eZ

eZ Publish 4.x

Technical Manual

Contents

Contents   3

Contents

Contents 7

Contents   9

Contents

Contents   12

Contents

Contents

List of Figures

"4.x" is mostly documentation specifically for eZ Publish 4.0, only *reference section (page 512)* is common for all eZ Publish 4.x versions as well as eZ Publish 5.x "LegacyStack", please select the version you are using for the most up to date documentation!

Chapter 1

Installation

This part of the 4.x documentation is for eZ Publish 4.0, only *reference section (page 512)* is common for all eZ Publish 4.x versions as well as eZ Publish 5.x "LegacyStack", please select the version you are using for the most up to date documentation!

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 29) 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 43) section.

Automated installation

This installation method (also named kickstart) 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" (page 54) section.

1

1.1   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 30). 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 35)" and "Installing eZ Publish on Windows (page 39)" sections (depending on the target OS) will take you through the necessary steps.

1.1.1   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.

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.x branch. 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 the latest version of the 1.3 branch. (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 preprocessor) 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 5.2.5 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 60) 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 463), 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 (page 2418) 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). This requires eZ Components version 2007.1.1 or higher to be installed. 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 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 293) 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" and "Installing eZ Publish on Windows" documentation pages (depending on the target OS) for more information.

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 then 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.

1

1.1.2   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 30)" 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 "postgresql").
<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".

Downloading eZ Publish

The latest stable version of eZ Publish can be downloaded from http://ez.no/download/ez_publish.

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 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.

Home directory example

If eZ Publish was unpacked to a web-served directory located inside the home directory of a user with the username "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 setup wizard (page)" section for a detailed description of the web based setup wizard.

1.1.3 Installing eZ Publish on Windows

The requirements for doing a normal installation must be met! Read the "Requirements for doing a normal installation (page 30)" 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

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:

1

```
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 "postgresql").
<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://ez.no/download/ez_publish. 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 60)" section for a detailed description of the web based setup wizard.

1

1.2 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 29). 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.

1.2.1   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 30) section.

1.2.2   Manual installation on a Linux/UNIX based system

The requirements for doing a manual installation must be met. Please read the previous section (page 44) 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 35) section. The last step is explained under the "Manual configuration of eZ Publish" (page 47) section.

1.2.3 Manual installation on Windows

The requirements for doing a manual installation must be met. Please read the previous section (page 44) 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 39) section. The last step is explained under the "Manual configuration of eZ Publish" (page 47) section.

1.2.4 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/*

Note that due to a bug in eZ Publish 4.0.0, the modfix script did not alter the permission settings for the "autoload" directory. This problem was fixed in eZ Publish 4.0.1.

If you know the user and group of the webserver 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 webserver 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 (page 2323) 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 (page 138) 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 (page 138)" and "Configuration (page 136)" 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 "SiteLanguage-List (page 2423)" 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 (page 2431)" setting and thus only the default language will be shown. Refer to the "Configuring the site languages (page 249)" 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 (page 1784)" 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 login and password

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

- Username: admin

- Password: publish

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

1.3   Automated installation

The automated installation method (also known as "kickstart") is for experienced users. It provides an automated version of the "Normal installation method" 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.

1.3.1   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 30) 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.

The next section (page 56) explains how eZ Publish can be configured to do an automated installation of itself.

1.3.2   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" (page 35) or "Installing eZ Publish on Windows" (page 39) for information about the first three steps (database setup, download and unpacking). The rest of the steps are explained below.

Configuring the kick-start 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 ar 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.

1

1.4   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).

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).
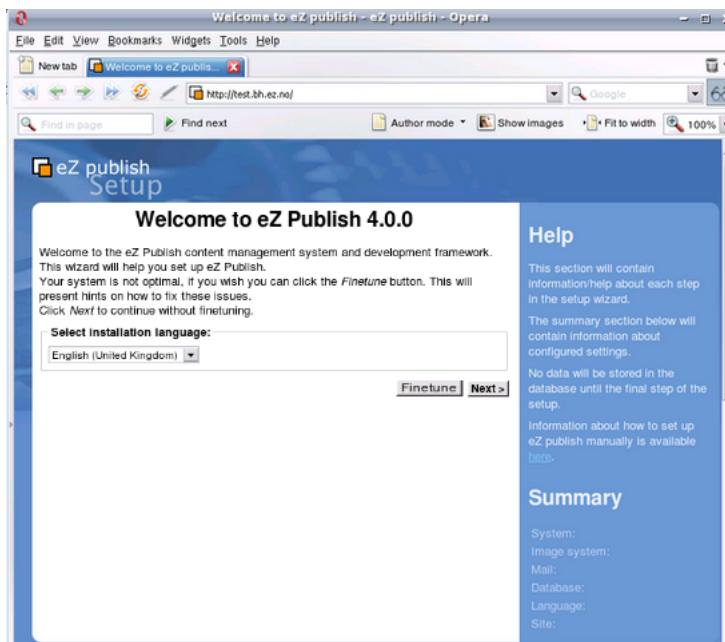
Figure 1.1: *Step 1: Welcome page*

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).
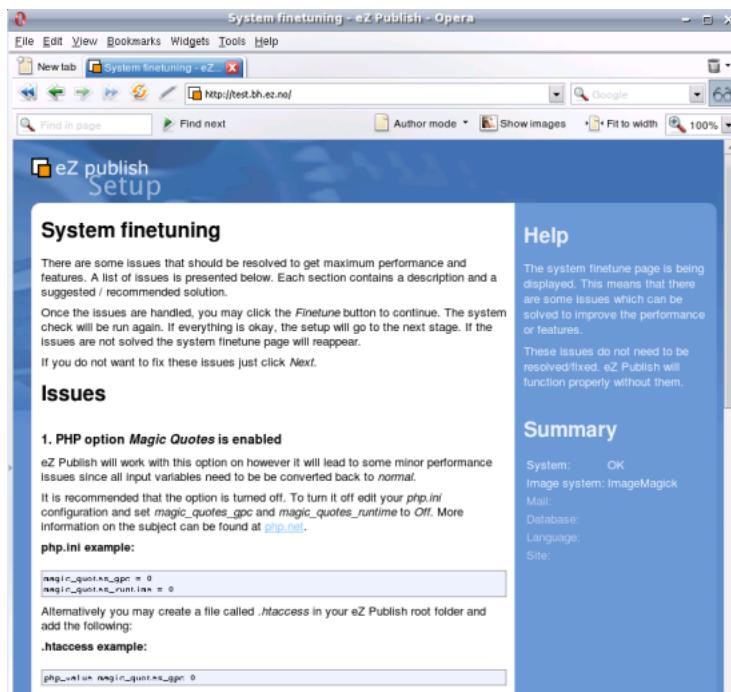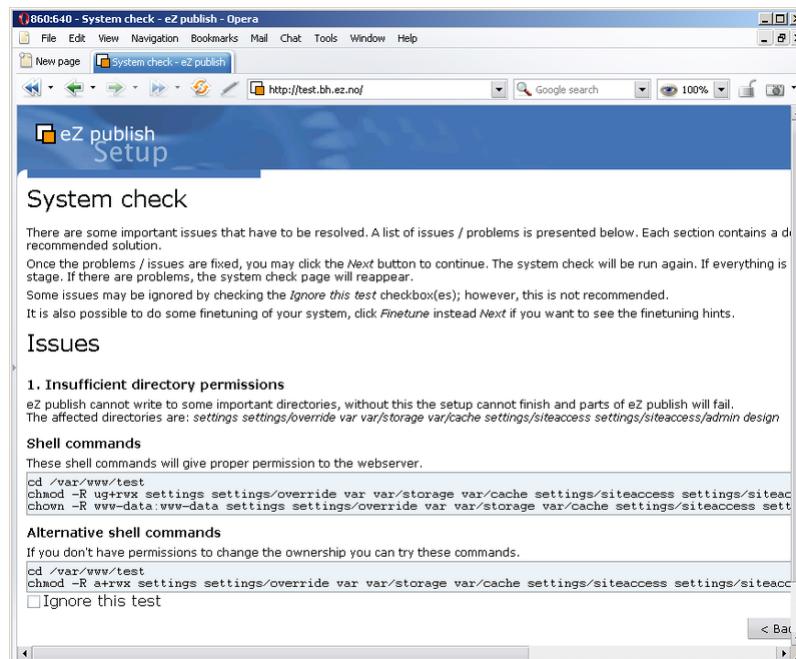
System check

(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.

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

Figure 1.2: *System finetuning*

Figure 1.3: *Step 2: Issues*

(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 checkbox labelled "Ignore this test". However, it is recommended to fix all issues rather than ignoring them.

Outgoing E-mail

(see figure 1.4)
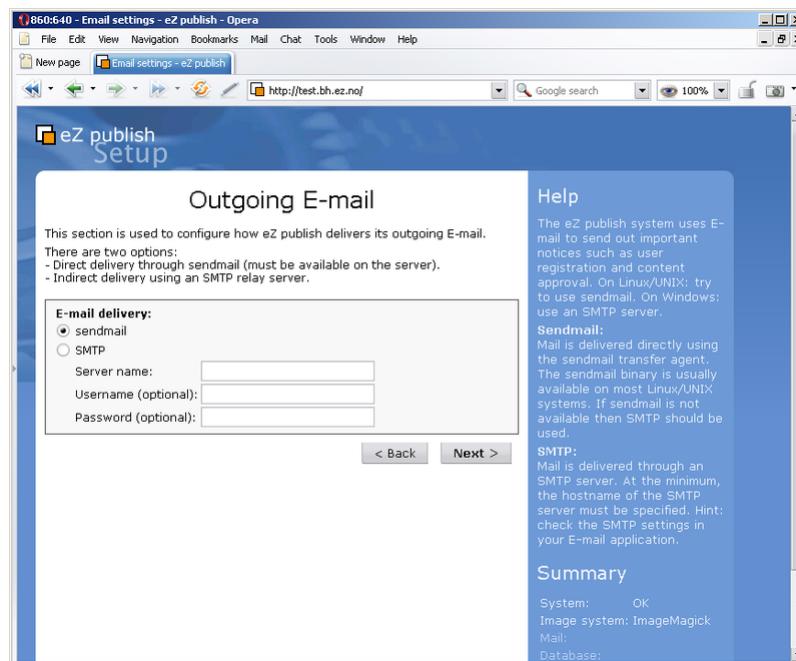
Figure 1.4: *Step 3: Outgoing E-mail*

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)
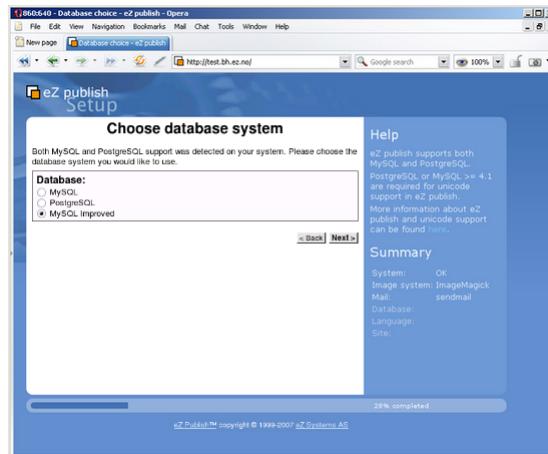
Figure 1.5: *Step 4: Database choice*

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)
Information about the hostname of the server running the database engine, and a username/ password combination needs to be provided. After you click "Next", if MySQL or MySQL Im-
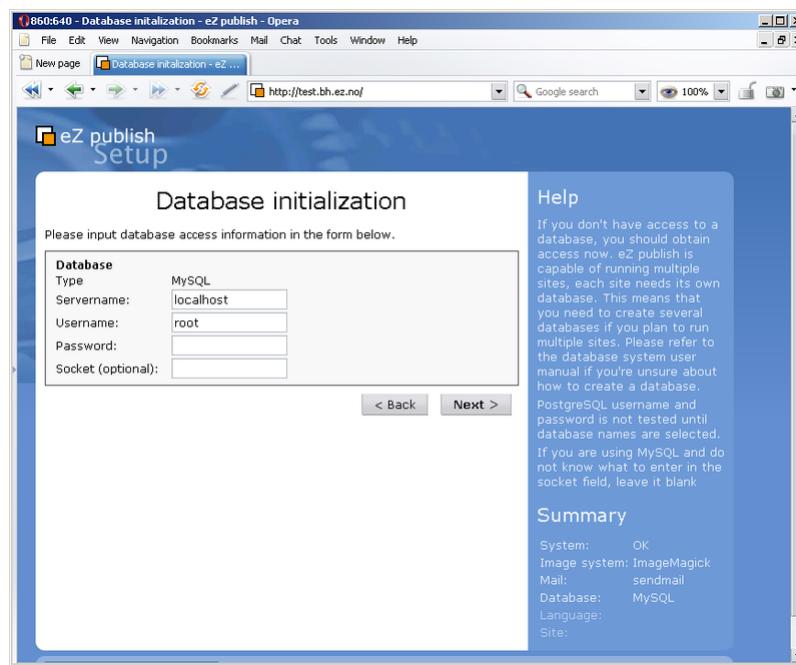
Figure 1.6: *Step 5: Database initialization*

proved 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 username/ 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.)

Language support

(see figure 1.7)
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 checkboxes 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 (page 245) 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.
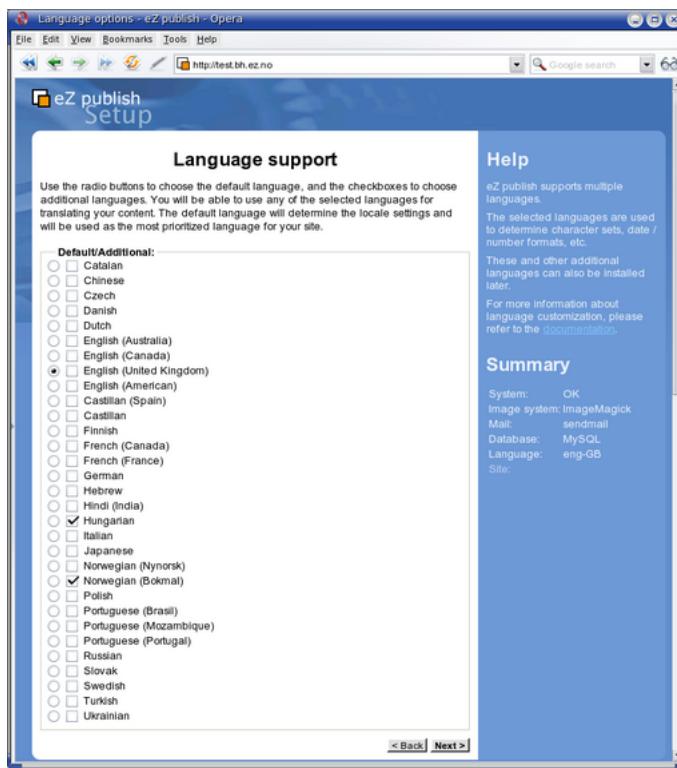
Figure 1.7: *Step 6: Language support*

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)
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
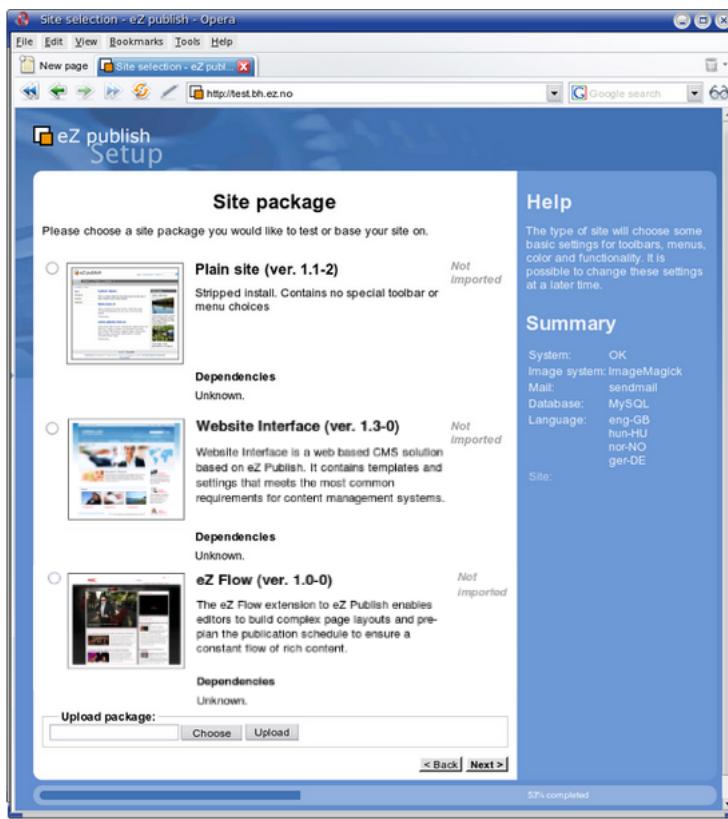
Figure 1.8: *Step 7: Site selection*

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.)

(see figure 1.9)

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
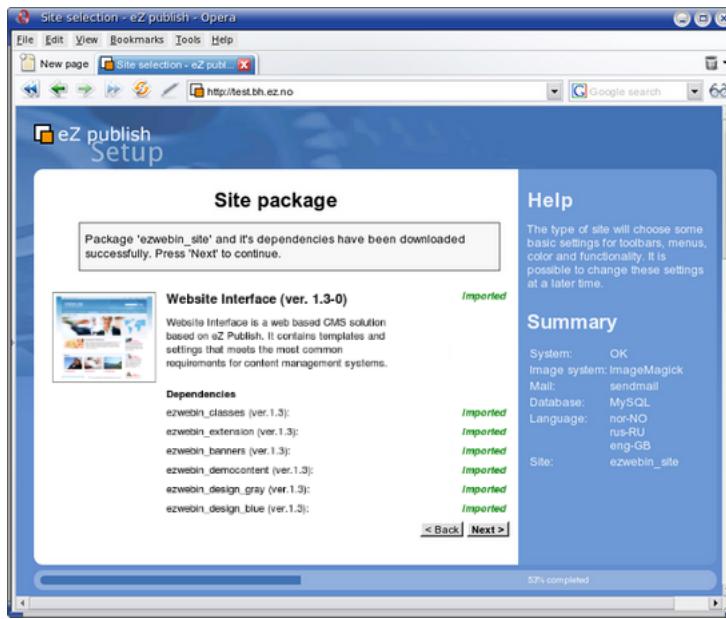
Figure 1.9: *The list of imported packages*
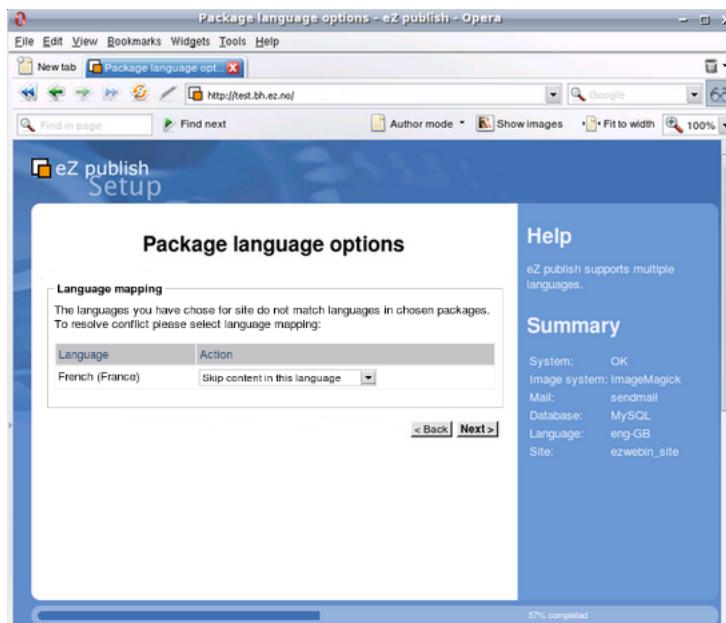
Figure 1.10: *Package language options*

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:

- 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.)

2. 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 (page 339) the packages and installing (page 341) them.

Access method

(see figure 1.11)
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
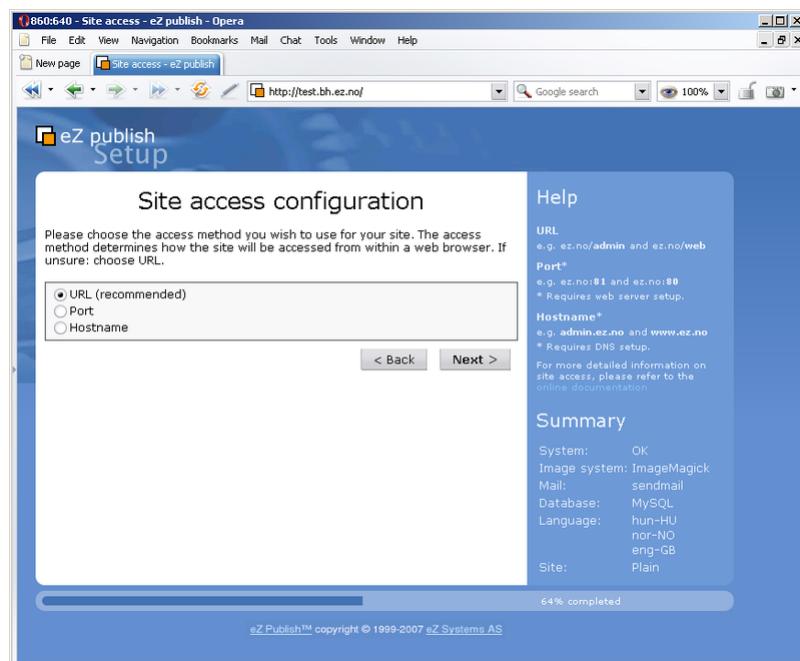
Figure 1.11: *Step 8: Site access configuration*

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 hostname 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.

Hostname

When this access method is used, each site is assigned a unique hostname. 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)
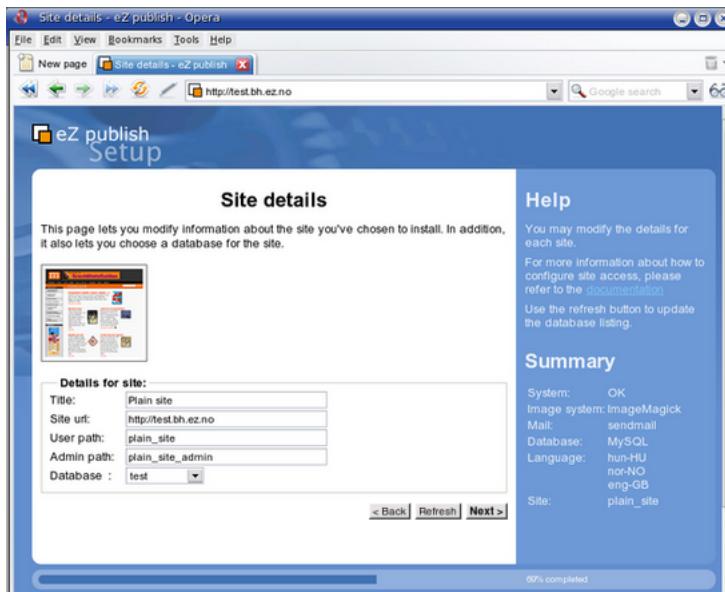
Figure 1.12: *Step 9: Site details*

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 hostname 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

- I've chosen a new database

Use the last option if another database has been chosen.

Site security

(see figure 1.13)
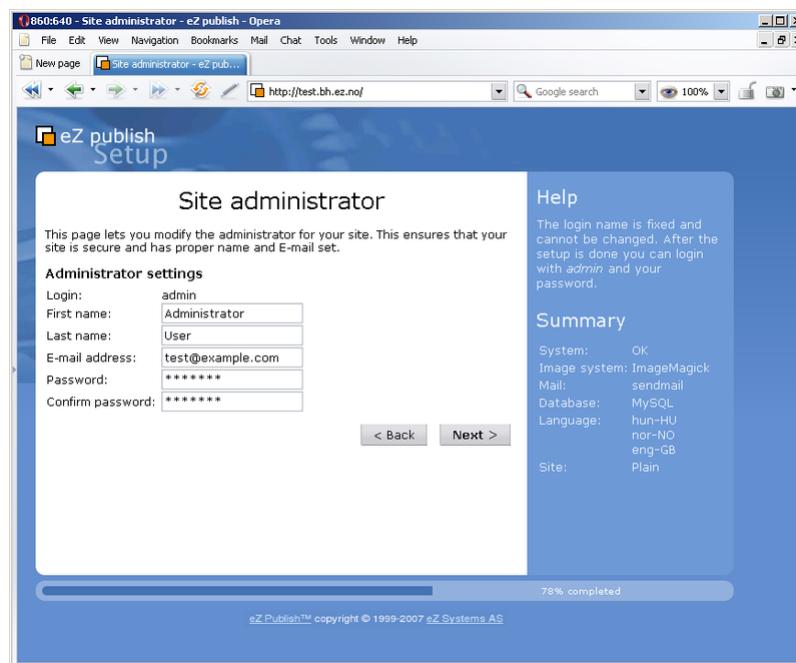
Figure 1.13: *Step 10: Site administrator*

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 username (login) is set to "admin" by default and can not be changed. If you need another username 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)

- The test results

- The type of database that is being used
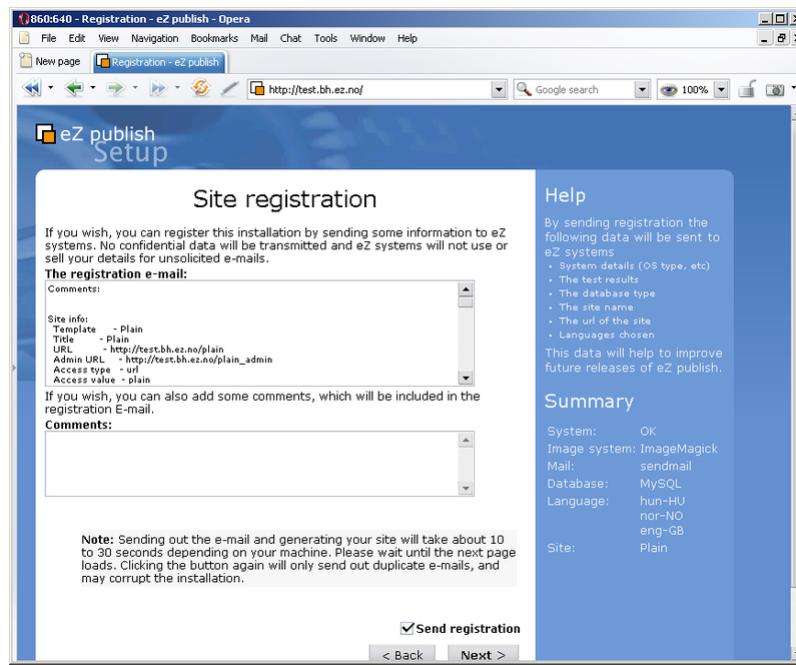
Figure 1.14: *Step 11: Site registration*

- The name of the site

- The URL of the site

- The languages that were chosen

Finished

(see figure 1.15)

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.
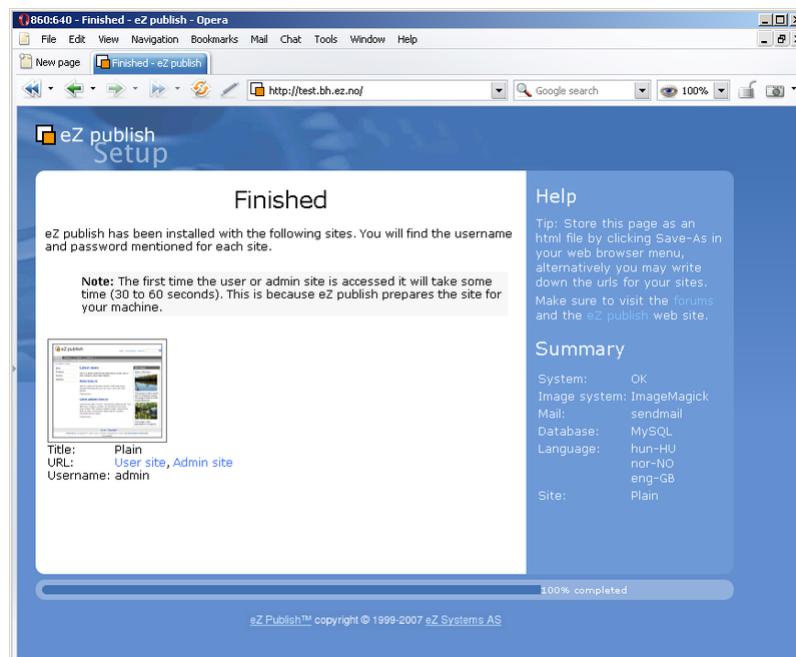
Figure 1.15: *Step 12: Finished*

1.5 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 content/treemenu/? /index_treemenu.php [L]
        RewriteRule ^/var/storage/.* - [L]
        RewriteRule ^/var/[^/]+/storage/.* - [L]
        RewriteRule ^/var/cache/texttoimage/.* - [L]
        RewriteRule ^/var/([^/]+/)?cache/(texttoimage|public)/.* - [L]
        RewriteRule ^/design/[^/]+/(stylesheets|images|javascript)/.* - [L]
        RewriteRule ^/share/icons/.* - [L]
        RewriteRule ^/extension/[^/]+/design/[^/]+/
(stylesheets|flash|images|javascripts?)/.* - [L]
```

```
        Rewriterule ^/packages/styles/.+/(stylesheets|images|javascript)/[^/]+/
.* - [L]
        RewriteRule ^/packages/styles/.+/thumbnail/.* - [L]
        RewriteRule ^/favicon\.ico - [L]
        RewriteRule ^/robots\.txt - [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 "/var/www/ezpublish-3.6.0".
[SERVER_NAME]	The host or the IP address that Apache should look for. If a match is found, the virtual host settings will be used.
[SERVER_ALIAS]	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
```

