", "demonstration" and so on. However, eZ Publish does in fact support wildcard searching, but it must be turned on by adding the following lines to a configuration override for "site.ini":

```
[SearchSettings]
EnableWildcard=true
```

When the wildcard search feature is turned on, it is possible to use the asterisk character as a wildcard, for example like this: "demo*". In this case, eZ Publish will return a list of objects that contain words starting with "demo". For example, it would return objects containing words like "demonstration", "demolition", etc. When this notation is used, the result will also return objects that contain the word which was specified before the asterisk. In other words, objects containing only the word "demo" will also be returned.

Please note that the asterisk can only be used after a word. This means that the following search queries are invalid: "*demo" and "some*thing".

Warning! There is a good reason for the wildcard search being turned off by default. It requires a lot more processing time than the standard search. This means that the server might have to be upgraded in order to produce faster results and to achieve less overall system load.

Logical operators

Inline logical operators like "AND" and "OR" are not supported. This means that it is not possible to specify search queries like "cars AND minivans" or "trucks OR vans". However, it is in fact possible to do an AND search. This can be done by making use of the "Search for all of the following words" input field in the advanced search interface. For example, if the user inputs "cars bikes" then the system will return a list of objects that contain both of these words. The order of the words is insignificant.

Search statistics

The setup part of the administration interface provides access to a page that reveals information about words/phrases that have been searched along with the average results that have been returned. The following screenshot shows the search statistics interface.

(see figure 4.131)

The "Reset statistics" button will simply clear the search log.

Search statistics

10 25 50

Phrase	Number of phrases	Average result returned
events	2	3.00
"demo"	2	13.00
dashboard	1	0.00

[Reset statistics](#)

Figure 4.131:

4.17 WebDAV

WebDAV is an abbreviation for "Web-based Distributed Authoring and Versioning" (published as an open standard under RFC 2518). WebDAV is a set of extensions to the HTTP protocol which allows users to collaboratively edit and manage files on a web server. This is achieved by making use of a WebDAV compatible client / application. For example, it is possible to use recent versions of KDE's Konqueror or Microsoft's Internet Explorer. Using a WebDAV compatible client, the user connects to the server and is able to browse and manage files in a similar way as with a network share or an FTP server. In other words, what this protocol does is that it makes it possible to browse, create, remove, upload, download, rename, etc. files and directories on a web server. One of the most important advantages of this technology is that it uses port 80 for network traffic. This means that if you are able to surf the site from your workstation, you can also use WebDAV to administer it. It does not require firewall reconfiguration.

eZ Publish and WebDAV

From version 3.2 and up, eZ Publish provides a built in WebDAV server. This implementation allows users to communicate with eZ Publish using a WebDAV compatible client. Once connected, it is possible to browse and manage the node tree of a site. The tree will be displayed as if it were a filesystem (as directories and files).

When a user connects to the eZ Publish WebDAV server for the first time, the system will display a list of the siteaccesses that have been made available for WebDAV. This list can be configured using the "SiteList[]" directive under "[SiteSettings]" in a configuration override for "site.ini". Please note that the system does not ask for a username/password combination at this stage. In other words, anyone with network access will be able to see the names of the available siteaccesses. The following screenshot shows what the user will see if there are two available siteaccesses, "example" and "plain_user".

(see figure 4.132)

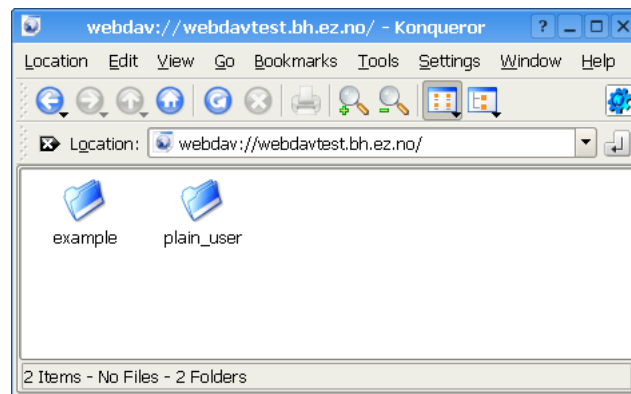


Figure 4.132: WebDAV - Virtual top folder

When a siteaccess is chosen, the system will attempt to authenticate the user by asking for a

username/password combination. The next screenshot shows this.
(see figure 4.133)

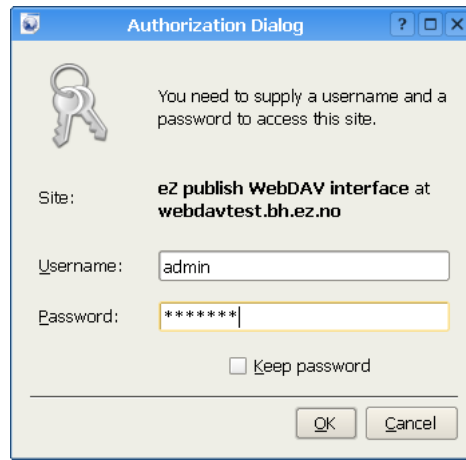


Figure 4.133: WebDAV - Login

The provided use rname and password must belong to a valid eZ Publish user that exists for the selected siteaccess. Furthermore, the user must have sufficient privileges in order to be able to see the contents of the node tree. The following screenshot shows a WebDAV client displaying the contents of the root node of an eZ Publish siteaccess called "plain_user".