1

1.5.1   eZ JS Core rewrite rules

These re-write rules are described under the description of the eZ JS Core extension, but since this
extension is bundled with eZ Publish and as such an important part of the installation process,
please find the eZ JS Core rewrite rules described here as well.

Rewrite rules for lib folder

The eZ JSCore extension requires correct re-write rules in order to load the necessary yui (ya-
hoo user interface) files from the file system. This is required only when the INI setting "*ezjs-
core.ini[eZJSCore]LoadFromCDN*" is disabled.
Add the following line to your Virtual-Host setup:

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

Rewrite rules for script/css packing

eZ JsCore packs javascript/css files together to decrease server requests and file size for faster
download. On eZ Publish 4.0.x you'll need to make the following changes if you havn't already
when installing ezoe (Online Editor 5.0).

- .htaccess

       from:

```
RewriteRule !\.(gif|jpe?g|png|css|js|html)|var(.+)storage.pdf(.+)\.pdf$
index.php
```

     to:

```
RewriteRule !(\.(gif|jpe?g|png|css|js|html?)|var(.+)storage.pdf(.+)\.pdf)$
index.php
```

- Virtual Host mode

       add:

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

   If you don't manage to get these working, you can disable this feature by setting
ezjscore.ini[eZJSCore]Packer=disabled

(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

1.5.2   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" 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 content/treemenu/? /index_treemenu.php [L]
        RewriteRule ^/var/storage/.* - [L]
        RewriteRule ^/var/[^/]+/storage/.* - [L]
        RewriteRule ^/var/cache/texttoimage/.* - [L]
        RewriteRule ^/var/[^/]+/cache/(texttoimage|public)/.* - [L]
        RewriteRule ^/design/[^/]+/(stylesheets|images|javascript)/.* - [L]
        RewriteRule ^/share/icons/.* - [L]
        RewriteRule ^/extension/[^/]+/design/[^/]+/
(stylesheets|flash|images|javascripts?)/.* - [L]
        RewriteRule ^/packages/styles/.+/(stylesheets|images|javascript)/[^/]+/
.* - [L]
        RewriteRule ^/packages/styles/.+/thumbnail/.* - [L]
        RewriteRule ^/favicon\.ico - [L]
        RewriteRule ^/robots\.txt - [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>
```

1.6   Removing eZ Publish

1

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.

1.7   Extensions

Extensions are plugins 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 examplified by an imaginery extension called "ezfoo".

1.7.1   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	
	`tar -zxvf ezfoo-extension-1.0.tar.gz`
	or
	`tar -zxvf ezfoo-extension-1.0.tgz`
bz2	

	`tar -jxvf ezfoo-extension-1.0.bz2`
zip	
	`unzip ezfoo-extension-1.0.zip`

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".

1.7.2   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 siteaccesses.

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 checkboxes. To activate the sample extension, select "ezfoo" as shown in the screenshot below and click the "Apply changes" button.

(see figure 1.16)
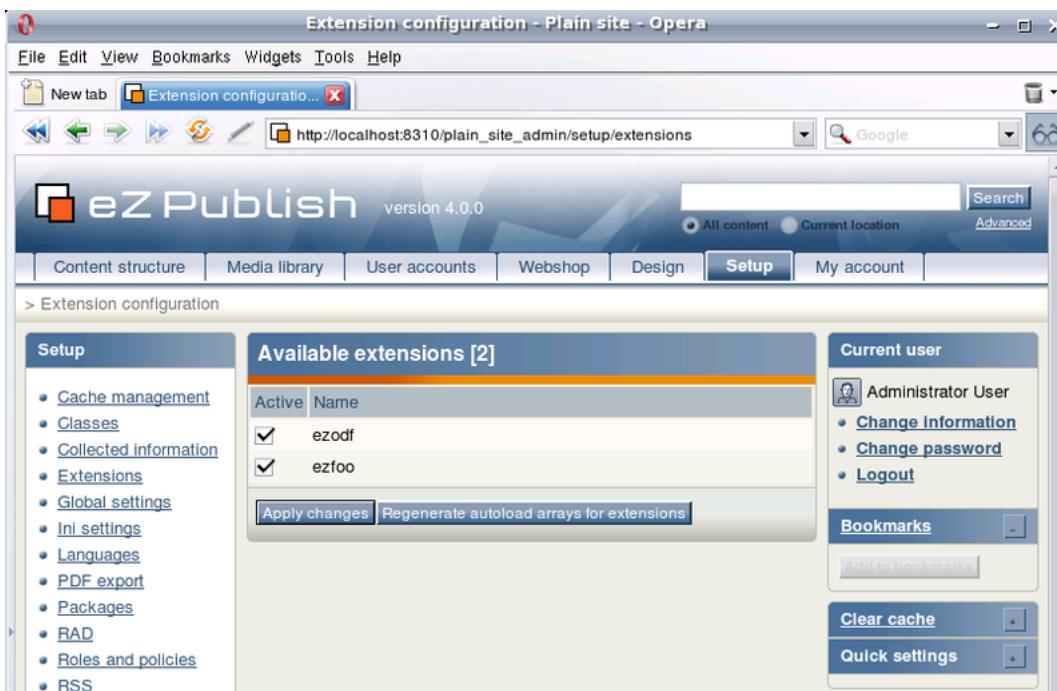
Figure 1.16: *Screenshot of the extension configuration in the administration interface.*

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 (page 2323) configuration file.

Activating for the whole installation

To enable the sample extension for all of your siteaccesses, 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):

```
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 a certain siteaccesses

To enable the sample extension for only a single siteaccess 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.

1.8 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 30) 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 (page 2491)" 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.
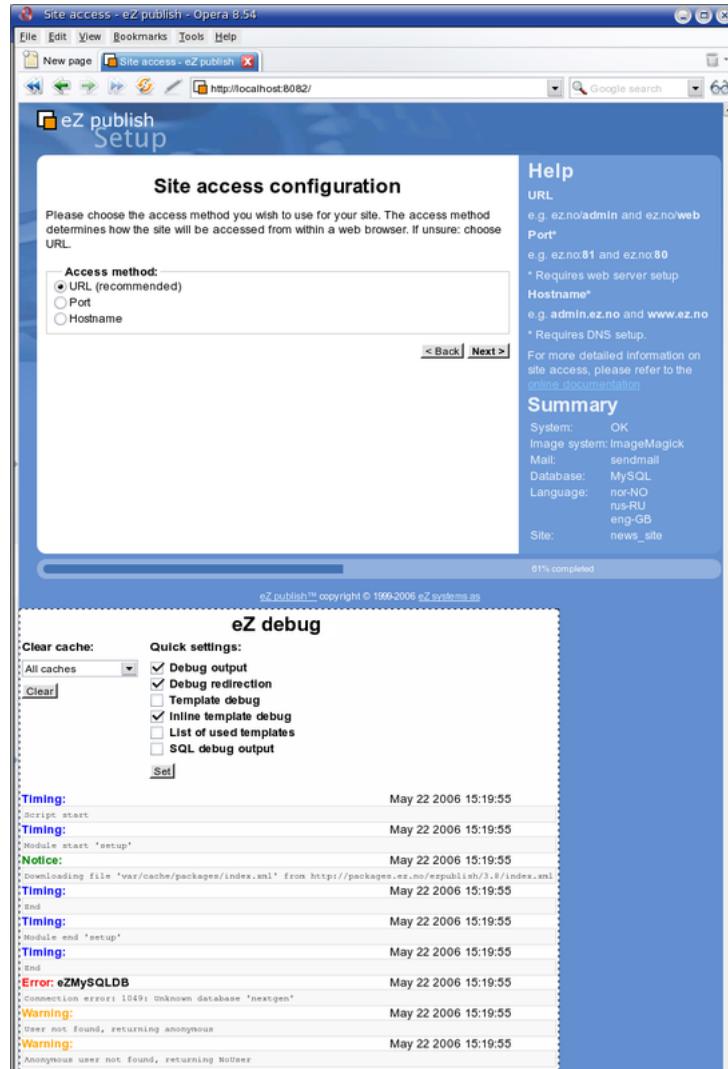
Figure 1.17: *The debug output appears at the bottom of the page*

Chapter 2

Concepts and basics

This part of the 4.x documentation is for eZ Publish 4.0, only *reference section (page 512)* is common for all eZ Publish 4.x versions as well as eZ Publish 5.x "LegacyStack", please select the version you are using for the most up to date documentation!

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 workflow system

- How the access/permission system works

- How the webshop 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)
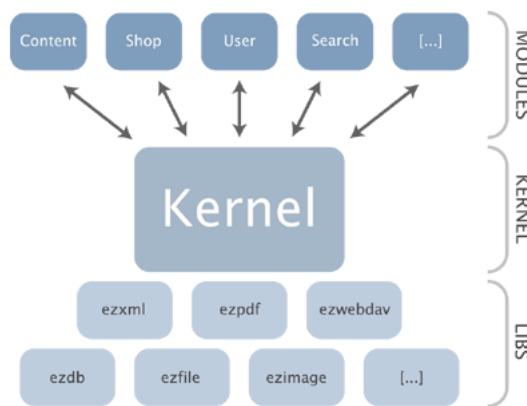
Figure 2.1: *Libraries, kernel and modules.*

The libraries

The libraries are the main building blocks of the system. These are reuseable 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 (page 2808).