(see figure 4.134)

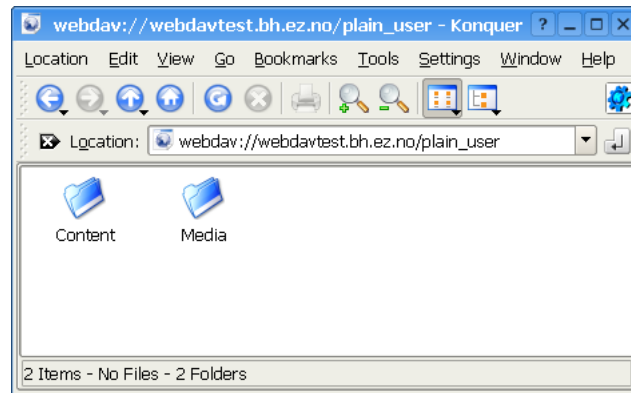


Figure 4.134: WebDAV - Top level nodes

As the screenshot indicates, the user will be able to browse and manage the contents of the "Content" and "Media" top level nodes. The next screenshot shows an example of what the user will see after doing some exploration of the node structure of the "Content" top level node.

(see figure 4.135)

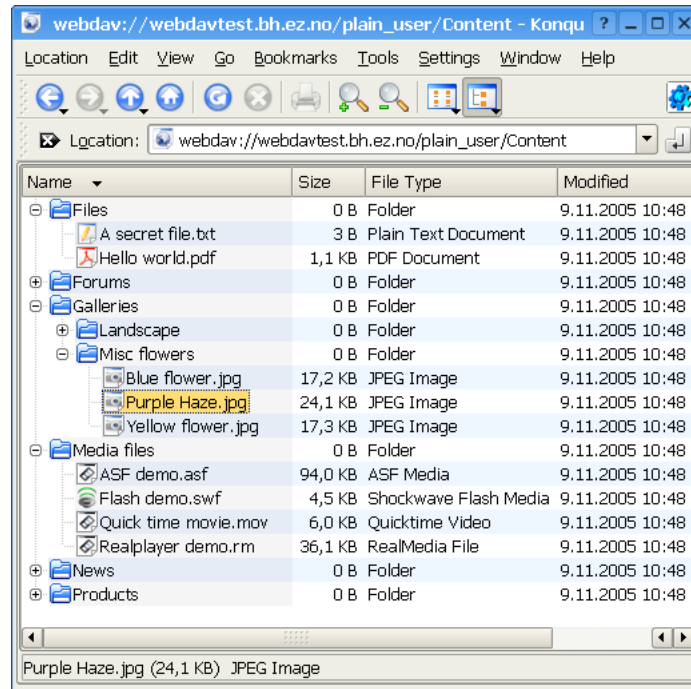


Figure 4.135: WebDAV - Content node tree

Browsing and downloading

The default behavior is that all nodes are displayed as directories. The reason for this is because in eZ Publish any object can be placed under any other object by the way of nodes. Displaying the nodes as directories makes it possible to explore the structure of the node tree. However, not all nodes are displayed as directories.

Nodes that reference objects containing datatypes that store files will be displayed as files instead of directories. This means that for example nodes that make use of the image, media or the file datatype will be displayed as files. When downloaded, eZ Publish will send the file that is contained in the attribute which is represented by a datatype capable of storing a file. If several attributes are represented by such datatypes, it is the contents of the first attribute that will be used.

The "FolderClasses[]" directive in "webdav.ini" can be used to configure which types of nodes that should be shown as directories in the WebDAV client. The default configuration assures that nodes referencing folder objects are always displayed as directories. Adding a class that makes use of a datatype capable of storing a file will result in an override of the behavior described in the previous paragraph. In other words, this setting makes it possible to display different types of nodes as directories even if they contain files.

Uploading

Any type of file can be uploaded to the system. Files will be stored using instances of the file class. In other words, every time a file is uploaded, eZ Publish will create a file object where the file attribute will contain the uploaded data. In addition, a node will be created at the location where the file was uploaded in the tree. This is the default behavior. However, not all file types will be created as file objects.

 It is possible to configure the system so that it creates different kinds of objects based on the type of the uploaded file. For example, the default configuration makes sure that uploaded images are created as image objects. This behavior is controlled by mapping MIME types to classes. The mappings can be configured using the "MimeClassMap[]" directive under "Create-Settings" in a configuration override for "upload.ini". The "DefaultClass" directive determines which class that should be used if there is no suitable mapping. This is usually set to "file", which means that unrecognized file types will be created as file objects. The following example shows the default mappings.

```
MimeClassMap []
MimeClassMap[image]=image
MimeClassMap[video/quicktime]=quicktime
MimeClassMap[video/x-msvideo]=windows_media
MimeClassMap[video/vnd.rn-realvideo]=real_video
MimeClassMap[application/vnd.rn-realmedia]=real_video
MimeClassMap[application/x-shockwave-flash]=flash
```

Each entry in the "MimeClassMap[]" array must be further specified using a block that reveals details about the class that is being mapped to. The blocks must contain the following information:

- The identifier of the target class (appended with "_ClassSettings").
- The class attribute identifier which will get the file data.
- The class attribute identifier which will get the name.
- A pattern that defines the name of the object.