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, workflows, 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 (page 645). A module can be broken down into the following components:

- Views

- Fetch functions

A view provides an actual web interface. For example, the "search" (page 810) view of the "content" (page 679) 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" (page 977) fetch function of the "user" (page 974) 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

2.2   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
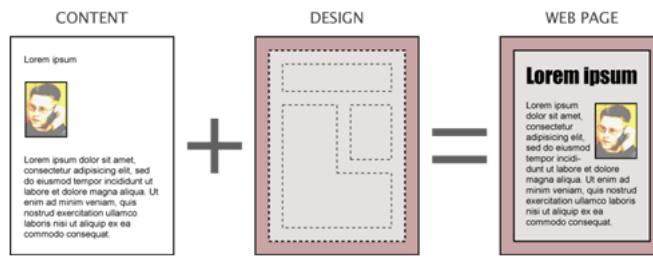
Figure 2.2: *Content + Design = Web page*

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)
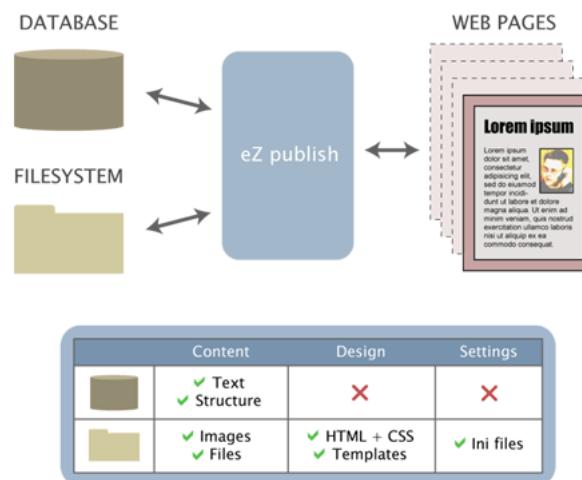
Figure 2.3: *Storage overview*

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 runtime. 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 (page 593)	Stores a single line of unformatted text
Text block (page 591)	Stores multiple lines of unformatted text
XML block (page 601)	Validates and stores multiple lines of formatted text
Integer (page 541)	Validates and stores a numerical integer value
Float (page 532)	Validates and stores a numerical floating point value

Please refer to the "Datatypes" (page 514) 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)

Figure 2.4: *Example of a content class.*

An eZ Publish distribution comes with a set of general purpose classes (page 618) 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 in action.

(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

Figure 2.5: *The class edit interface.*

- 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 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" and "sort order" 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 checkbox labelled "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 it's 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" (page 514) 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)
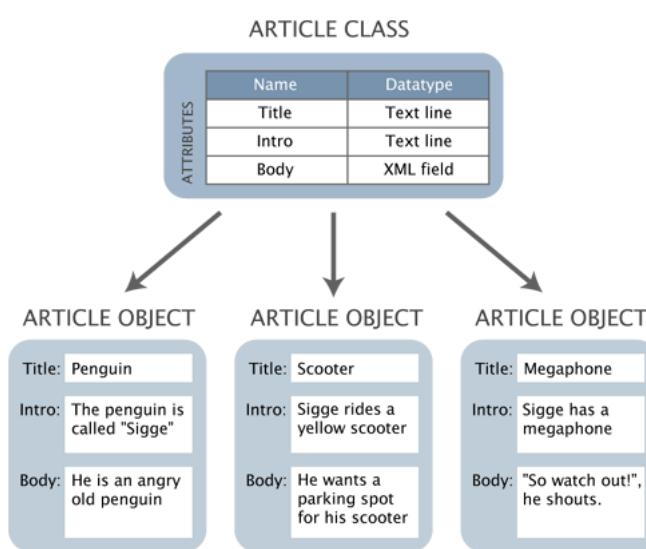
Figure 2.6: *Datatypes, attributes, a content class and objects.*

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.

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

2

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 132).

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

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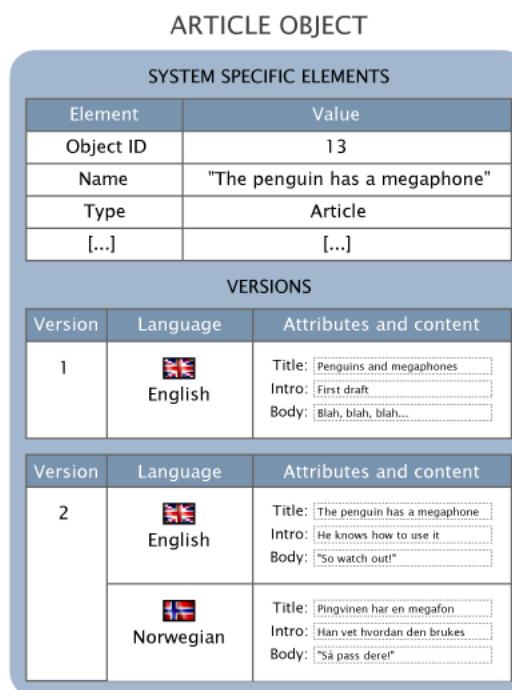
Figure 2.7: *Content object structure (with versions and translations)*.

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)

Figure 2.8: *The list of existing languages for translation of content*

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 "Topphastighet" as attribute names. Refer to the "Translatable class attributes (page 256)" 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 (page 241) part of the features section for further details.

2

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 three 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)
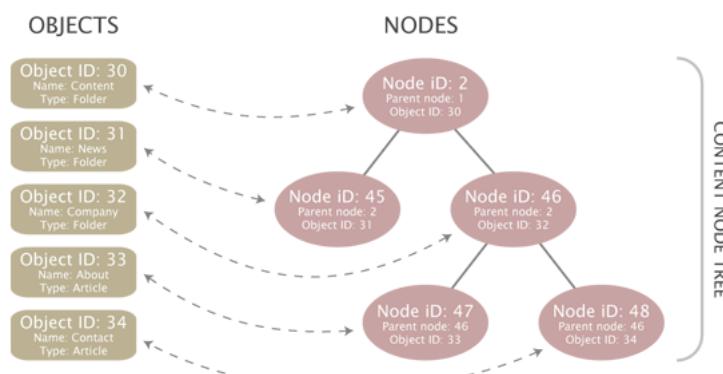
Figure 2.10: *Objects, nodes and the content node tree*

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)

CONTENT NODE TREE
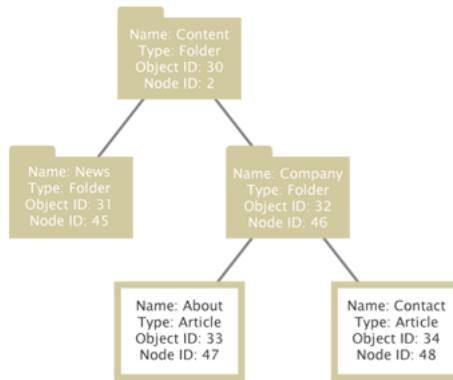
Figure 2.11: *Content node tree*
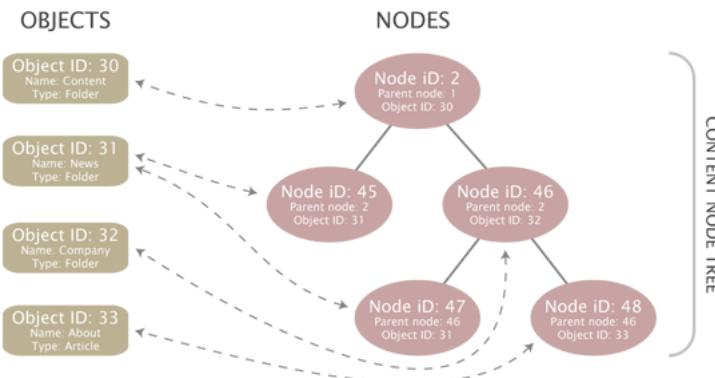
Figure 2.12: *Objects, node and the content node tree - multiple locations*

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.
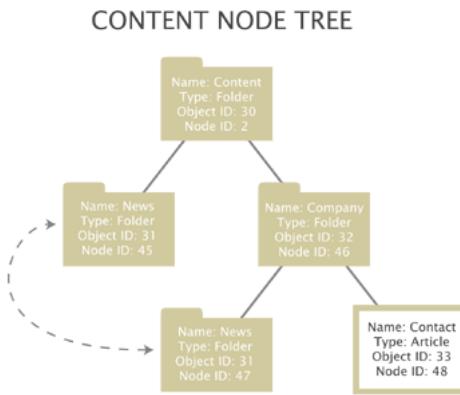
CONTENT NODE TREE

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)
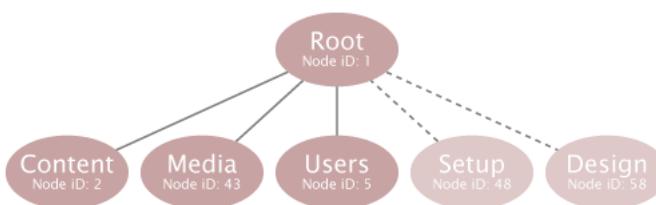
Figure 2.14: *Top level nodes*

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" (page 599) 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, unpublishing 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)
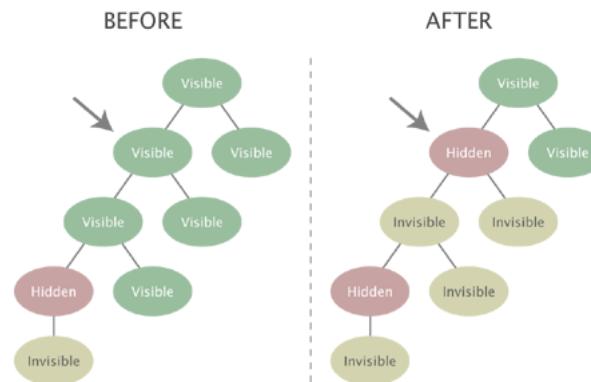
Figure 2.15: *Hiding a visible node*

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)
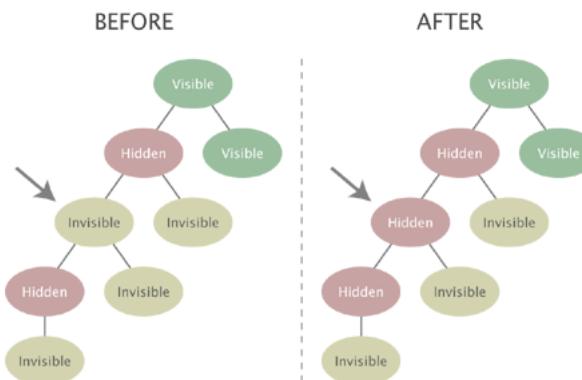
Figure 2.16: *Hiding an invisible node*

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 be remain invisible).

(see figure 2.17)
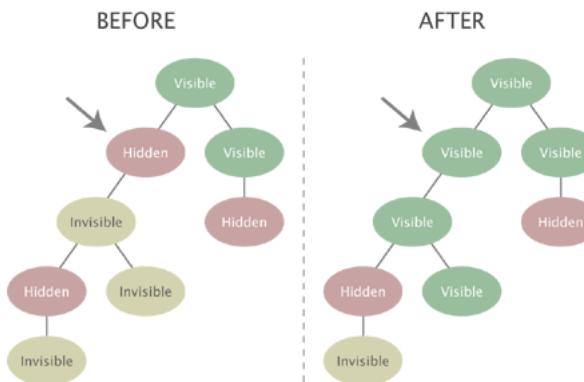
Figure 2.17: *Unhiding a node with a visible ancestor*

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-hided 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-hided.
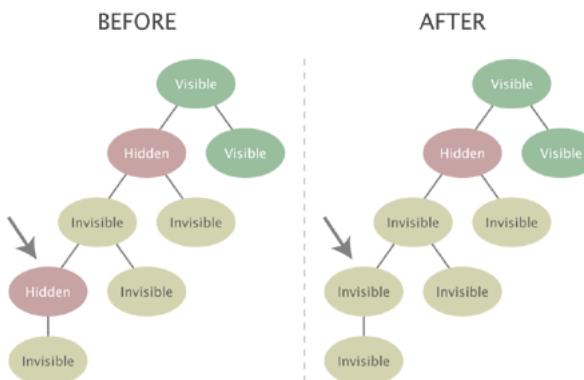
(see figure 2.18)

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" (page 601) 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 (page 601)" 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 (page 601)" 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" (page 572) or the "Object relations" (page 574) 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

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
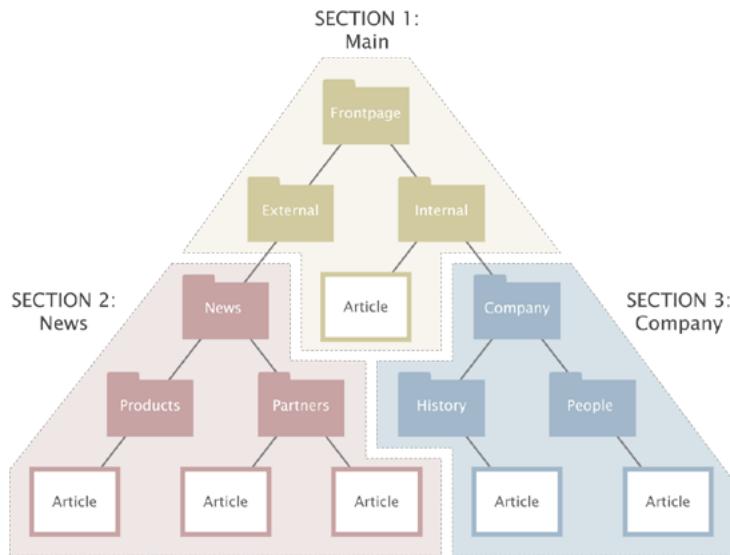
Figure 2.19: *Example of sections.*

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" (page 601) or the "URL" (page 597) 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 (page 518)	Checkbox.	No.
E-Mail (page 526)	Single line of text.	Yes.
Option (page 579)	Radio buttons or a drop-down menu.	No.
Text block (page 591)	Multiple lines of unformatted text.	No.
Text line (page 593)	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" (page 785) view of the "content" (page 679) 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 files should only be used for reference. In other words, they should never be modified. The "Configuration files" (page 1514) 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)
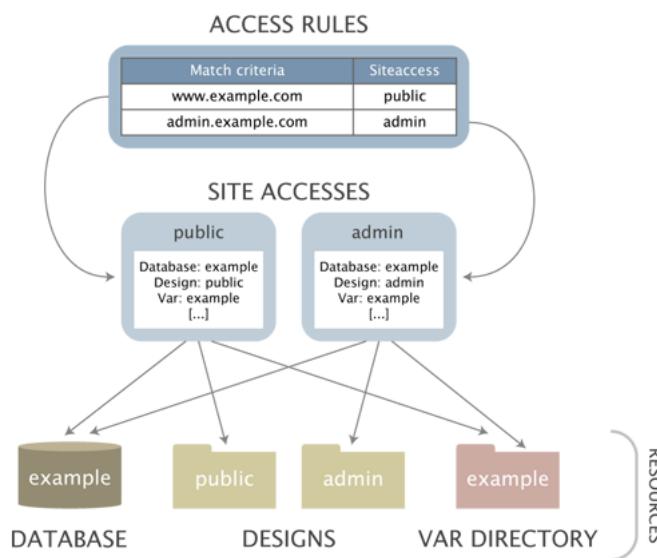
Figure 2.20: *Example of a setup with two siteaccesses.*

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 sub-directory is the actual name of the siteaccess.

Rules for siteaccess names

eZ Publish has a set of characters that are allowed for a siteaccess name. They are the alphanu-merical characters (the **[a-z], [A-Z]** and **[0-9]** ranges), and **the "_" (underscore)**.

These are the simple rules for characters converting:

- Characters not in the previous list (alphanumerical and underscore) are replaced by an "_" **(underscore)**;

- Multiple consecutive "" (double quotes) are replaced by a single quote;

- Leading and trailing "_" are removed.

*Examples: You can have an **http://example.org/** and an **http://admin.example.org/**sites, where the hostname is used for matching the SA. In this case, **example.org**will be changed to **example_org** and **admin.example.org** to **admin_example_org**. Another example is an URL with a space: **http://example.com/main area/**which would lead to SA **"main_area"**.*

The following illustration shows a setup with two siteaccesses: admin and public.

(see figure 2.21)
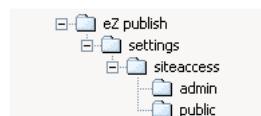
Figure 2.21: *Siteaccess directory example.*

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 auto-matically 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)
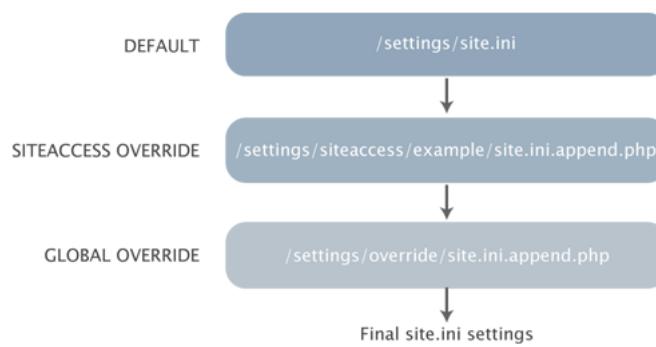
Figure 2.22: *Configuration override example.*

Rules for Online Editor and extensions

This applies for Online Editor and other extensions:
if a configuration array is initialized in "extension/<extension_name>/settings/<ini_file>" and you want to insert a new value into this array:
1. Do not edit "extension/<extension_name>/settings/<ini_file>", because it will be overwritten next time when you upgrade the extension.
2. Do not create "extension/<extension_name>/settings/siteaccess/<siteaccess_name>/<ini_file>", because it will be overridden by "extension/<extension_name>/settings/<ini_file>".
3. Do not create or edit "settings/siteaccess/<siteaccess_name>/<ini_file>", because it will be overriden by "extension/<extension_name>/settings/<ini_file>".
4. To affect all siteaccesses where <extension_name> is used, specify new value in "settings/override/<ini_file>".
5. To only affect a certain siteaccess, you need to use the following workaround:
- create a new custom extension specially for these settings: "extension/<custom_extension>/"
- specify new value in "extension/<custom_extension>/settings/<ini_file>"
- enable the custom extension for the target siteaccess by adding

```
ActiveAccessExtensions[]=<custom_extension>
```

to "settings/siteaccess/<siteaccess_name>/site.ini.append.php"
- make sure the configuration array is not re-initialized in "settings/override/<ini_file>" (if exists).

Note: The name of <ini_file> here should always contain either ".append" or ".append.php" in the end. The only exception are default configuration files (for example "settings/design.ini" or "extension/ezoe/settings/ezoe.ini").