The following example shows the default class map block for images.

```
[image_ClassSettings]
FileAttribute=image
NameAttribute=name
NamePattern=<original_filename_base>
```

The example above will tell eZ Publish that when an image is uploaded, the actual file data should be put into an attribute identified by the string "image". The name of the image should

be stored using an attribute called "name" (as before, it is the identifier of the attribute that is used). The "NamePattern" tells the system about how it should generate the name for uploaded images. It may contain plain text and special tags enclosed by angle brackets ("<" and ">"). The following table reveals the tags that can be used.

Tag	Description
original_filename	The name of the uploaded file, like it was on the local machine (for example "test.jpg").
original_filename_base	The name of the uploaded file without an extension (for example "test").
original_filename_suffix	The extension of the uploaded file (for example ".jpg").
mime_type	The MIME type of the uploaded file (for example "image/jpeg").

Custom upload handling

It is possible to use custom upload handlers in order to process uploaded files in a special way. Custom upload handlers must be provided as extensions. A handler must be automatically triggered whenever a certain type of file is uploaded to the system. This can be done by making use of the "MimeUploadHandlerMap[]" directive under "[CreateSettings]" in "upload.ini". For example, the following line will make sure that uploaded images (regardless of type) are handled by a class called "ezimageuploadhandler" located in "ezimageuploadhandler.php".

```
MimeUploadHandlerMap[image]=ezimageuploadhandler
```

It is also possible to have only a specific type of file be processed by the upload handler. The following example demonstrates how to only handle JPEG images.

```
MimeUploadHandlerMap[image/jpeg]=ezimageuploadhandler
```

The upload handler itself must be put in a directory called "uploadhandlers" in an extension, like this:

```
eZ Publish
|
|-extensions
|  |
|  |-example
|  |  |
|  |  |-uploadhandlers
|  |  |
|  |  |-ezimageuploadhandler.php
```

The following code shows the skeleton of a custom upload handler.


```
class eZExampleUploadHandler extends eZContentUploadHandler
{
    function eZExampleUploadHandler()
    {
        $this->eZContentUploadHandler( 'Example file handling', 'example' );
    }

    /*!
     Handles the uploading of example files.
    */
    function handleFile( &$upload, &$result,
                        $filePath, $originalFilename, $mimeInfo,
                        $location, $existingNode )
    {
        // Implement your import/conversion routine here
        copy( $filePath, "var/cache/example.jpeg" );
    }
}
```

4.17.1 Setting it up

This section describes how eZ Publish can be configured in order to function as a WebDAV server. Please note that the DNS and the web server also needs to be configured.

Step 1: Enable the WebDAV server

The master WebDAV switch must be turned on. Create a global configuration override for "webdav.ini" and make sure that it contains the following lines:

```
[GeneralSettings]
EnableWebDAV=true
```

Step 2: Add the desired siteaccesses

In order to allow WebDAV access for a specific siteaccess, the name of the siteaccess must be specified in the "SiteList[]" array under "[SiteSettings]" in a configuration override for "site.ini". Make sure that the global configuration override for "site.ini" contains the necessary lines. The following example shows how WebDAV can be opened up for a siteaccess called "plain_user" and another one called "example".

```
[SiteSettings]
SiteList []
SiteList []=plain_user
SiteList []=example
```

Step 3: Clear all caches

The eZ Publish part of the configuration is done. Clear all caches in order to make sure that the system uses the updated version of the configuration.

Step 4: Setup a DNS entry

Set up a DNS entry (for example a subdomain) that will be used to access the WebDAV server. The entry must point to the IP address of the web server. For example, if you're using "www.example.com" to access the web pages, you could set up "webdav.example.com" for WebDAV.

Step 5: Configure the web server

There is a file called "webdav.php" in the root of the eZ Publish directory. This file provides the actual WebDAV interface. The web server must automatically execute this file whenever a

WebDAV client sends a command to the server. The following lines show an example of how this can be done in the configuration file of the Apache web server.

```
<Virtualhost 128.39.140.28>
  <Directory /path/to/ezpublish>
    Options FollowSymLinks Indexes ExecCGI
    AllowOverride None
  </Directory>
  DocumentRoot /path/to/ezpublish
  RewriteEngine On
  RewriteRule . /webdav.php
  ServerAdmin admin@example.com
  ServerName webdav.example.com
</VirtualHost>
```

Note: make sure that you have a "NamedVirtualHost" line before the declaratoin of the virtual hosts.

Step 6: Test

Launch a WebDAV compatible client / application and attempt to connect to the server.

Internet Explorer

Recent versions of Microsoft's Internet Explorer (6.0.2800.1106 or later) contain a built-in WebDAV client. The target address must be opened as a web folder.

1. Start Internet Explorer.
2. Access the "File" menu and select "Open", a dialog should appear.
3. Type in the address of the WebDAV server along with a hash ("#") character at the end, like this: `http://webdav.example.com/#`

(see figure 4.136)

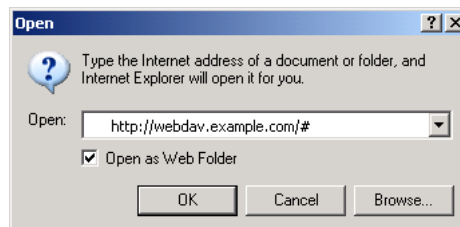


Figure 4.136: WebDAV - IE open dialog

4. Make sure that the "Open as web folder" check-box is checked.
5. Click OK. You should be able to see the available siteaccesses as directories.

KDE/Konqueror

Make sure you have a recent version of Konqueror (3.1.3 or later). Open up a Konqueror window and attempt to browse the WebDAV server by accessing it using a URL that resembles the following example: "webdav://webdav.example.com/".

(see figure 4.137)

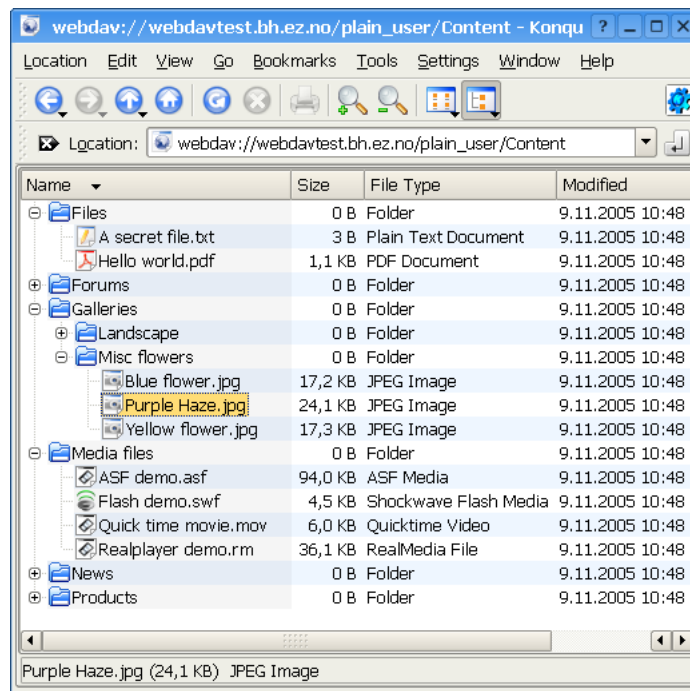


Figure 4.137: WebDAV - Content node tree

4.18 User defined Object States

The introduction of freely definable object states, possibly grouped in object state collections and coupled to the role/policy system of eZ Publish, enables a wide range of applications including typical workflow processes. By default eZ Publish has only one content object state group: "Lock" with object states "Locked" and "Not locked". But by creating freely definable object states groups with their own set of object states, the possibilities of the application can be more specifically tailored to user needs.

How does it work?

Object states are typically used in workflows and other workflow-like processes. It's useful to note that although they are used in workflow processes, not all workflows use object states, nor do object states employ the eZ Publish workflow engine or the workflow system themselves. In this article the term "workflow" should generally be understood as the succession of steps rather than an eZ Publish concept.

A workflow defines an organized sequence of actions that will be executed while the workflow is running. A simple example of a workflow could be a newly written article (object) that is sent for approval to an editor. Since by default eZ Publish only has one content object state group, "Lock", the status of this article or object can only change from unlocked to locked during this workflow process. And once an object has been locked and submitted into such a workflow process, the object remains locked until it is either rejected or accepted. So during the time between submitting an object and its approval no further editing or correcting can be done.

But the possibilities of such workflow processes can be broadened simply through the use of freely defined object states. This allows for more states to be added, thus proving more steps in the process. By attaching policies to each state you are able to control who can do what to an object while it is in a specific state.

Because the states of an object changes throughout the work process, the transition between the states of objects must also be governed through specifically assigned policies to define who is allowed to change the state. Simply put: policies must be assigned to both the states individually, as to each state change. Keep in mind that an object can have multiple object states, but can only have one object state per object state group. [Learn more about the use of object states in an editorial workflow...](#)

Limitations

Some limitations of this system must be taken in consideration. First of all, freely defined object states are assigned on object level, therefor they operate on published versions only, not on translations or drafts. The second limitation is that the roles and policies, that are at the heart of this system, are not yet elaborate enough to cover all possibilities.

Not so much a limitation as it is an observation, that, as mentioned before, object states can be used in workflow processes, but not all workflows use object states.

Why not sections?

Sections are also often used to imitate workflow object states. Although the use of sections also makes it possible to limit and control access to content through roles and policies, it isn't capable of defining policies on state changes. Whereas user definable object states can grant policies to both the states themselves as to the transitions between states. This being said sections are still used for secure sites, where for example anonymous users are only allowed to see the default view.

How to create Object States

1. To create custom object states you must first create a Content Object States Group to place them in. This can be done in the Administration Interface, under the Setup-tab.

 - (a) Click on the "States"-link in the left menu. Here you'll be able to create a new Content Object States Group.

2. In the newly created Content Object State Group you can then create object states matching the different phases of your work process by clicking "Create new". The first state in the group will be applied by default to all newly created objects.
3. Next you must give the correct rights to each user or user group so they can do their part of the task and only their part, before handing over the content to the next group or user:

 - (a) Roles and policies can be created or modified through the User Accounts-tab. In the left menu, under "Access control" you'll find the "Roles and Policies"- link. Note that you can also be redirected to this location via the "Roles and Policies" link in the Setup-tab.
 - (b) For each user group associated to this work process, specific roles and policies must be defined to limit their access and contribution to their specific tasks.
 - (c) Furthermore roles must be assigned to control the state transitions between each phase of the workflow.