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
<?php /*

[article_full_ezno]
Source=node/view/full.tpl
MatchFile=article/full.tpl
Match[class_identifier]=article
Subdir=templates

*/ ?>
```

2

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" (page 2503) 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 (page 2527)" 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" 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 "Host-MatchType" (page 2501) 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.

2

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 (page 679)	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 (page 974)	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 (page 881)	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" (page 645) 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" 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 (between parenthesis) 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.

Another example: if the view "V" of the module "M" has the ordered parameters: A and B, and the unordered parameters C and D, the following possible URLs can be written: *(/<module>/<view>/[list-of-ordered-parameters/][(<unordered-parameter-name)/ <unordered-parameter-value>])*:

/M/V
/M/V/foo (here parameter A = foo)
/M/V/foo/bar (here parameter A = foo, B = bar)
/M/V/foo/(C)/bar (here parameter A = foo, C = bar)
/M/V/foo/(D)/bar (here parameter A = foo, D = bar)
/M/V/foo/(D)/bar/(C)/baz (here parameter A = foo, C = baz, D = bar)

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
```

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 146) 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 143) 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 276) 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 (page 271) 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)
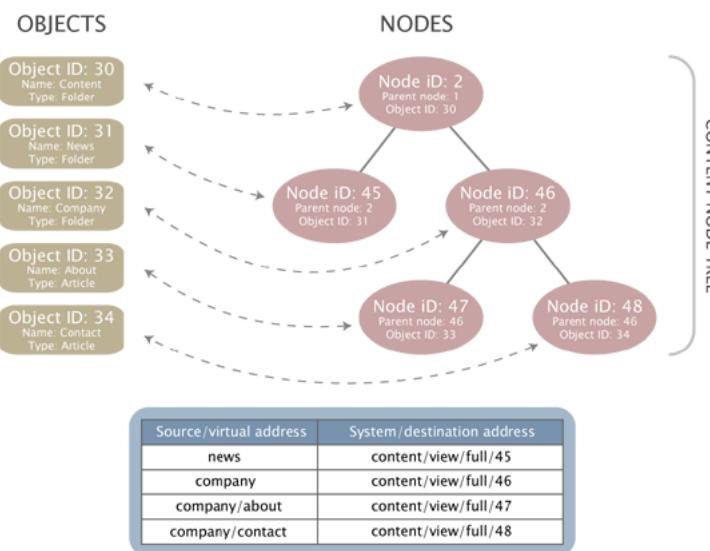
Figure 2.23: *Objects, nodes and nice URLs.*

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.

In eZ Publish 3.9 and earlier versions, URL transformation rules were more restrictive and only
supported some ASCII characters (lowercase Latin letters from "a" to "z", digits and underscores).
Special symbols were converted to underscores and special characters were converted using the
built in transliteration feature. For example, the Norwegian characters "", "" and "¥" were con-
verted to "ae", "oe" and "aa". If the system was about to generate a virtual URL that already
existed, it would simply append an underscore at the end of the newly generated address and
thus the risk of having duplicate URLs was eliminated.

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 276)" for more information. In addition, wildcard based URL forwarding 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

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 subdirectory 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" (page 1196) 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 226) 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)
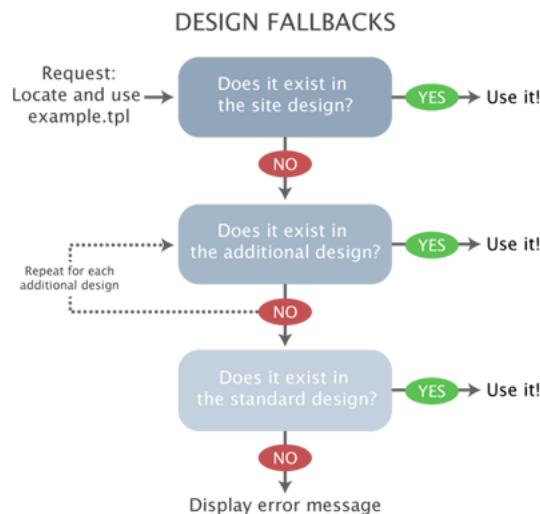
Figure 2.24: *The design fallback mechanism.*

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)
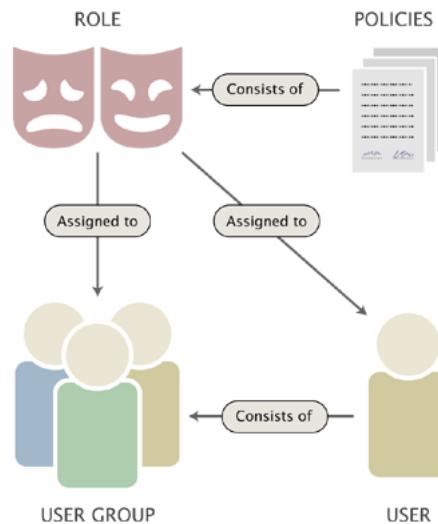
Figure 2.25: *Users, groups, policies and roles.*

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" (page 643) 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" (page 599) 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 (page 2577) is exceeded. A failed login attempt is a combination of a valid username 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 login attempts are stored in a database table called "ezuservisit". An account's failed login counter is automatically reset upon a successful login. In other words, as long as you log in with a valid username/password combination, the failed login 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 nonexisting 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.
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 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 (page 2165) to see how you enable the eZ Webshop functionality in eZ Publish.

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 2.26)

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.

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

Figure 2.26: *The integrated e-commerce solution.*

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 (page 375)" 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 (page 417)" 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 (page 582)	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 (page 570)	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 (page 579)	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 (page 561)	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 (page 563)	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 (page 586)	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).

2.10   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 2.27)
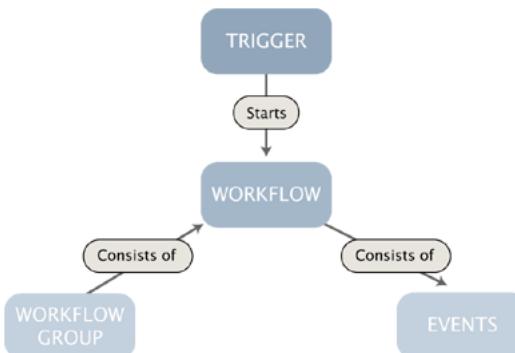
Figure 2.27: *The workflow system.*

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" (page 1127) 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

Chapter 3

Templates

This part of the 4.x documentation is for eZ Publish 4.0, only *reference section (page 512)* is common for all eZ Publish 4.x versions as well as eZ Publish 5.x "LegacyStack", please select the version you are using for the most up to date documentation!

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" (page 1162) and the "l10n" (page 1184) 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 pagelayout 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
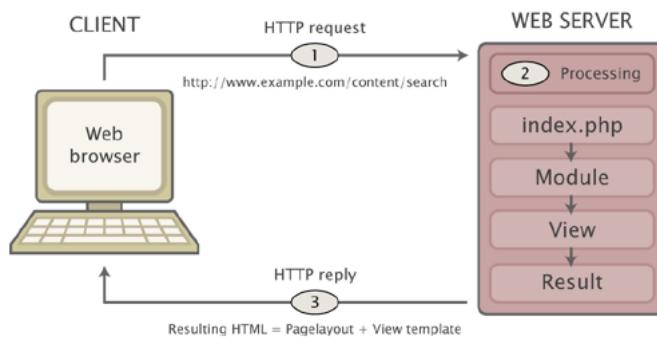
Figure 3.1: *Client - server cycle.*

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 pagelayout. The following illustration shows how the module/view result (generated by different modules/views - depending on the request) is included in the pagelayout:

(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 173) 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 175) 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 226) section for a detailed description of the template override mechanism.

3

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" (page 821) view of the "content" (page 679) 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 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)
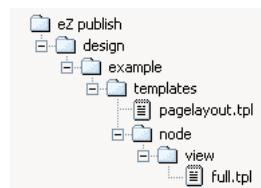
Figure 3.3: *Location of pagelayout and full view template in example design.*

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 be placed in the "override/templates" directory of the main design used by the siteaccess. Please

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

refer to the documentation of the template override system (page 226) 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 (page 1051) 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 223) 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 is 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

3.2   The pagelayout

The pagelayout is the main template. Among other things, it dictates the overall look of a site. The filename of the pagelayout 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 pagelayout within the current design (specified by the siteaccess), it will attempt to use the pagelayout template that is provided by one of the fallback designs. The following illustration shows the location of the pagelayout template located in a design called "example".

(see figure 3.5)

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

The pagelayout 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 pagelayout:

```
<!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 pagelayout 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 pagelayout.

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" (page 1338). 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 143) 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 181) 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 pagelayout 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 pagelayout 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 185) 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 pagelayout 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 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-

3

	icon.ico" in the images folder of a site design.
Copyright	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 pagelayout 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 143) 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 (page 1117) 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	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. The standard page head (page 181) 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).
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 "ezcontentnavigationpart"). 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

		URL of the node (for example "/content/view/full/44"). **Key:** url_alias **Description:** The virtual URL of the node (for example "company/about_us"). **Key:** node_id **Description:** The ID number of the node. The node being viewed will have its "url" and "url_alias" components set to false. In addition, the "node_id" 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).
title_path	array	Almost the same as the "path" array (see above). When a node is being viewed, the standard page head (page 181) template uses the "title_path" array to build the TITLE component of the HEAD section.
section_id	string	The ID number of the section which the object referenced by the node being viewed belongs to.
node_id	string	The ID number of the node that is being viewed.
navigation_part	string	Contains the name identifier of the current navigation part (for example "ezcontentnavigationpart"). This variable is used by the administration interface to determine which part the user interacts with.
content_info	array	Contains miscellaneous information about the node

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.

3

		Variable: url_alias **Type:** string **Description:** The virtual URL of the node (for example "company/about_us").
		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.
		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".
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" (page 1391)
and "rdelim" (page 1393) (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

3.3.1   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 (page 1298) 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" (page 1228) or the "false" (page 1210) 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" (page 1138) operator. Associative arrays can be created with the hash (page 1150) operator. The following examples demonstrate the creation of arrays and hashes.

3

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" (page 1012) 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" (page 1062)) 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

3.3.3   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" (page 1403) function. Some templates have preset variables, for example, the main template (pagelayout) provides access to a collection of variables (page 185).

Creating and destroying variables

All variables used in a template must be declared and defined by the "def" (page 1398) function (short for define) before they can be used. A variable exists until the "undef" (page 1407) 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" (page 1398) and "undef" (page 1407) functions.

Changing the contents of variables

The contents/value of a variable can be changed at any time using the "set" (page 1403) 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" (page 1403) function.

Accessing array elements

The elements of a simple/vector array can only be accessed using numerical indexes. This method is called "index lookup". 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 syntaxes (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" (page 1258) 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.

Important note: he *dumb (page 1267)* template operator is an alias of *attribute()* available in 4.7 and higher. It is simpler to use and makes it possible to inspect the contents of arrays, hashes and objects.

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" (page 1051) 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" (page 1032) 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_	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'
match_ingroup_id_list	boolean	false

Again, this object consists of a lot of information. As mentioned above, the "attribute" (page 1258) 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 (page 1430) 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 (page 1432) 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 (page 1441) 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:

3

```
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 (page 1436) 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 (page 1437) 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 (page 1438) 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 (page 1374) and operators (page 1135) 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)
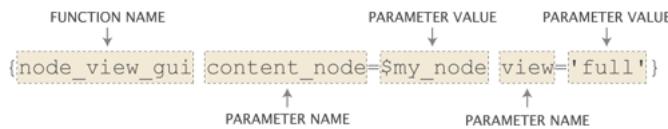
Figure 3.7: *Typical components of a function call.*

The example above calls the "node_view_gui" (page 1424) 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.

(see figure 3.8)
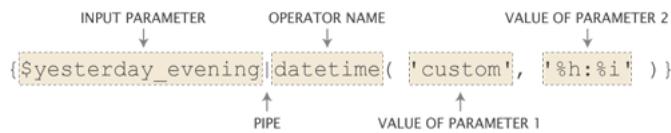
Figure 3.8: *Typical components of a template operator call.*

The example above demonstrates the usage of the "datetime" (page 1179) 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" (page 1390) 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 for more information about this feature.

Output washing

Variables that may contain bogus strings should always be washed using the "wash" (page 1331) 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" (page 1298) 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, ' sandwitch' )}
```

The following output will be produced:

```
Liver sausage sandwitch
```

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) 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

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” (page 1342) operator makes sure that a URL works regardless of the location of the eZ Publish folder, the access method (page 143) and the environment that eZ Publish is running in (non virtual host, virtual host, 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 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 and uses the host access method, 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 150) section of the "Concepts and basics" chapter.

ezimage

The "ezimage" (page 1339) 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 for more information about this feature.

ezdesign

The "ezdesign" (page 1338) 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 for more information about this feature.

3

3.5   Information extraction

Information that is stored by eZ Publish can be extracted using the "fetch" (page 1174) 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"
(page 1472) 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.

3

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" (page 1051) 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" (page 1051) objects. The list fetch function of the content module can take several parameters. These parameters are optional and can be used to finetune 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" (page 714) 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" (page 1051) 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" (page 1331) 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" (page 1184) 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" (page 1184) 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" (page 1042) 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" (page 1413) 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".

```
<img src="{$node.object.data_map.my_image.content[image_size].full_path}" ... />
```

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" (page 1413) 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

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" (page 1390) 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" (page 1444) 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)
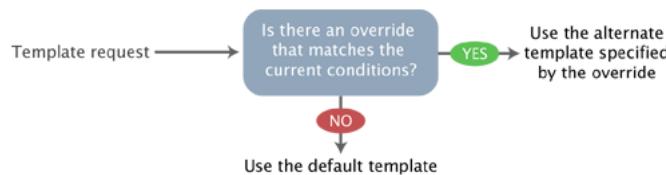
Figure 3.9: *The override system.*

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)
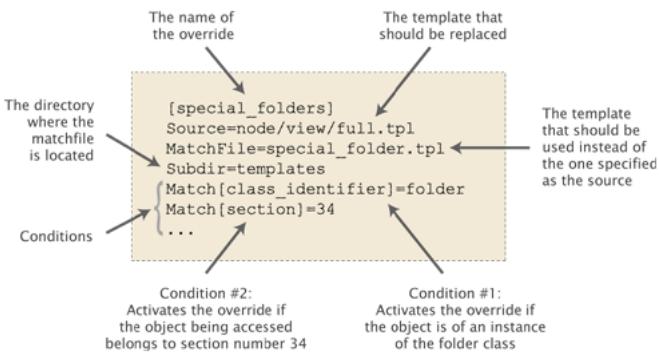
Figure 3.10: *Template override example.*

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 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" 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)
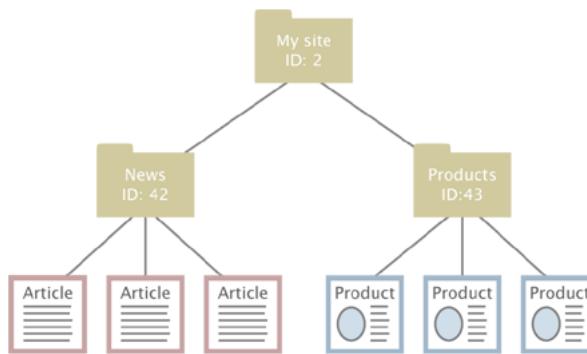
Figure 3.11: *Example content node tree.*

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" subdirectory 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)
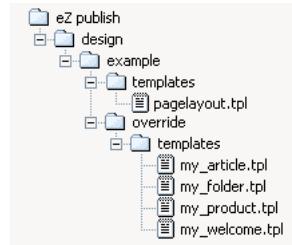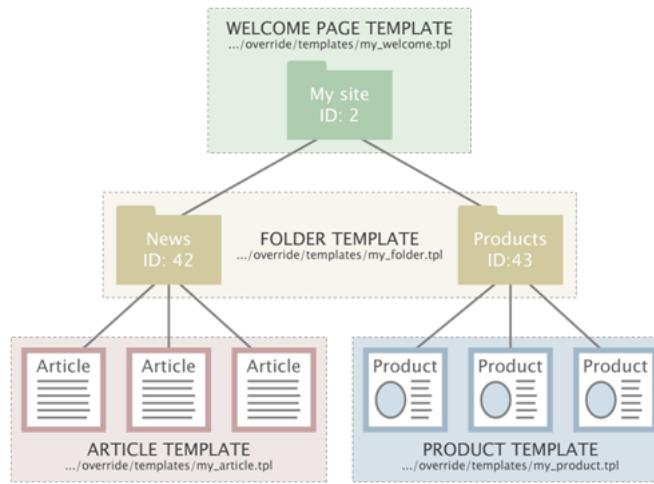
Figure 3.12: *Pagelayout + override templates in example design.*

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.

3

Chapter 4

Features

This part of the 4.x documentation is for eZ Publish 4.0, only *reference section (page 512)* is common for all eZ Publish 4.x versions as well as eZ Publish 5.x "LegacyStack", please select the version you are using for the most up to date documentation!

This chapter contains information about miscellaneous eZ Publish features along with instructions revealing how to configure and use them.

4.1 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 (username 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	- Timestamp - IP address - User (name:ID)	login.log
user-failed-login	Failed login attempts	- Timestamp - IP address	failed_login.log

		• User (name:ID)	
content-move	Location change of content	• Timestamp • IP address • User (name:ID) • Old parent node ID • New parent node ID • Object ID • Object name • Comment	content_move.log
content-delete	Removal of content	• Timestamp • IP address • User (name:ID) • Node ID • Object ID • Object name • Comment	content_delete.log
role-change	Role and policy changes	• Timestamp • IP address • User (name:ID) • Role ID • Role name •	role_change.log

4

		Comment	
role-assign	Role assignment to users and groups	- Timestamp - IP address - User (name:ID) - Role ID - Role name - Object name - Comment	role_assign.log
section-assign	Section assignments	- Timestamp - IP address - User (name:ID) - Section ID - Section name - Node ID - Object ID - Object name - Comment	section_assign.log
order-delete	Removal of webshop orders	- Timestamp - IP address - User (name:ID) - Order ID - Comment	order_delete.log

4

Configuration

By default, the auditing feature is turned off. In order to use audit trailing on your site, enable the "Audit (page 1521)" setting located in the "[AuditSettings]" section of an override for the "audit.ini" configuration file. Using the "AuditFileNames (page 1519)" 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 (page 2382)" setting can be used to specify where the audit log files should be stored. The default directory is "log/audit". This path refers tp VarDir (page 2384) setting, defined in site.ini.

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=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 "<VarDir>/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 (username 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 (page 1521)" setting located in the [AuditSettings] section of

an override for "audit.ini" is enabled and add a new element to the "AuditFileNames (page
1519)" 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

4.2   Policy functions

The built-in access control mechanism (page 158) 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 146). 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 (page 841)" module are specified in "kernel/notification/module.php".

```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 (page 811)" 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 (page 915)" 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 "discount-groupview (page 946)" 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" section of the "Concept and basics" chapter for an overview of the available function limitations.

4

4.3   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 "Topphastighet" as attribute names. Refer to the "Translatable class attributes (page 256)" 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 245)" section for information about setting locale for your site, translating the administration interface, creating custom locales etc.

Default language

From 3.8, the "ContentObjectLocale (page 2431)" 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 261)" 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 254). 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, it's *initial/main language* will be set to the language that was used during creation. For example, if an article is created in Hungarian, it's 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).