4. Now that only authorized users can modify the state of an object and push it through the workflow, a handover-channel should be created between the different contributors. This implies that user(group)s should be notified when an object is ready for them to fulfill their task. However, this feature is not yet included by default in eZ Publish 4.1, but soon will be. For now this must be done manually by sending the URL of the object to the next user(group) in line.

An example

It can be useful to implement a private waste bin and for this object states can easily be defined. In a private waste bin an owner can place articles without deleting them permanently, but still preventing others to read or edit them. The simplicity of removing an article out of the personal waste bin and making it visible for others is an added advantage. In this implementation an object also retains its node location(s).

How to create a personal waste bin step-by-step

To create a private waste bin through the Administration Interface, navigate to the Setup-tab. In the left menu, click on the the "States"-link. Here you will see a list of already existing content object state groups (by default you'll only find "Lock") and you will be able to create a new content object state group:

(see figure 4.138)



Figure 4.138:

For this example a group "Private Wastebin" was created, which includes two object states: "Viewable Article" and "In Private Wastebin". Obviously the first allows an object to be viewed by all users, whereas the latter only allows the owner to view and edit the object. Note that the first state will always be applied as default state.

(see figure 4.139)

Providing the correct rights to the owner is the next step. This must be done via the "Roles and Policies"-link, which can be accessed both under the left menu "Access control" in the "User Accounts"-tab, as through the "Roles and Policies"-link in the "Setup"-tab. Beneath the list of roles you can create a new role. The role "Owner" is added and includes policies that only allow the owner of an object to edit and read the content of an article when it has the object state "In Private Wastebin". Furthermore the owner is given the sole right to control the state transitions. Next all user(group)s are assigned to the role "Owner".

(see figure 4.140)

Private Wastebin [Content object state group]

ID	Identifier	Name
3	private_wastebin	Private Wastebin

Description
This group allows you to place your documents in your private wastebin without deleting th

[Edit](#)

Translations [1]

Language	Loc
English (United Kingdom)	eng

Object states in this group [2]

<input type="checkbox"/>	ID	Identifier	Name	Object cou
<input type="checkbox"/>	3	viewable_article	Viewable Article	130
<input type="checkbox"/>	4	in_private_wastebin	In Private Wastebin	0

[Remove selected](#) [Create new](#)

Figure 4.139:

Finally the policies in the Anonymous role must also be altered in order to prevent anonymous users from viewing articles in "In Private Wastebin". In the policy regarding reading content the limitation of "State_Group_Private_Wastebin" is set to "viewable article".

(see figure 4.141)

Owner [Role]

Name:
Owner

Policies [4]

Module	Function	Limitation
content	create	No limitations
state	assign	Owner(Self)
content	edit	Owner(Self) , StateGroup_private_wastebin(In Private Wasteb
content	read	Owner(Self) , StateGroup_private_wastebin(In Private Wasteb

Edit

Users and groups using the <Owner> role [3]

<input type="checkbox"/>	User/group	Limitation
<input type="checkbox"/>	Members	No limitations
<input type="checkbox"/>	Editors	No limitations
<input type="checkbox"/>	Anonymous Users	No limitations

Remove selected **Assign**

Subtree **Assign with limitation**

Figure 4.140:

Use of private waste bin

After an article has been created and sent for publishing in the User Interface, the owner will be able to define the object state.



Module	Function	Limitation
content	read	Section(Standard) , StateGroup_private_wastebin(Viewable Article

Figure 4.141:

(see figure 4.142)



Content object state group	Available states
Private Wastebin	Viewable Article
	Viewable Article
	In Private Wastebin

Figure 4.142:

This can also be done with an already published article:

(see figure 4.143)

In the Administrator Interface, the state can be altered when you navigate to the "Content Structure"-tab. When viewing an article here, make sure the "state assignment widget" at the top of the page is enabled. If so, you'll be able to define the object state.

(see figure 4.144)

For another example of the use of object states, please read "[eZ Publish Knowledge Series: Editorial workflow with Object States](#)".

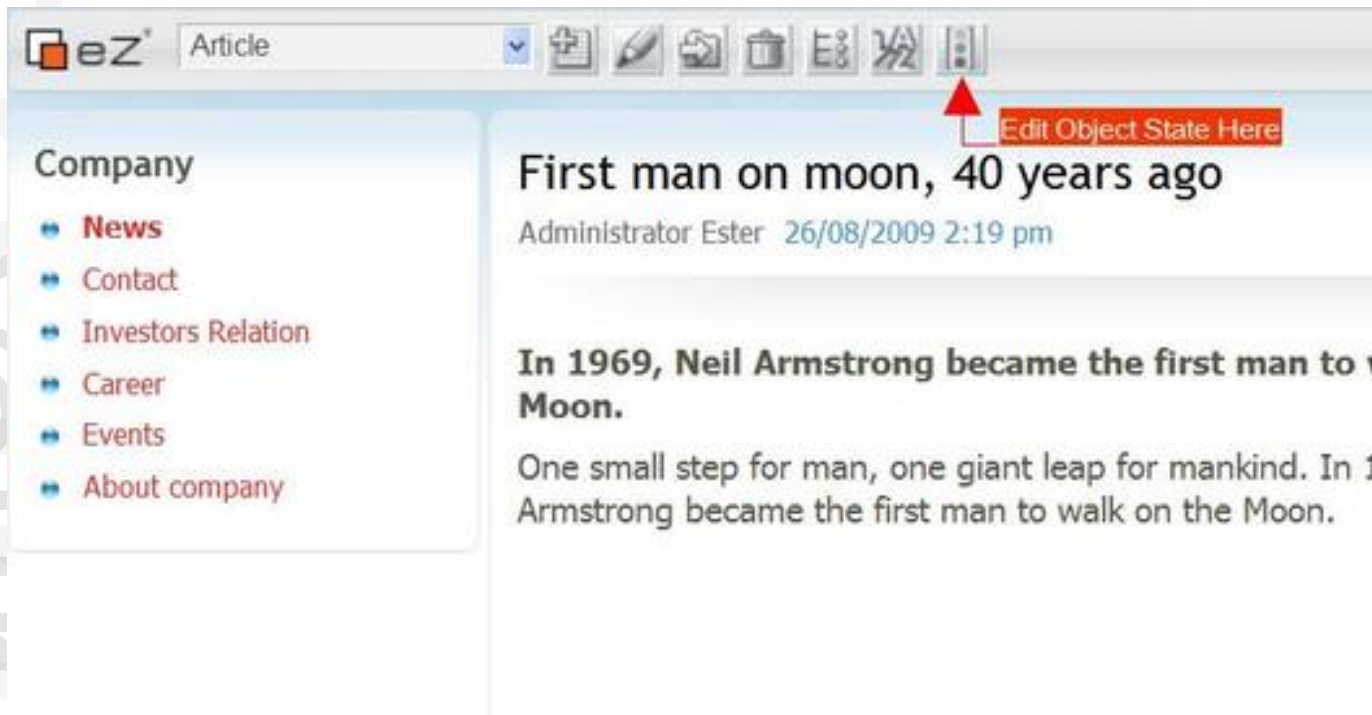


Figure 4.143:



Figure 4.144:

4.19 Language switcher

Since eZ Publish 4.1, there is a new way to create links that take the users to an other translation siteaccess. This had previously been problematic, when the URL was not available in the selected translation, because of differences in translation.

The URLs used with the language switcher have the following form:

```
/switchlanguage/to/<destinationTranslationSiteaccess>/<nodeID|URLAlias>
```

This feature will most often be used via the bundled module switch language. This is an optimized way to translate node IDs and URL aliases into the destination language, for it is done before someone clicks on a language switcher link. This is something which has to be calculated per URL otherwise.

Operation

When a user clicks on a language switcher URL, the user is redirected to an URL, representing the destination siteaccess and the destination content. More often than not the destination content is the translated content originally being viewed, but there are some exceptions to how this works. Here is how:

Origin	Destination	Destination URL
Node	Translated node	Translated URL alias
Node with no available translation [#]_	Original node	Translation siteaccess, same URL Alias
Module	Module	The URL is passed through for modules

Note: [#] Nodes can still be viewable in the other siteaccess. For instance, English content can be viewable in the Norwegian siteaccess if English is in the prioritised languages.

Use

Using the language switcher tool, is simply a matter of creating links in the pattern shown above. The links are easy to create with the template language, but there is also another convenience method for creating language switcher URLs (for which you should only need a minimal amount of logic in the templates). You can access this feature via the template operator, language_switcher(). See the next section how to configure this operator.

The operator will return an array, indexed by siteaccess name, containing a language switcher URL and a human readable text.

```
{def $translations = language_switcher( $module_result.content_info.node_id )}
{foreach $translations as $siteaccessName => $lang}
{*
```

```
Note we are not using the $siteaccessName index here, but useful
for matching against current siteaccess for instance
*}
<a href={$lang.url|ezurl}>{$lang.text|wash}</a>
{/foreach}
```

The input to "language_switcher(\$var)" can either be the node ID, the current URL alias or in the case of a module, just the requested URL.

Configuration

In order to make the language switcher easy to use within templates, some configuration must be done via settings. This is one of the most convenient places to define the translation siteaccess on the site, and this will allow the €language_swither()€ operator to not use normal siteaccesses when creating the links automatically.

Below is an example of what a setup can look like:

```
site.ini

[RegionalSettings]

# Example mapping between translation siteaccesses and the name to use for the
# language switcher link, e.g. the name which will be used when making links to
# these siteaccesses.
# Example: TranslationSA[<name of siteaccess>]=<name of language switcher
link, pointing to this siteaccess>
TranslationSA[]
TranslationSA[eng]=English
TranslationSA[nor]=Norwegian
TranslationSA[fre]=French
```

Override possibility

Each site has a unique set of requirements, that is why the language switcher class can be fully overridden with a custom class.

The class is configured at site.ini:

```
site.ini

[RegionalSettings]

LanguageSwitcherClass=ezpLanguageSwitcher
```

The implementation class needs to implement the ezpLanguageSwitcherCapable interface.