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 sitaccess. 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

4.3.1   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 (page 1184)" 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 (page 1182)" 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 (page 2429)" 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. 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 letter 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:

4

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 Germany	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 Italy	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 - Bokm¥l (Norway)
nor-NO@intl.ini Norway	Norwegian (intl)
nor-NO@spraakraad.ini	Norwegian (Standard set by Spr¥kr¥det)
pol-PL.ini	Polish (Poland)
por-BR.ini	Portugese (Brazil)
por-MZ.ini	Portugese (Mozambique)
por-PT.ini	Portugese (Portugal)
por-PT@euro.ini	Portugese (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

4.3.2   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 (page 60) 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 siteaccesses. 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 (page 254) 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 siteaccess. 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 4.1)

In this case, you will have the following settings for locale, default language and site languages after the setup wizard is finished:
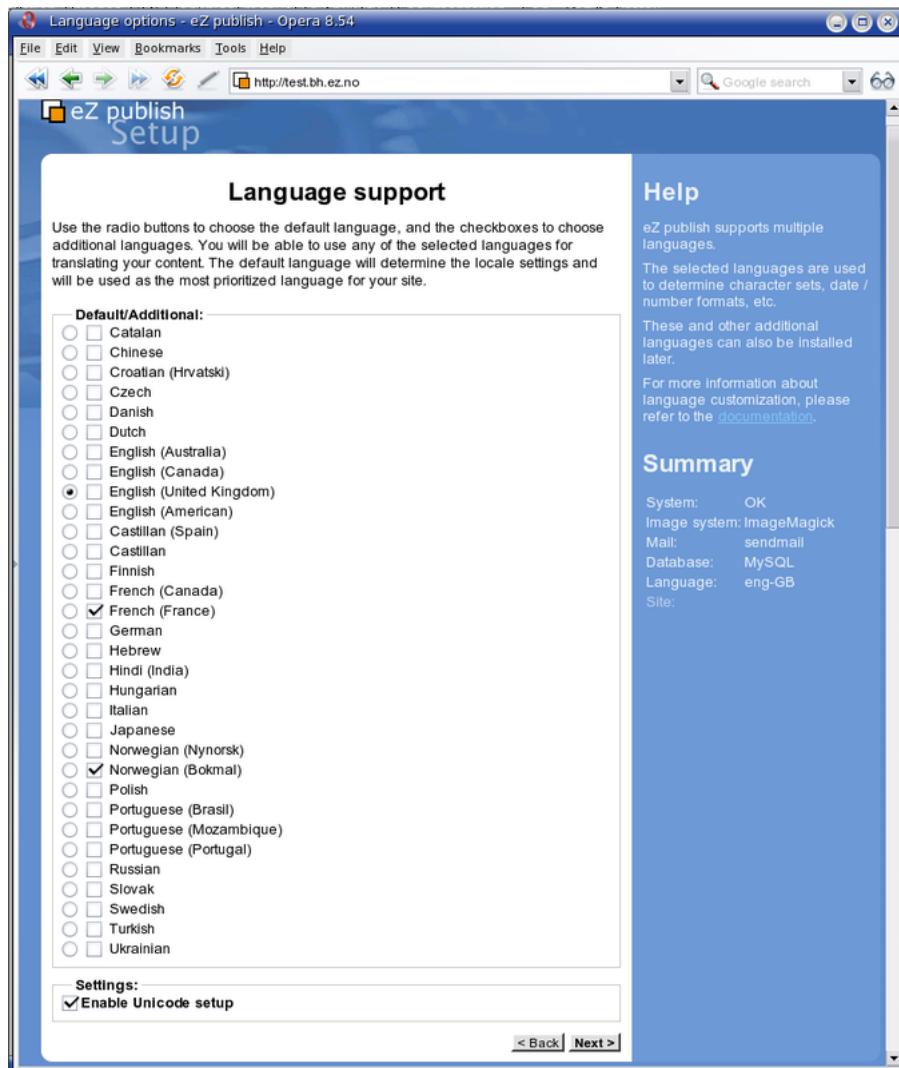
Figure 4.1: *The language selection step in the setup wizard.*

```
[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 (page 2429)" 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. 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 4.2)

Figure 4.2: *The "Create here" interface.*

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"

`[RegionalSettings]`
`SiteLanguageList[]`
`SiteLanguageList[]=eng-GB`
	`[RegionalSettings]`
	`SiteLanguageList[]`
	`SiteLanguageList[]=fre-FR`
	`SiteLanguageList[]=eng-GB`

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

4.3.3   Managing the translation languages

The administration interface allows you to manage the translation languages 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 4.3)

Figure 4.3: *The list of translation languages.*

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 and Norwegian. 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 first.

(see figure 4.4)

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).
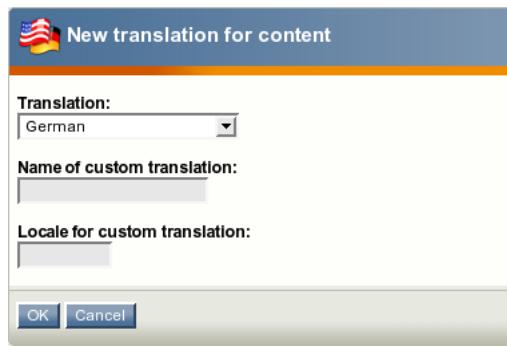
Figure 4.4: *Adding a new translation language.*

To remove one or more languages from the system, select the languages that you wish to remove (use the checkboxes located in the first column) and click the "Remove selected" button.

4.3.4   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, it's 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 4.5)

    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 254)" 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.

Figure 4.5: *Choose a language for a new content class.*

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 4.6)

    The system will display the language selection interface for content classes (see the following screenshot).

   (see figure 4.7)

   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).
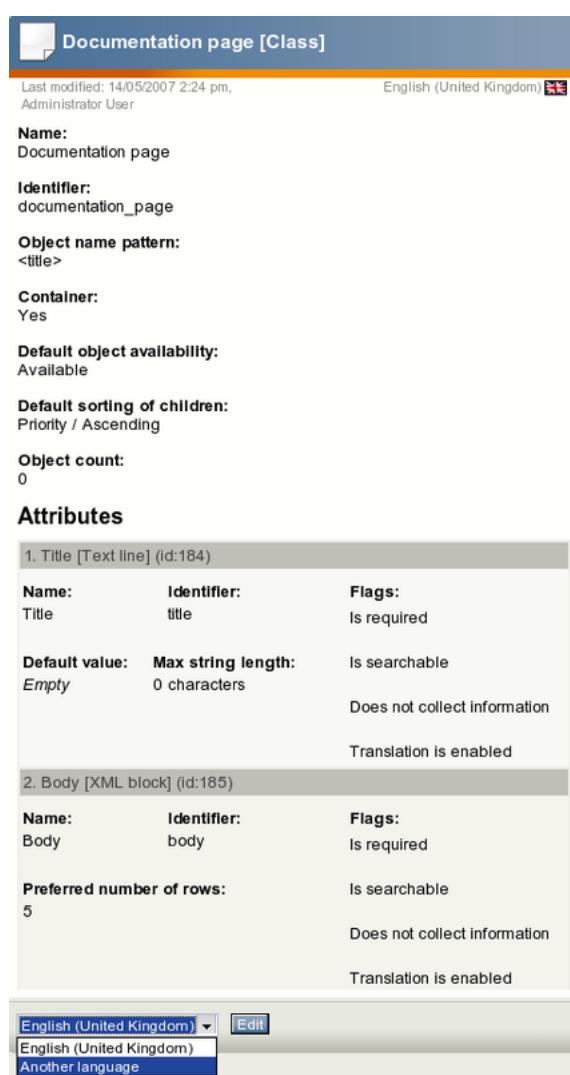
Figure 4.6: *The main window of the class view interface.*
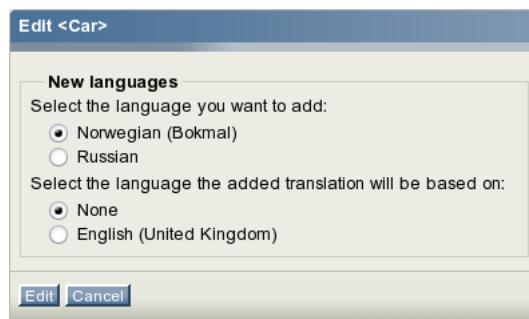
Figure 4.7: *The language selection interface for class attribute names.*

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).
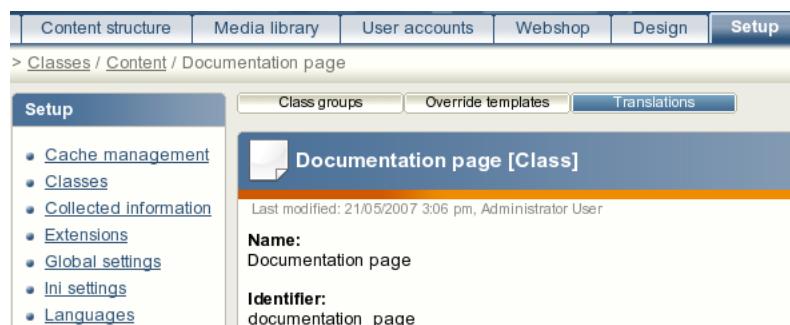
   (see figure 4.8)

Figure 4.8: *Enabling the translations window.*

 A bluish background indicates that the switch is on and thus the "Translations" window will be active/visible. The following screenshot shows how this window looks like for a class that exists in three languages.

Figure 4.9: *Translations window.*

(see figure 4.9)

 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.

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.

4.3.5 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 245)" 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 (page 520)" datatype is being edited.

(see figure 4.10)

Figure 4.10: *List of countries containing translated country names.*

4.3.6 Multilingual 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 4.11)

Figure 4.11: *The "Create here" interface.*

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 (page 254)" 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. 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" checkbox 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 4.12)

Figure 4.12: *The list of classes.*

   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" checkbox as shown in the screenshot below and click the "OK" button to save your changes.
   (see figure 4.13)

Please note that the changes will not affect any of the existing articles. Only new articles will be affected.

Figure 4.13: *The class edit interface.*

4.3.7   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 and French languages.

(see figure 4.14)

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 4.14: *The language selection interface.*

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" submenu and select the language that you wish to edit as shown in the following screenshot.

   (see figure 4.15)
   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 4.16)
   The system will bring up the edit interface.

Figure 4.15: *The context menu.*

Figure 4.16: *Selecting the language for editing.*

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" submenu and select the language that you wish to edit. The system will bring up the edit interface.

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 how the draft list interface looks like when the user is editing three translations of the same article (this interface can be accessed by clicking the "My account" tab and selecting the "My drafts" link on the left).

(see figure 4.17)

Figure 4.17: *The "My drafts" interface.*

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" submenu 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.

Figure 4.18: *The reduced language selection interface.*

(see figure 4.18)
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" submenu 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.

4.3.8   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

4.3.9   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 4.19)
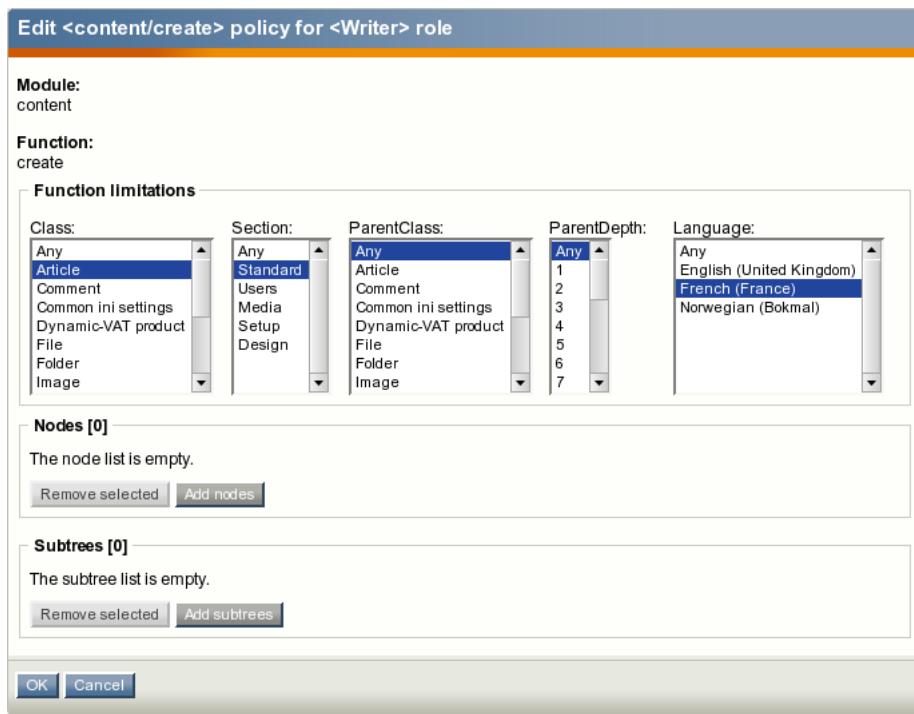
Figure 4.19: *The policy edit interface.*

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 4.20)
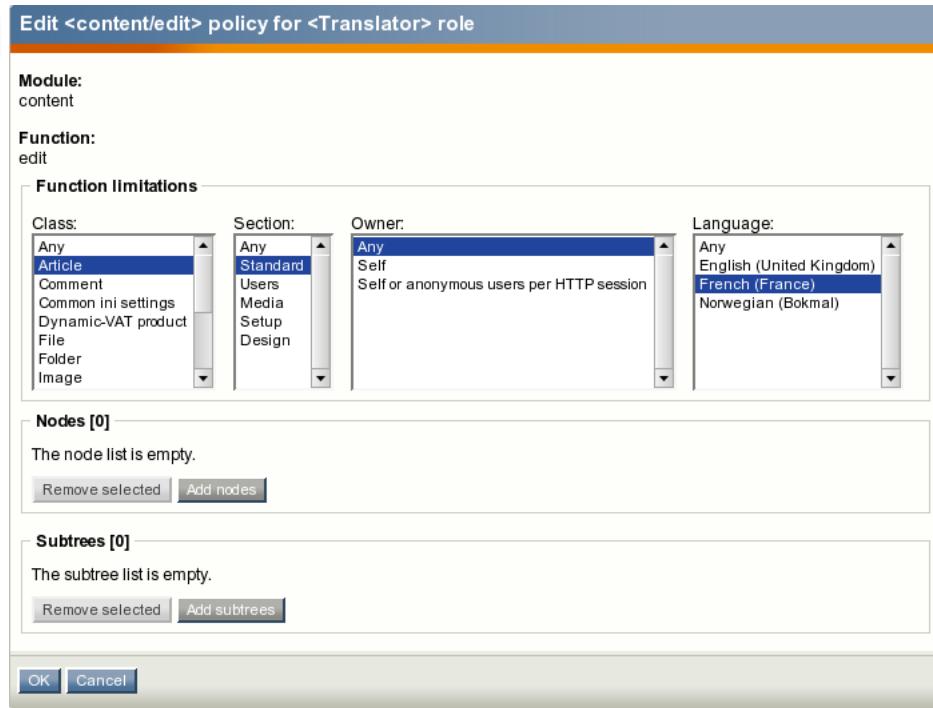
Figure 4.20: *The policy edit interface.*

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 4.21)
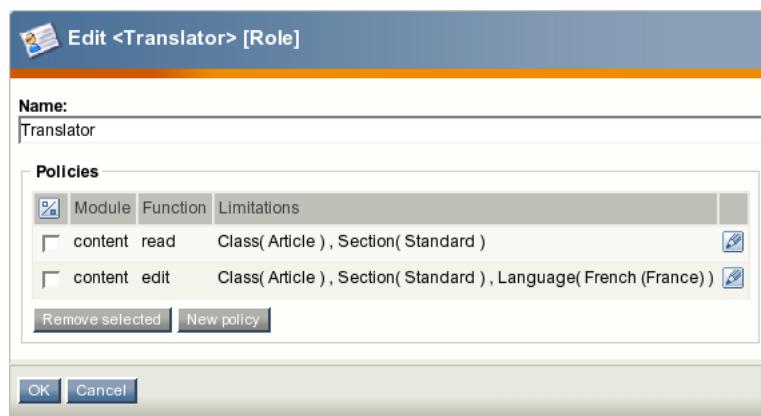
Figure 4.21: *The role edit interface.*

4

4.4   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) has been 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 283):

- 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 4.22)

Figure 4.22:

Use the "Alias should redirect to its destination " checkbox 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]`	`[RegionalSettings]`
`SiteLanguageList[]`	`SiteLanguageList[]`
`SiteLanguageList[]=eng-GB`	`SiteLanguageList[]=fre-FR`
`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 (page 2563)" 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

4.4.1   Managing URL aliases

The administration interface makes it possible to easily manage the virtual URLs 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 4.23)

Figure 4.23: *The interface for managing the URL aliases of a content node*

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 "Company" node. There are three manual aliases: "articles/company_info", "MaCompagnie" and "mycompany". While the "MaCompagnie" alias is associated with French language, the "articles/company_info" and "mycompany" aliases are in English. This means that the same page (node) can be accessed using any of these aliases if both French 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 (page 2421)" 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 "Relative to parent" check-box can be used to specify where the alias/URL to be added should start. If checked, it will start from the parent node. Otherwise, the provided alias is created from the root of the site. 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 "Relative to parent" check-box was checked or unchecked.

The "Generated aliases" window

In the example above, the "Company" node exists in both English and French languages and thus it has two auto-generated aliases "Company" and "Compagnie". 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 towards the bottom of the interface. The "Company" page can be accessed using any of these aliases if both French 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" checkb-ox 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 4.24)

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, two global aliases for the "search" view of the "content" module have been added. While the first alias is associated with the English language, the second one is in French. This makes it possible to access the "search" view using the "findme" or "trouve-moi" aliases if the current siteaccess is configured to use both 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 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.

Figure 4.24: *The interface for global URL aliases managing*

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 siteaccesses 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 4.25)

The interface displays all wildcard aliases in the system and enables you to create new ones. In
the example above, three wildcard aliases have been added. While the first two aliases are in
English, the third one is in German.
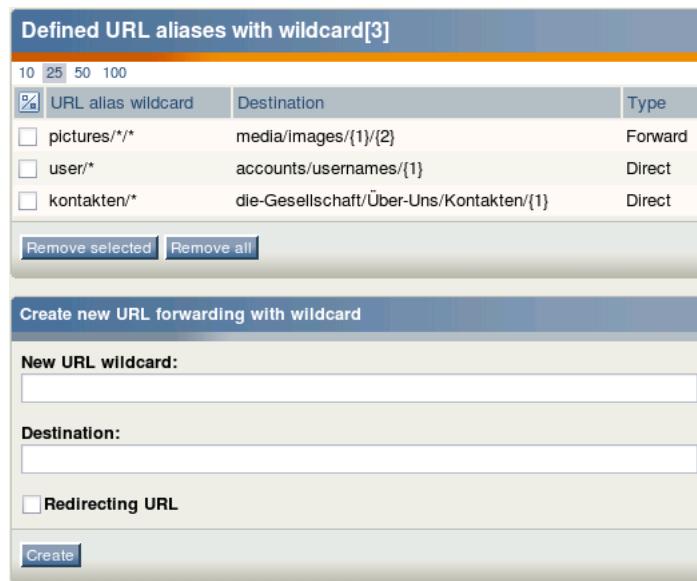
Figure 4.25: *The interface for managing wildcard URL aliases*

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, the first alias is 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" checkbox 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.

4.4.2   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 (page 2562)" 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 exits 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, it's 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
<?php
    class StripWords
    {
        function process( $text, $languageObject, $caller )
        {
            return str_replace( "hell", "", $text );
        }
    }
?>
```

4

The filter class "StripWords" implements a method called "process" which has three parameters: the text to filter, the language object (eZContentLanguage (page 1031)) 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] (page 2560)" section of the "site.ini" for more information about the "Filters" setting.

4

4.4.3 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
<?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

4.5   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

    - Template block cache

- **–** Expiry cache

- **–** URL alias cache

- **–** RSS cache

- **–** User info cache

- **–** Class identifier cache

- **–** Sort key cache

Other files are stored using the file system, including (but not limited to):

- INI files

- Template files

- Compiled templates

- PHP files

- Log files

- Caches that are not related to content:

  - **–** Global INI cache
  - **–** INI cache
  - **–** Codepage cache
  - **–** Character transformation cache
  - **–** Template cache
  - **–** Template override cache

Content view cache

When eZ Publish is displaying a page (a content node), it executes the "view" view of the "content" module and include the output in the page layout. If the output is cached (page 447), the cache file(s) will be read and served. If not, the system will fetch the content stored in the eZ Publish object database (page 111), render the necessary templates, generate a web page and store the resulting XHTML on the file system before serving it. As previously mentioned, these files can be stored in the database and thus the files (along with changes) are easily and immediately available to all servers in the cluster.

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 307) and Setting it up for an eZ DFS FileHandler (page 312).