Extra

The language switcher feature provides one more convenience for template developers, namely the ability to fetch the URL alias of a node in a user-selected translation. This is achieved through the following template fetch function:

```
{def $demo = fetch( 'switchlanguage', 'url_alias', hash( 'path',  
$node.url_alias, 'locale', 'nor-NO' ) ) }
```

The above example will give the URL alias for the Norwegian translation, if the URL alias cannot be fetched, or does not exist for the given translation "false" is returned.

The full list of parameters are:

```
fetch( 'switchlanguage', 'url_alias', hash( [ 'node_id', <node_id>, ]  
[ 'path',  
<node_path> ] ) )
```

4.20 Queue handling in asynchronous publishing

As of version 4.5 eZ Publish is set up with a new and improved system, eZ Asynchronous Publishing, for deferring publishing requests to a queue for asynchronous handling of the published objects. The feature aims at increasing the number of objects that can be published concurrently. This is typically a function used in an advanced environment with a large simultaneous load on the database.

A daemon handles the publishing requests and the queue in the background. As a user you will only get information as to the status of your published objects, all processing of the queue is automatic and no user interaction/intervention is possible.

Feature limitation: The asynchronous publishing is not supported on the Windows platform.

4.20.1 Enable the publishing queue functionality

Asynchronous publishing is disabled by default and needs to be enabled by overriding a setting in content.ini. Please enter the setting as described below:

```
[PublishingSettings]
# Enable/Disable the asynchronous publishing feature
AsynchronousPublishing=enabled
```

Customize the queue size

In the PublishingSettings you can also the size of the queue by deciding how many parallel publishing operations should be allowed:

```
[PublishingSettings]
# how many parallel publishing operations should be allowed
# default: 10
PublishingProcessSlots=10
```

Making the feature optional

You disable the eZ Asynchronous Publishing feature through the same INI setting in content.ini:

```
[PublishingSettings]
# Enable/Disable the asynchronous publishing feature
AsynchronousPublishing=disabled
```

If disabled, the operation won't be deferred to the daemon at all, and publishing will happen in real time. This also allows siteaccess based enabling/disabling of the feature.

4.20.2 Setting up the asynchronous publishing service with an init script

This is the method that we recommend on production. Standard init scripts are provided, and can be used to have the daemon started on boot.

Create a symbolic link to the script matching your platform in `/etc/init.d/`. You can choose the name you want for the symbolic link. This allows you to setup multiple daemons on your server for multiple sites.

- For Red Hat Enterprise Linux (RHEL), CentOS, Fedora, SUSE:

```
bin/startup/rhel/ezasynchronouspublisher
```

- For Debian and Ubuntu:

```
bin/startup/debian/ezasynchronouspublisher
```

This is an example for Debian:

```
cd /etc/init.d
ln -s /path/to/ezpublish/bin/startup/debian/ezasynchronouspublisher
chmod +x ezasynchronouspublisher
```

The daemon can now be started using:

```
/etc/init.d/ezasynchronouspublisher start
```

The daemon options are listed by:

```
/etc/init.d/ezasynchronouspublisher help
```

The daemon can now be configured to start up on boot using:

```
chkconfig ezasynchronouspublisher on
```

Customized variable settings for init scripts

The init scripts come with default settings that should match most platforms. If your web server user is not the platform's default and/or the PHP CLI executable is not part of the web server user's PATH, this can be configured.

First, copy the default configuration file for your platform to the correct directory for your OS:

- RHEL:

```
/etc/ezasynchronouspublisher.conf
```

- Debian:

```
/etc/default/ezasynchronouspublisher.conf
```

Example for Debian/Ubuntu:

```
cp bin/startup/<platform>/ezasynchronouspublisher.defaults /etc/default/ezasynchronouspublisher.conf
```

Then edit this file, uncomment and set the required variables to the required values as according to the comments in the settings file.

In the case where both the file "ezasynchronouspublisher.defaults" and "ezasynchronouspublisher.conf" exists, the .conf file will have precedence.

Start your information

4.20.3 Starting the daemon manually

The daemon can be started manually by running the following, as your web server user:

```
ezroot$ php bin/php/ezasynchronouspublisher.php
```

The script will run interactively. To start it in daemon mode (so that it actually detaches from the current session and will keep running even if you log out), the `-n` flag can be added:

```
ezroot$ php bin/php/ezasynchronouspublisher.php -n
```

4.20.4 Various configurations

There are limitations as to how the daemon can be setup in different configurations. The basic principle is one daemon per database and it can be summarized like this:

- **One server - one eZ Publish installation - one database - one daemon:** In this configuration there should be one daemon monitoring one publishing queue.
- **One server - separate eZ Publish installations - separate databases - one daemon per database:** In this configuration there should be one daemon per database. Each daemon is started by a separate script in /etc/init.d/
- **Several servers - one eZ Publish installation - one database - one daemon:** In a clustered configuration there can only be one daemon set up on one of the servers. One publishing queue, one daemon.

4.20.5 Publishing objects asynchronously

When you have enabled "Asynchronous publishing" in "content.ini" and started the daemon, the system will handle your publishing requests as described below.