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 timestamp 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 (page 2836)" 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
...
```

4.5.1 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: The eZ DFS File Handler is not available in eZ Publish 4.1. This documentation applies to eZ Publish 4.2 and above as regards to eZ DFS File Handler.

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 webserver's user on each eZ Publish server. Also it 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 installations 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 312)".

4

4.5.2 Cluster Configuration Settings

Please, refer to the documentation on the left menu about Cluster Configuration Settings for more information on this matter.

4

Cluster DB Configuration Settings

Summary

Configuration Settings for Cluster DB

Description

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". 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 [eZDFS-ClusteringSettings].

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 ar 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
```

4.5.3 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 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=eZDBFileHandler
DBBackend=eZDBFileHandlerMysqlBackend
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 "eZDBFileHandlerMysqlBackend" 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
<?php
define( 'STORAGE_BACKEND',      'mysql'     );
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 '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 will improve performances. 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/classes/clusterfilehandlers/dbbackends/mysql.php
```

The MySQL table definitions when using eZ DB File Handler can also be imported directly from:

```
(...)/kernel/sql/mysql/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.

4

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 rewrit-
ten 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 dis-
played, 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 clusterized. 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

4.5.4   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 con-
tents 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
DBBackend=eZDFSFileHandlerMysqlBackend
```

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 "eZDFSFileHan-
dlerMysqlBackend" (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.

Note: Folder indicated in *MountPointPath* shouldn't contain anything but files handled by eZ Publish cluster since files within this folder are maintained by cluster maintenance scripts (page 320), and can be potentially removed. If you need to store files here (i.e. custom cache files), be sure to use *eZClusterFileHandler* in your own PHP code.

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
<?php
define( 'STORAGE_BACKEND',          'dfsmysql'   );
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.

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]
```

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 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 clusterized. 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

4.5.5   Reverting a cluster setup

You can always switch from using the clustering 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=eZDBFileHandlerMysqlBackend
#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 307)" or "Setting it up for an eZ DFS FileHandler (page 312)" to cluster your installation with the new cluster file handler.

4

4.5.6   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 ourIssue 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.

Note: ANY file located in *MountPointPath* is likely to be removed. Thus any file that is NOT contained in eZ Publish cluster will be removed if using -D switch. Please read DFS installation guide (page 312) and issue #19059 for more information.

Two new options have been added to dfscleanup.php: *iteration-limit* and *path*. The first one limits how many items are fetched at the same time when purging, and the second one limits the path we check items for, like *var/ezwebin_site/storage*.

```
--path=VALUE                      Path to limit checks to (e.g.: <span
class="code-keyword">var</span>/storage/
content - Default: <span class="code-keyword">var</span>/)
--iteration-limit=VALUE       Amount of items to remove in each iteration when
performing a purge operation. Default is all in one iteration.
```

4

4.6   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 are 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

4.6.1   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 177) 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 (page 60) 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/4.0. 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

4.6.2   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 4.26)

Figure 4.26: *The "Local" system repository is empty.*

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 4.27)
   Choose the "Content class export" wizard as shown in the screenshot and click the "Create package" button.

Figure 4.27: *The package creation dialog.*

3. The package wizard starts from asking what classes to include to the package being created
   (look at the next screenshot).
   (see figure 4.28)
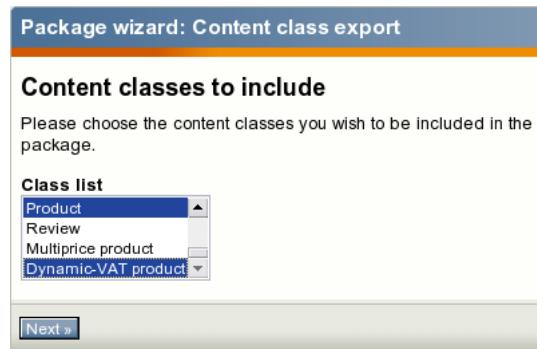
Figure 4.28: *The content class export dialog.*

Select the desired class(es) from the list and click the "Next" button.

4. 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 4.29)

5. The system also needs some information about the package maintainer. Enter this informa-
   tion and click the "Next" button.
   (see figure 4.30)

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 4.31)
   After clicking the "Continue" button the wizard will create the package and display its
   summary.

Figure 4.29: *The package creation wizard: package information step.*

Figure 4.30: *The package creation wizard: information about the package maintainer.*

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

Figure 4.31: *The package creation wizard: changelog.*

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 4.32)

Figure 4.32: *The package creation dialog.*

2. The package wizard starts from asking which objects to include to the package being created (look at the next screenshot).
   (see figure 4.33)
   The following text describes how you can use the "Add node", "Add subtree" and "Remove selected" buttons for choosing the desired objects.

   • The "Add node" button makes it possible to add individual objects to the package

Figure 4.33: *The content object export dialog (no objects selected).*

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 "New article" is
selected.
(see figure 4.34)
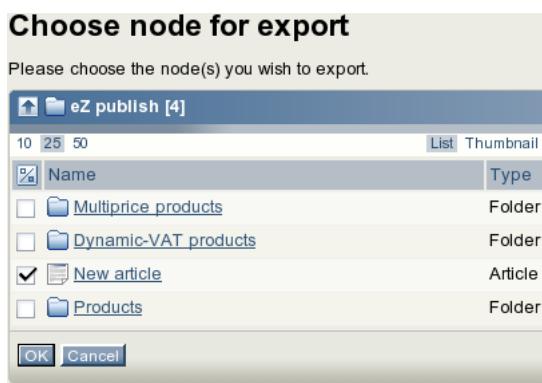
Figure 4.34: *Browse the content tree and select which nodes that will be exported.*

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 dia-

log 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 4.35)
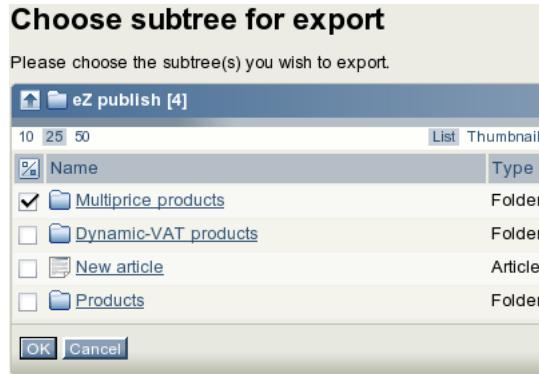
Figure 4.35: *Browse the content tree and select which subtrees that will be exported.*

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 4.36)

Figure 4.36: *The content object export dialog (one node and one subtree selected).*

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 how this dialog looks like.
(see figure 4.37)
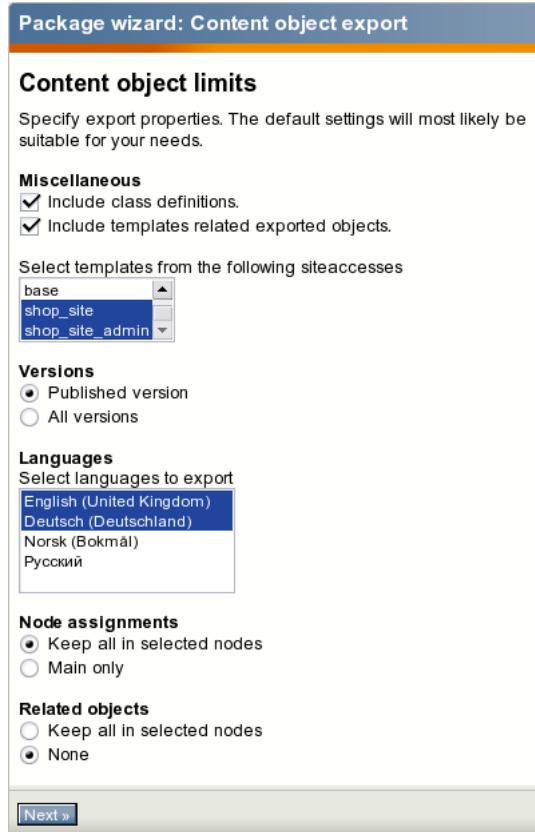
Figure 4.37: *The content object package creation wizard: export properties for content objects.*

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 4.38)*

Figure 4.38: *The extension package creation wizard.*

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 4.39)*
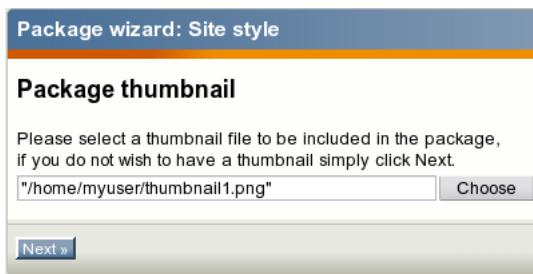
Figure 4.39: *The site style package creation wizard: choose thumbnail.*

Choose the image file and click the "Next" button.

3. 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 177) 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 4.40)*
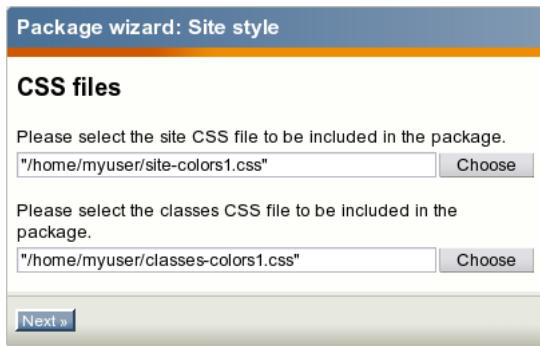
Figure 4.40: *The site style package creation wizard: select CSS files.*

4. If you use images in your theme, you can upload them in the next screen (look at the screenshot). *(see figure 4.41)*
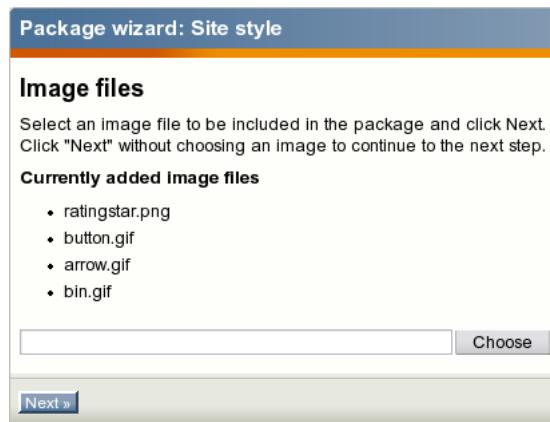
Figure 4.41: *The site style package creation wizard: add images.*

Click "Next" when you finish adding images.

5. 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.

4

4.6.3   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 4.42)

Figure 4.42: *The list of packages.*

   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 4.43)
   Click the "Export to file" button in order to download the ".ezpkg" file.

Figure 4.43: *The package summary view interface.*

4.6.4 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 4.44)*

Figure 4.44: *The list of packages.*

2. Choose the desired ".ezpkg" file on your local computer (as shown in the following screenshot) and click the "Import package" button.
(see figure 4.45)
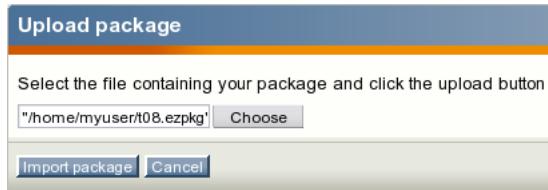
Figure 4.45: *The import package interface.*

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" subsection to learn how the design theme from the imported site style package can be applied to your site.

4.6.5   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 4.46)

Figure 4.46: *Removing a package.*

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.

4.6.6   Installing packages
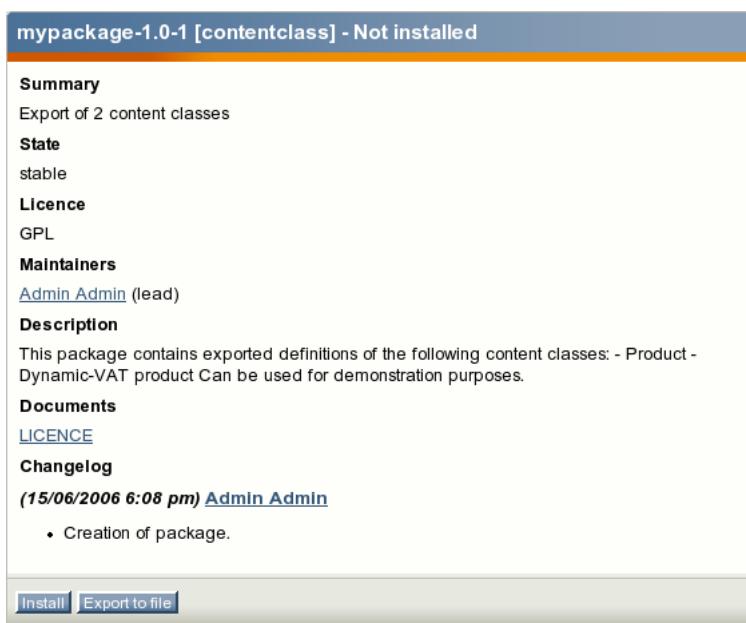
It is possible to install packages that are located under internal repositories. Note that this is not true for site packages and design packages. When you install a package, the system will create content classes and content objects, apply configuration settings and so on. Please note that from eZ Publish 3.8, information about installed packages is stored in the "ezpackage" table within the database.

The following subsections 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. The system will display the package summary as shown in the following screenshot.
   (see figure 4.47)

Figure 4.47: *The content class package summary.*

   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).
   (see figure 4.48)
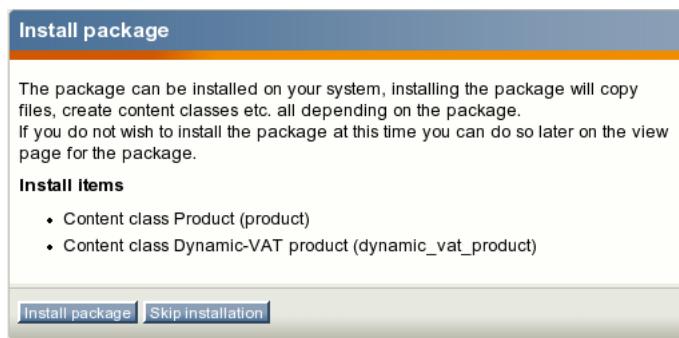
Figure 4.48: *The content class package installation wizard, step 1.*

Read this information carefully and click the "Install package" button to continue. Use the "Skip installation" button to abort the operation.

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 4.49)
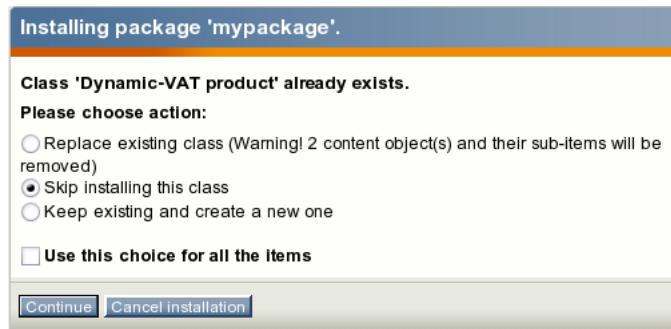
Figure 4.49: *The content class package installation wizard, step 2.*

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 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 subsection, 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. Find the package you wish to install, click on its name and 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 4.50)
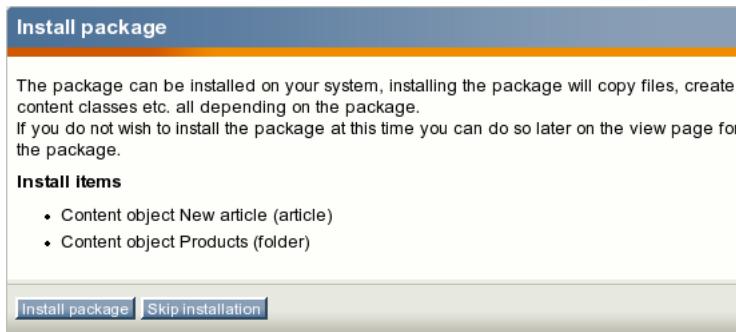
Figure 4.50: *The content object package installation wizard, step 1.*

Read this information carefully click the "Install package" button. Use the "Skip installation" to abort the operation.

3. 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 4.51)

Figure 4.51: *The content object package installation wizard, step 2.*

Make your choice and click the "Next" button.

4. The next dialog reveals where the installed objects will be located and allow to choose another location if needed.
   (see figure 4.52)
   Choose the desired location and click the "Continue" button.
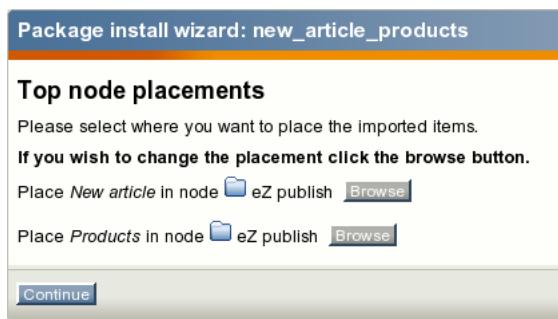
Figure 4.52: *The content object package installation wizard, step 3.*

5. If some of the objects being installed already exist (i.e. there is another object with the same remote_id (page 1032)), the system will ask how this installation conflict should be handled (see the screenshot below).
(see figure 4.53)
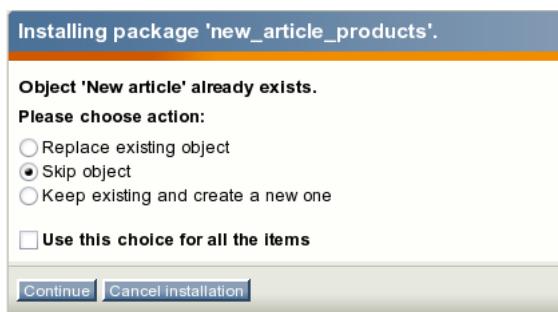
Figure 4.53: *The content object package installation wizard, step 4.*

After clicking the "Continue" button, the system will install the package and display a summary.

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 4.54)
Click the "Install package" button to continue. Use the "Skip installation" button to abort the operation.
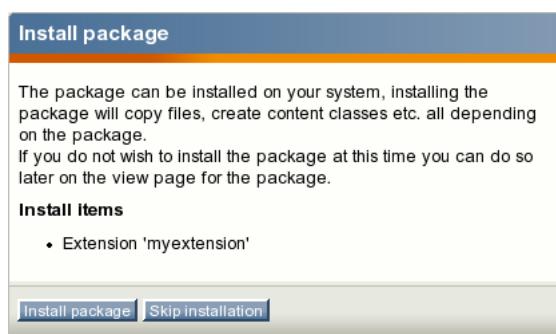
Figure 4.54: *The extension package installation wizard, step 1.*

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 4.55)
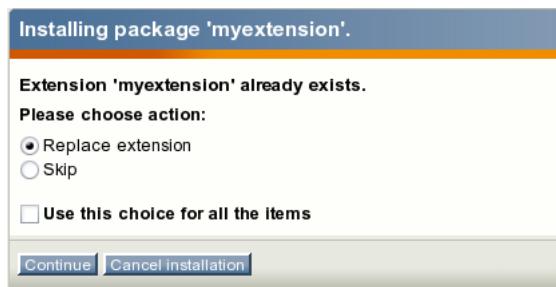
Figure 4.55: *The extension package installation wizard, step 2.*

After clicking the "Continue" button, the system will install the package and display a summary.