When you have edited your object in the "Object edit interface" and pressed "Send for publishing", the request to publish the object is queued. This queue is database based and monitored by the daemon, "ezasynchronouspublisher", to know what operations are to be performed. On your screen you can follow the status of the publishing process.

You will get a message confirming the publishing: "Your content is being published" and then presented with the current status of the queued item. There are three status types:

- Pending - The publishing request is now in a queue of publishing requests
- Working - The request is processed to find a queue of publishing request and a place in the queue
- Published - The publishing request has been processed and your object is published

If something irregular happens in the publishing process the system will issue error messages according to the situation.

4.20.6 Make prioritizations in the queue via filtering hooks

Filtering hooks

The asynchronous publishing system comes with filtering hooks that can easily be implemented in order to prevent items from being published asynchronously.

You can configure as many filters as you want, as PHP classes. Each filter will be called sequentially, and will either accept the object for asynchronous publishing, or reject it. If a filter rejects an object, filters processing will be stopped, and the object will be published synchronously.

Priority handling

By default, Asynchronous Publishing will handle publishing operations in a very simple FIFO fashion. First item in, first item out. The system will work well in common situations, there are, however, a few cases where you don't want this. For instance, if you have regular imports of massive amount of content, you clearly don't want the publishing queue to be blocked for X minutes by the import, leaving your editors in despair while they wait forever for their content to be published.

The hook system uses the SignalSlot eZComponent. By reading the documentation for it, you will notice that you can attach multiple calls to any hook by simply calling several times the connect method. Hooks will be called in the order they were registered.

Hooks

- Implementation of a filtering hook
- preQueue hook
- postHandling hook
- queueReader handler

Implementation of a filtering hook

In order to implement a filter, you must create a class that extends the "ezpAsynchronousPublishingFilter" abstract class.

This class implements the "eZAsynchronousPublishingFilterInterface" interface, and enforces the definition of the "accept()" method. The "accept()" method must return a boolean. "True" means the object can be published asynchronously, while "False" instructs the system to skip the asynchronous publishing feature, and publish the content directly.

The abstract class provides you with the "eZContentObjectVersion" being published, as the "\$version" class property. From this property, you can easily test the content object's property (publishing time, author, content class, section), read attributes, and so on.

Example:

```
<?php
;/**
 * Exclude from asynchronous publishing any object that isn't an article
 * @return bool
 */
class eZAsynchronousPublishingClassFilter extends
ezpAsynchronousPublishingFilter
{
    public function accept()
    {
        $contentObject = $this->version->contentObject();
        return in_array( $contentObject->attribute( 'class_identifier' ),
$this->validClasses );

    }

    private $validClasses = array( 'article' );
}
?>
```

The class above will only publish asynchronously objects of class article.

Settings

Each filter must be registered using INI settings from content.ini. Below is an example for the class filtering class above:

```
[PublishingSettings]
AsynchronousPublishingFilters []=eZAsynchronousPublishingClassFilter
```


One line similar to the one above must be added for each filter.

Source
information

preQueue hook

These hooks are called right before an object is sent to the publishing queue. Using these, you can freely read & write data based on the operation: content type, author, publishing time, you choose. The system doesn't provide any means to store extra data. That means you must design your own system for that, but on the other hand, this gives full flexibility over your mechanisms.

Add `MyPriorityHandler::queue()` before the item is queued::

```
ezpContentPublishingQueue::signals()->connect( 'preQueue',  
'MyPriorityHandler::queue' );
```

Hooks can also be attached using INI settings if you want the call to be made every time::

```
[PublishingSettings]  
AsynchronousPublishingQueueHooks []=MyPriorityHandler::queue
```

These hooks are given two parameters:
* the content object id (integer)
* the content object version (integer)
They're not supposed to return any value.

postHandling hook

These hooks are called right after an item has been processed by the publishing queue. They can for instance be used to delete / update data created in a preQueue hook.

Add `MyPriorityHandler::cleanup()` after the item has been handled::

```
ezpContentPublishingQueue::signals()->connect( 'postHandling',  
'MyPriorityHandler::cleanup' );
```

Hooks can also be attached using INI settings if you want the call to be made every time::

```
[PublishingSettings]  
AsynchronousPublishingPostHandlingHooks [] =MyPriorityHandler::cleanup
```

These hooks are given three parameters:

- the content object id (integer)
- the content object version (integer)
- the publishing process status (as one of the `ezpContentPublishingProcess::STATUS_*` constants)

They're not supposed to return any value.

queueReader handler

This handler is the one in charge of reading from the queue and deciding what content is given to the queue handler. A default queueReader handler is provided with your default eZ Publish distribution. It simply reads the oldest inactive item from the processes queue, and returns it.

It can be changed to a new one by implementing the "ezpContentPublishingQueueReaderInterface" and configuring the new class in "content.ini".

Here is a possible class definition::

```
<?php
class myPublishingQueue implements ezpContentPublishingQueueReaderInterface
{
    public static function next()
    {
        // do your deeds...
        if ( $process )
            return $process;
        else
            return false;
    }
}

?>
```

And the matching INI (content.ini) configuration:

```
[PublishingSettings]
AsynchronousPublishingQueueReader=myContentPublishingQueue
```

Chapter 5

Reference

Reference

/eZ-Publish/Technical-manual/4.x/Reference

Shyare

information