4

4.6.7 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 437), 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 4.56)
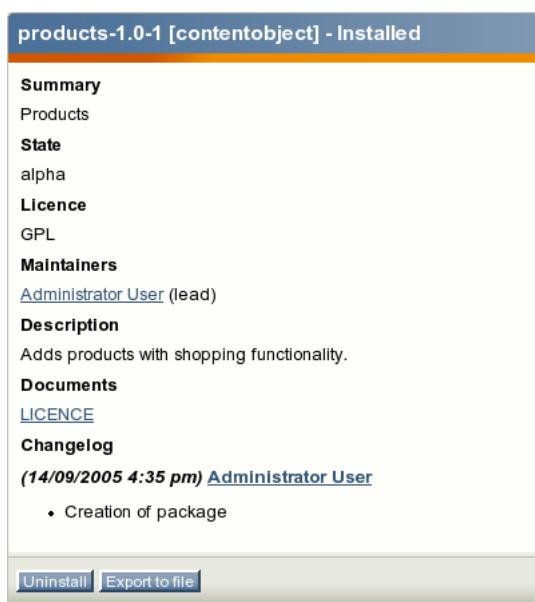
Figure 4.56: *The package uninstallation wizard, step 1.*

   Click the "Uninstall" button.

2. The system will display a list of items that will be removed (look at the next screenshot).
   (see figure 4.57)
   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 125).

3. If some of the items being removed have been modified after the package installation, the system will ask for confirmation before removing them (as shown in the following screenshot).
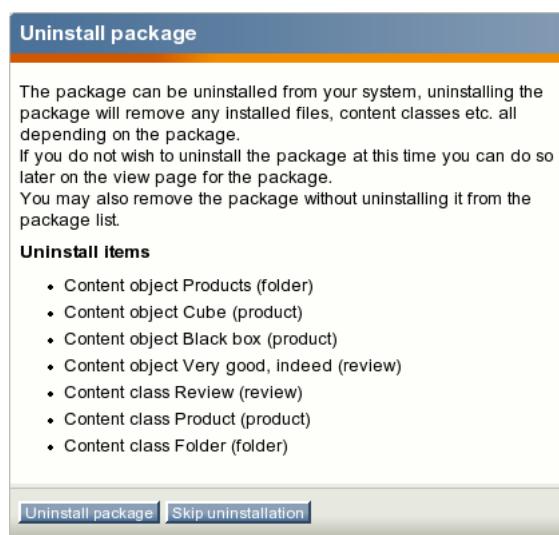
Figure 4.57: *The package uninstallation wizard, step 2.*

(see figure 4.58)
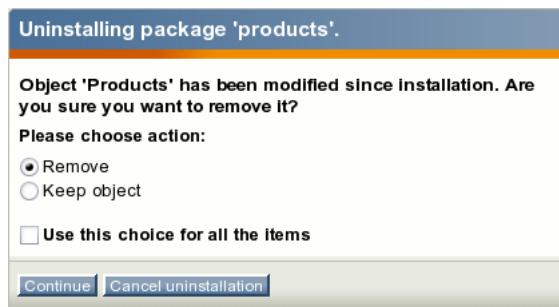
Figure 4.58: *The package uninstallation wizard, step 3.*

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.

4.6.8   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

4.6.9   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 installa-
tion/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 4.59)

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:
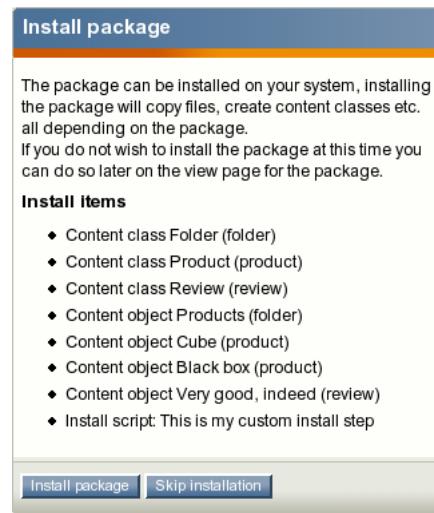
Figure 4.59: *Displaying a custom install script in the list of items during the package installation process*

```
<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 4.60)

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 (according to the description given in the "post-install/myinstallscript.xml" file). Add the following lines to the "myinstallscript.php" file:
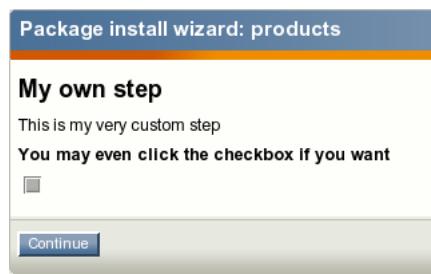
Figure 4.60: *Displaying a custom wizard step during the package installation process*

```php
<?php
class myInstallScript extends eZInstallScriptPackageInstaller
{
  function myInstallScript( &$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( &$package, &$http, $step, &$persistentData,
&$tpl, &$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

4.7   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 167), 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 463), 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 364)" and "Running cronjobs (page 367)" sections for further details.

4

4.7.1   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

4

	Apache log files.		
workflow.php	Processes the work-flows.	Every 15-30 minutes.	Enabled

Cleaning up expired data for webshop

The eZ Publish webshop (page 162) 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 (page 2461)" 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 (page 2462)" 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 (page 524)" 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 it's 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 4.61)

Figure 4.61: *Class attribute edit interface for the "Date and time" datatype.*

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 (page 1737)" 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 (page 1738)" configuration directive.

Delayed search indexing

If the delayed indexing (page 2449) 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" (page 601) or the "URL" (page 597) datatype is stored in the URL table (page 134) 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 (page 1800)" configuration directive located in the "[linkCheckSettings]" section of the "crobjob.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] (page 2416)" section of "site.ini" (requires CURL support in PHP).

Sending notifications

eZ Publish has a built-in notification system (page 463) 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/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] (page 2416)" section of "site.ini" (requires CURL support in PHP).

Clearing expired template block caches

If you are using the "cache-block (page 1383)" 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 (page 2537)" 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 (page 524)" 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 (page 1766)" configuration array and put the ID number of the parent node for your articles to the "RootNodeList (page 1767)" 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 (page 780)" 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 167), 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 132) can not be published without your approval. (This can be done by creating a new "Approve" event (page 1128) 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.

4.7.2   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 (page 1804) configuration array specifies the directories where eZ Publish will search for built-in cronjob scripts (the "cronjobs" directory is used by default).

- The ExtensionDirectories (page 1802) directive specifies the extension directories where eZ Publish will search for additional/custom cronjob scripts. By default, eZ Publish will search in the "cronjobs" subdirectory inside your extension(s).

- The Scripts (page 1803) array contains a list of cronjob scripts that will be run when the main "runcronjobs.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

4.7.3   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 364)" 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 (page 2527) 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
`crontab /var/www/ezpublish/ ezpublish.cron`	Install a new crontab from the "ezpublish.cron" file (replace "/var/www/ezpublish" by the actual path to your eZ Publish directory).
`crontab -l`	Display the current crontab.
`crontab -e`	Edit the current crontab. The modified crontab will be installed automatically.
`crontab -r`	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.

4.8 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 (page 593)" 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 (page 593)" 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 user John 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 4.62)

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:

4
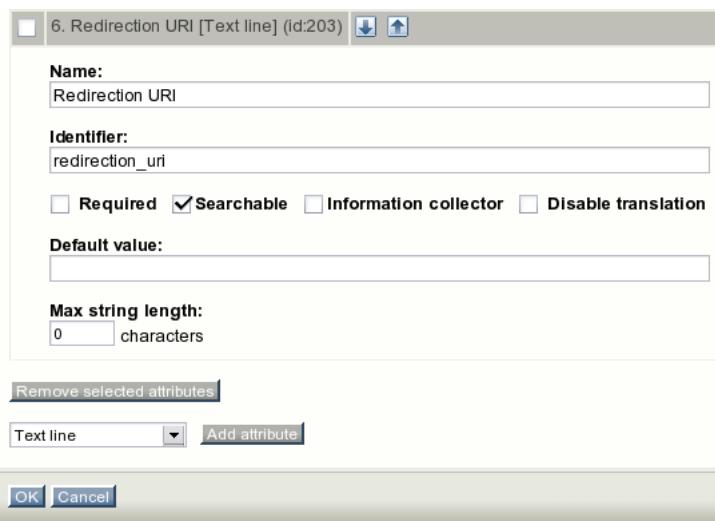
Figure 4.62: *A fragment of the class edit interface.*

```
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 John 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 4.63)

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 4.64)

2. Add the following line into the "[UserSettings]" section of an override for the "settings/ site.ini" configuration file:

```
LoginRedirectionUriAttribute[group]=start_page
```

where "start_page" is the attribute identifier.

Figure 4.63: *Setting the redirection URI for the user John*

Figure 4.64: *A fragment of the class edit interface.*

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. 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 login.

(see figure 4.65)

Figure 4.65: *Setting the redirection URI for the "Guest accounts" user group*

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 4.66)

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
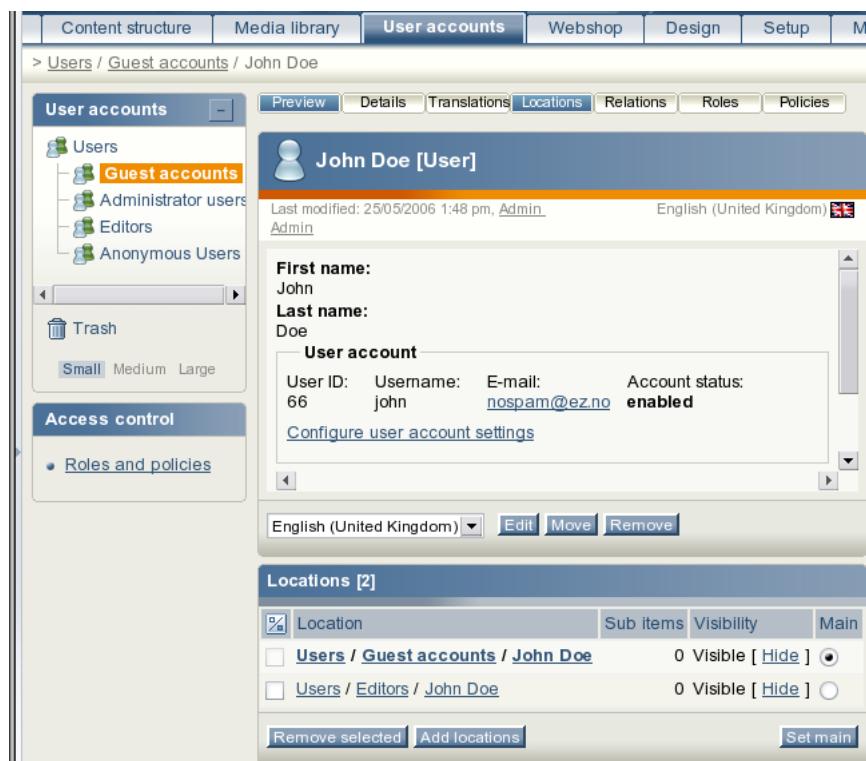
Figure 4.66: *A fragment of the object view interface for the user with two locations.*

4.9   VAT charging system

Charging the value added taxes in your webshop 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 (page)" section. Although the quantity of the VAT types is not limited, there must be at least one VAT type in your webshop 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 (page)" 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 (page 400)" 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

4.9.1 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 (page 582) and multi-price (page 570) datatypes).

(see figure 4.67)
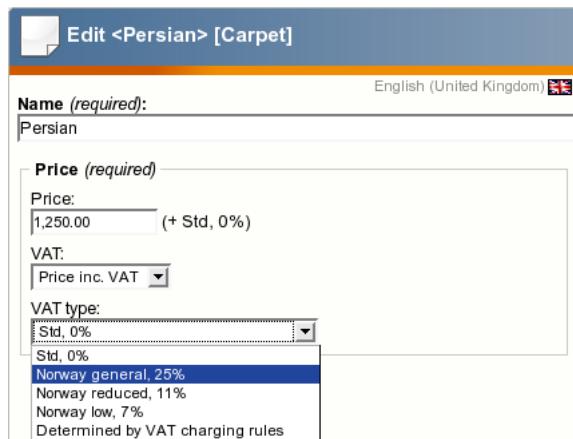
Figure 4.67: *Setting the VAT type on the object level.*

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 4.68)

Figure 4.68: *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.

4.9.2 Three approaches to VAT charging

The webshop system (page 162) 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 377) 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 386)

- Product category (page 384) (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 397)" 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 (page) 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)" 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 (page 520) 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 (page 584) 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 377)" 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 400)" 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 402)" section for more information.

4

4.9.3 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 394)" section.

Adding new attribute to a product class

It is necessary to add an attribute of the product category (page 584) 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 (page 584) datatype.
   (see figure 4.69)
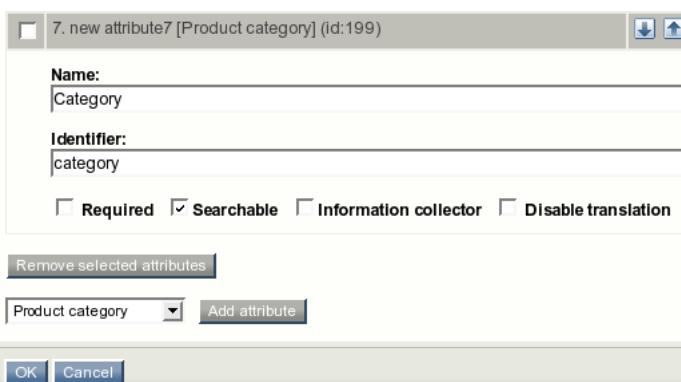
Figure 4.69: *Class attribute edit interface for the "Product category" datatype.*

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 (page 584) 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 4.70)

Figure 4.70: *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.

4.9.4   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 (page 1788)" configuration file.

Adding new attribute to a user class

It is necessary to add an attribute of the country (page 520) 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 (page 520) datatype.
   (see figure 4.71)

Figure 4.71: *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.

3. 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 (page 2334)[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=''
```

4

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 (page 520) 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

4.9.5   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 400)" 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 (page 520) 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 it 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 (page 520) 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

4.9.6   Managing VAT types

This section describes how you can add, remove and modify static VAT types (page 375) 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 4.72)

Figure 4.72: *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 4.73)

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.

Figure 4.73: *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 4.74)
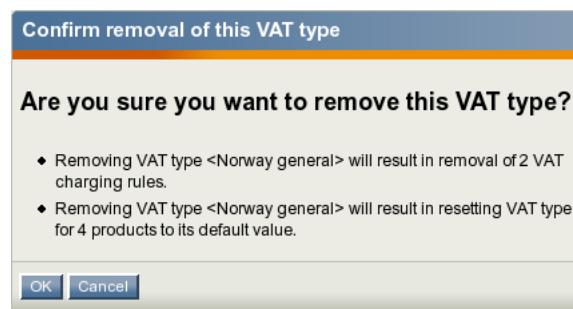
Figure 4.74: *The confirmation dialog.*

4

4.9.7 Managing product categories

This section describes how you can add, remove and modify product categories (page 384) 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 4.75)

Figure 4.75: *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 4.76)

Figure 4.76: *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 4.77)

Figure 4.77: *The confirmation dialog.*

4.9.8   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 375) and product categories (page 384) (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 4.78)

Figure 4.78: *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 4.79)

Figure 4.79: *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 4.80)

Figure 4.80: *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.

4

4.9.9 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

4.9.10   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
<?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.

4.10   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 ship-
ping handler.

4.11   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 2032) 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 DNs 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 LDAPEmailEmpty-AttributeSuffix. 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

4.11.1   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 LDAPUser-GroupAttribute to the LDAP attribute that holds the group information. Also, LDAPUserGroupAt-tributeType 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

4

	employeetype:cn="Technical writers",dc=Groups,dc=example,dc=com
Maps users to eZ Publish groups	janedoe: 'Editors' 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.

4.11.2 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

4.11.3   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 not 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 LDAPUserGroupAttribute-Type 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 ezp 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 caracters unread

When LDAPLoginAttribute and LDAPGroupNameAttribute contain upper case caracters 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

4.12 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 (page 162)" 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

4.12.1   Custom prices and auto prices

The multi-price datatype (page 570) 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 423) 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 423). 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 4.81)

4

Figure 4.81: *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 4.82)

Figure 4.82: *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 4.83)

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
☐	EUR	74.45(Auto)
☐	USD	88.63(Auto)
☐	NOK	600.00

Remove selected

Figure 4.83: *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

4.12.2 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

4.12.3   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 442)" 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 442)" 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.

4.12.4   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 4.84)

Figure 4.84: *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 4.85)*

Figure 4.85: *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 4.86)*

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.

Figure 4.86: *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 4.87)*

Figure 4.87: *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 4.88)*

Figure 4.88: *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 "curren-cynames.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 418) 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 418) 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.
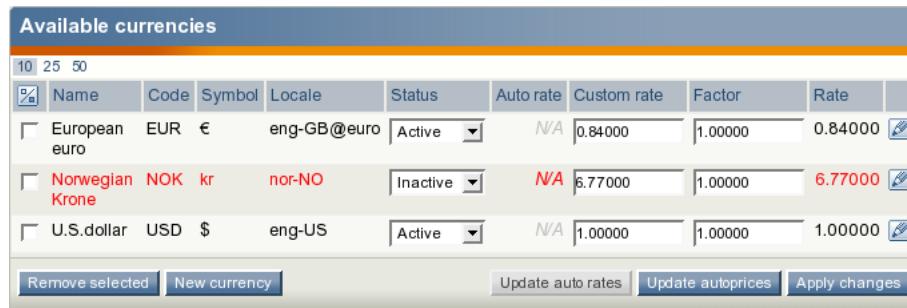
Custom rate	Rate factor	Auto rate	Final rate
0	1	0.85	0.85
0	1.4	0.85	0.85*1.4=1.19
0.75	1	0.85	0.75
0.75	1.4	0.85	0.75*1.4=1.05

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 4.89)

Figure 4.89: *Displaying inactive currency in the list of currencies.*

4.12.5   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.

2. 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 4.90)

Figure 4.90: *The list of currencies with disabled possibility to update auto rates.*

3. 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 442)" section and clear the eZ Publish caches. The "Update auto rates" button will become active.

4. Click the "Update auto rates" button. The auto rates will be automatically updated.
   (see figure 4.91)

Figure 4.91: *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 4.92)

Figure 4.92: *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 4.93)
   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.

Figure 4.93: *The list of currencies with one custom rate.*

4.12.6 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

4.12.7   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 441) interface.

4

4.12.8 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 (page 582) or multi-price (page 570) 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 (page 570) 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 (page 445)" 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 4.94)
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 4.95)
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 (page 418) 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 4.94: *The class edit interface for a product class.*

Figure 4.95: *Class attribute edit interface for the "Multi-price" datatype.*

system will create a base price (page 418) 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 (page 441) 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
```

4

```
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 (page 2334)[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).

4.12.9   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 435) are displayed for multi-price products.

(see figure 4.96)

Figure 4.96: *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

4.12.10 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/ezecb/ezecbhandler.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 eZECB 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.

4.12.11   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 425) 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 (page 582) datatype and create an attribute of the multi-price (page 570) 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 4.97)

Figure 4.97: *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

4.13   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 (page 821)" and "pdf (page 802)" views of the "content (page 679)" 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 173)) is put to "$module_result.content" and is included in the pagelayout (page 177) by accessing this template variable:

```
{$module_result.content}
```

When the pagelayout 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 pagelayout. Please note that the pagelayout itself is not cached by default.

Another thing worth noticing is that the view caches are also depending on some other parameters. For example:

4

- 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 173). 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 mod-
ule 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 ac-
cessed, 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

4.13.1 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 173) 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 (page 2336) 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}
```

4

Roles

The cache files are different for dissimilar role combinations. This means that the templates can have conditions based on roles (page 158) even when view caching is on.

User preferences

The following text describes handling of user preferences (page 1171) 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 (page 995)" view of the "user (page 974)" 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 (page 2334) setting located under the "[ContentSettings]" section of the "site.ini.append.php" configuration file.

4

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 435). 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 (page 2511) 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 (page 2384) configuration setting to be the same for all related siteaccesses.) If the RelatedSiteAccessList setting is not specified, the cache subsystem will use the AvailableSiteAccessList (page 2490) setting instead.

Note that the AvailableSiteDesignList (page 1781) 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

4.13.2 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 (page 553)" 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 (page 2765)" configuration setting.

This default behavior can be extended by configuring the smart viewcache cleaning system (page 456).

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 (page 2511).

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 4.98)

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.
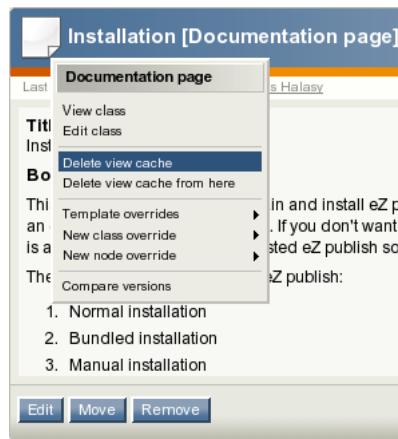
Figure 4.98: *Clearing the view cache using popup menu.*

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:

4

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

4.13.3 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 (page 553)" 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 (page 2765)" configuration setting.

If you want to use the smart viewcache cleaning feature, make sure the "view-cache.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 (page 1051) 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 predefined 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) of the object.

		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 (page 2765)" configuration setting). **Name:** keyword **Description:** Clear the view cache for the objects that have the same keyword as this object. **Name:** siblings **Description:** Clear the view cache for all the siblings of this node/object. **Name:** all **Description:** Clear the view cache for all the listed above.
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.

4

Example 1

Let's say that both view caching and svcs are enabled with the following part of a content structure:

(see figure 4.99)
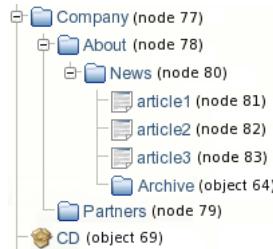
Figure 4.99: *A part of the site content structure.*

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 "Object-Filter[]" 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 "viewcache.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

4.13.4   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 (page 2329) 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 (page 2328) 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 (page 2327) setting located in the "[ContentSettings]" section of the "site.ini" configuration file.

4

4.14   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 132) can not be published without your approval. (This can be done by creating a new "Approve" event (page 1128) within a new workflow (page 167) 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 364)" and "Running cronjobs (page 367)" 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 4.100)

Figure 4.100: *The notification filter interface.*

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 (page 841) should only be used for testing and debugging.

4

4.14.1   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 subitems.
   (see figure 4.101)

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 4.102)*

3. The system will add a new subtree notification and show you a confirmation:
   (see figure 4.103)

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 "My Account" 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 4.104)
   This 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.

3. The system will bring up the browse interface which will allow you to select the desired nodes:
   (see figure 4.105)

4
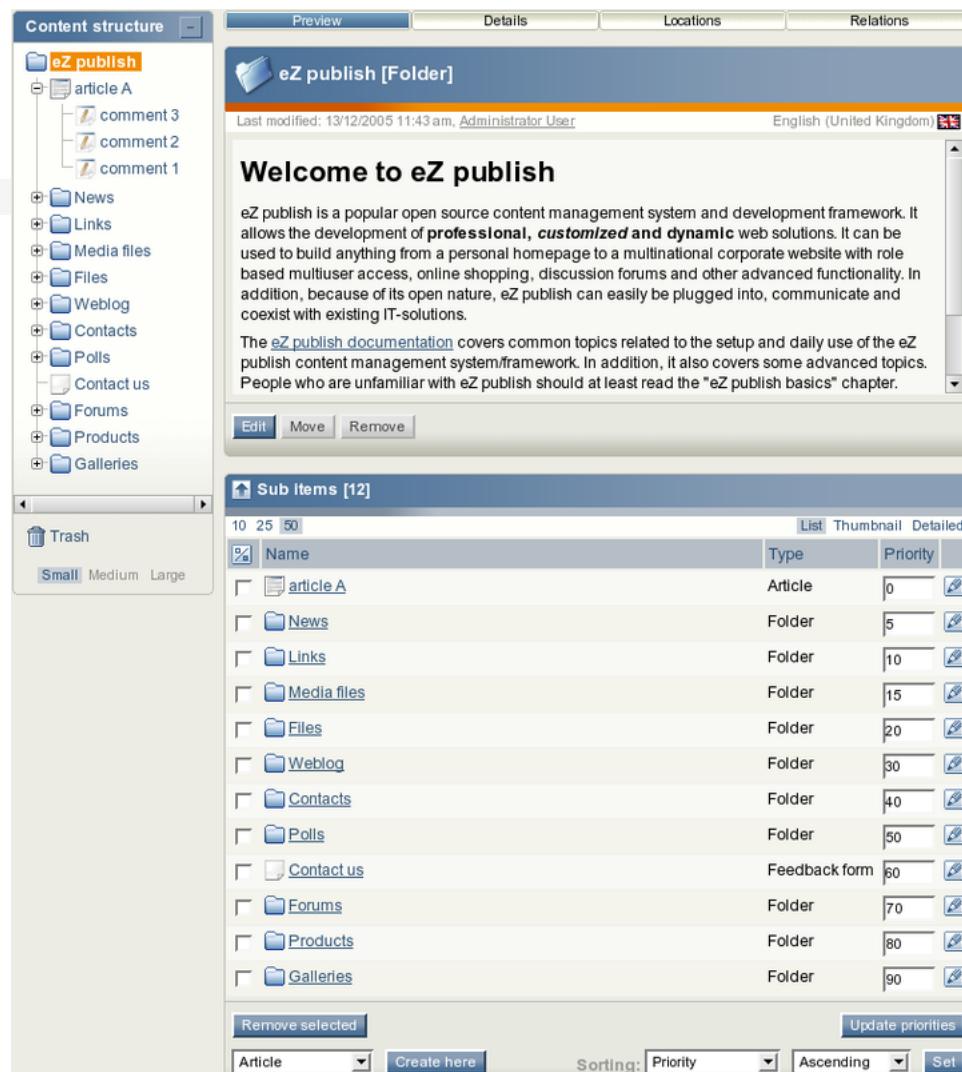
Figure 4.101: *Browsing the content tree.*

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 4.106)*

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 these
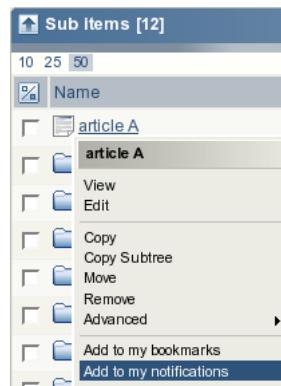
Figure 4.102: *Subscribing to subtree notifications using the context menu.*
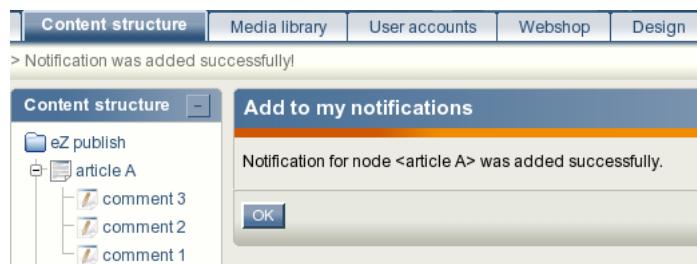
Figure 4.103: *The "notification added" confirmation for administrators.*

objects and add them to the "My item notifications" list.

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 "My Account - 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 4.107)
   To enable the digest mode, select the "Receive all messages combined in one digest" checkbox 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).

Figure 4.104: *Notification settings for administrators.*

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 "My Account" 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 4.108)*

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).

4
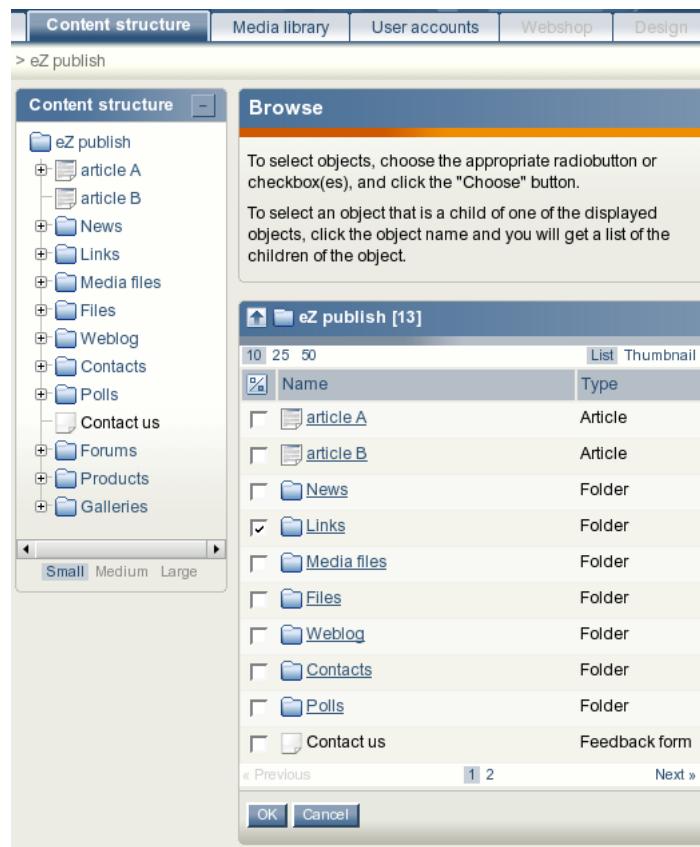
Figure 4.105: *Browsing the content tree.*

Figure 4.106: *The "Up" button*

Figure 4.107: *Digest settings*

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

Figure 4.108: *The list of items for subtree notifications.*

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 "My Account" 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 4.109)

Figure 4.109: *Settings for collaboration notifications.*

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.

4.14.2 Using an actual site

Subtree notifications are available for those users who are allowed to use the "notification" module (page 841) by the role/policy settings. Please refer to " Granting access to notifications (page 476)" 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 4.110)

Figure 4.110: *The "keep me updated" button.*

After clicking this button, the system will add a new subtree notification and show a confirmation:

(see figure 4.111)

Figure 4.111: *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 474)" 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 4.112)

Figure 4.112: *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

4.14.3 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}
```

4.14.4   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/" di-
rectory 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

4.14.5 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 (page 841).

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 4.113)

Figure 4.113: *The usergroup view interface.*

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 (page 841) 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 4.114)

Figure 4.114: *The list of roles.*

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 4.115)*
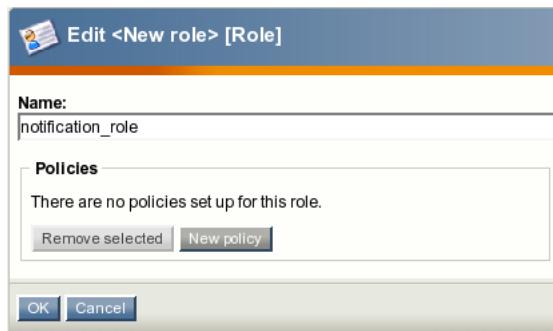
Figure 4.115: *Adding a new role.*

3. Specify the name of the role and click the "New policy" button.

4. The wizard will help you to create a new policy in two steps.
   (see figure 4.116)
   The above screenshot shows the first step. Select the "notification" module from the drop-down list and click the "Grant access to one function" button.
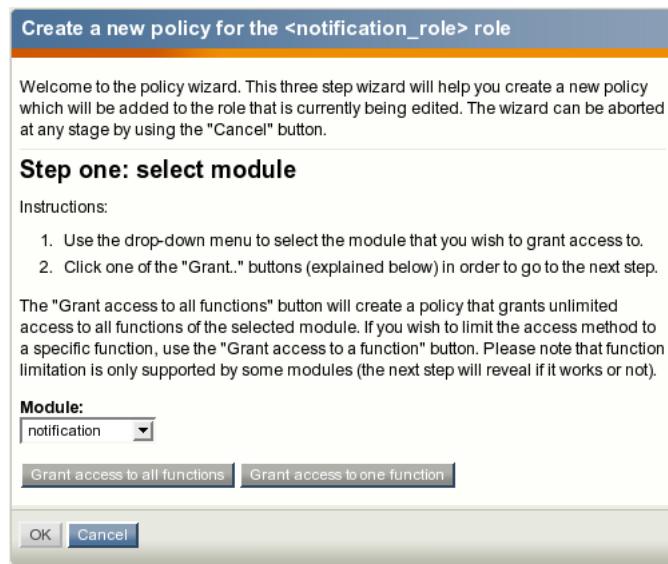
Figure 4.116: *The new policy wizard, step 1.*

5. You will be taken to the second step as shown in the screenshot below.
   (see figure 4.117)
   Choose the "use" function from the 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 (page 841).

6. 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.)

7. The new policy will appear in the role edit interface as shown in the following screenshot.
   (see figure 4.118)

8. Click "OK" to save your changes and go back the role view interface.
   (see figure 4.119)
   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.
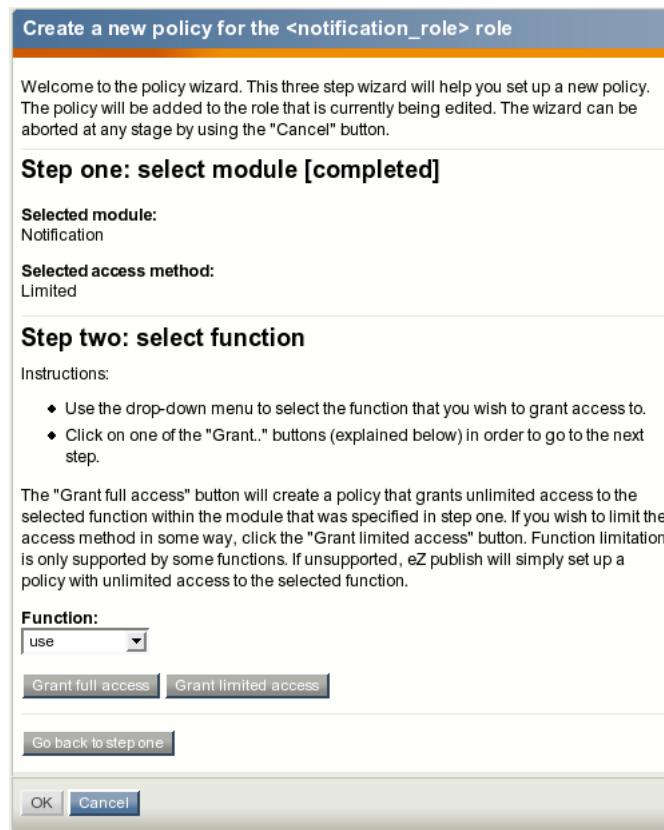
Figure 4.117: *The new policy wizard, step 2.*

Figure 4.118: *The role edit interface.*

2. Select the "Editors" user group as shown in the following screenshot and click the "OK" button.
   (see figure 4.120)

3. The "Editors" user group will appear in the list of users. The screenshot below shows the
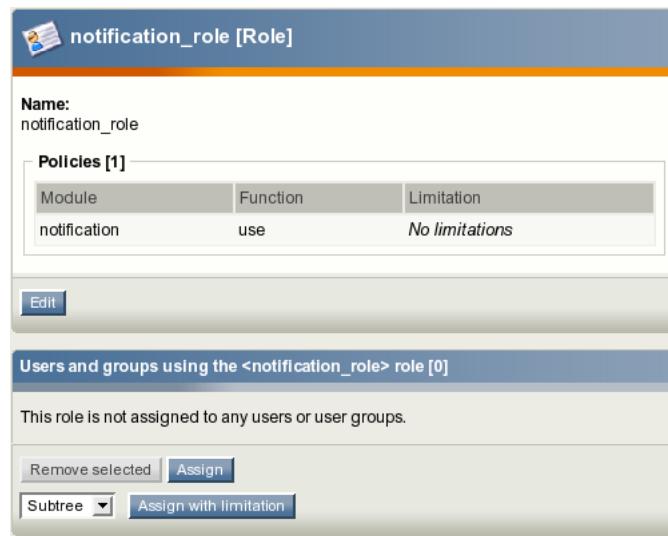
Figure 4.119: *The role view interface.*

Figure 4.120: *Assigning a role to a user group.*

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)*

Please note that you can assign the role to a single user in the same way as to a user group.

Figure 4.121: *The role view interface.*

4.14.6   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 (page 841) 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 485).

Settings

The available notification event types are specified in the "[NotificationEventTypeSettings]" section of the " notification.ini (page 2205)" 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 (page 2205)" 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 (page 2205)" 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.

4.14.7   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/" direc-
tory. 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 482) 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 (page 2205)" 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 (page 2205)" 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 (page 2205)" 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

4.14.8 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 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 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 167)" and "Approve (page 1128)" documentation chapters for more information about workflows, triggers and approval events. (A simple example of implementing an approval mechanism including creating a workflow, connecting it to a trigger function and approving entries can be found in the old documentation.)

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 476)" 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 476)" 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" section of the "Granting access to notifications (page 476)" 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 (page 841).

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 (page 841). 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.

4.15   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.

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" checkboxes while editing a class. Some datatypes (for example float, price, etc.) do not support indexing. Please refer to the datatype overview (page 514) 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 4.122)

Figure 4.122: *Standard search interface*

Advanced search

The advanced search interface makes it possible 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).

- 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 4.123)

Figure 4.123: *Advanced search interface*

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 4.124)

Figure 4.124: *Search statistics*

The "Reset statistics" button will simply clear the search log.

4.16   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 4.125)
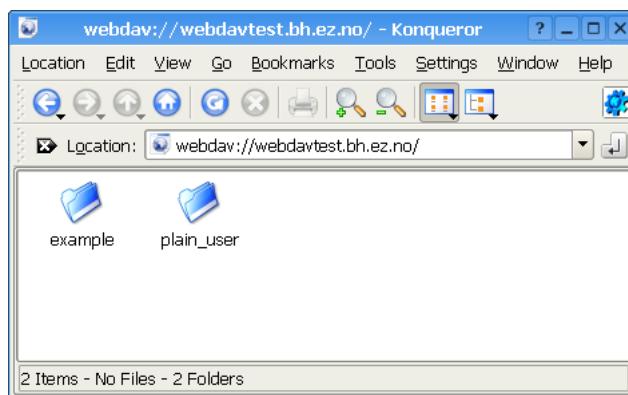
Figure 4.125: *WebDAV - Virtual top folder*

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 4.126)

Figure 4.126: *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 4.127)
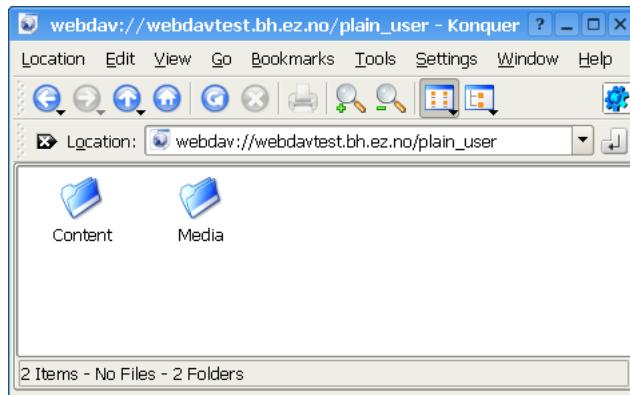
Figure 4.127: *WebDAV - Top level nodes*

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 4.128)
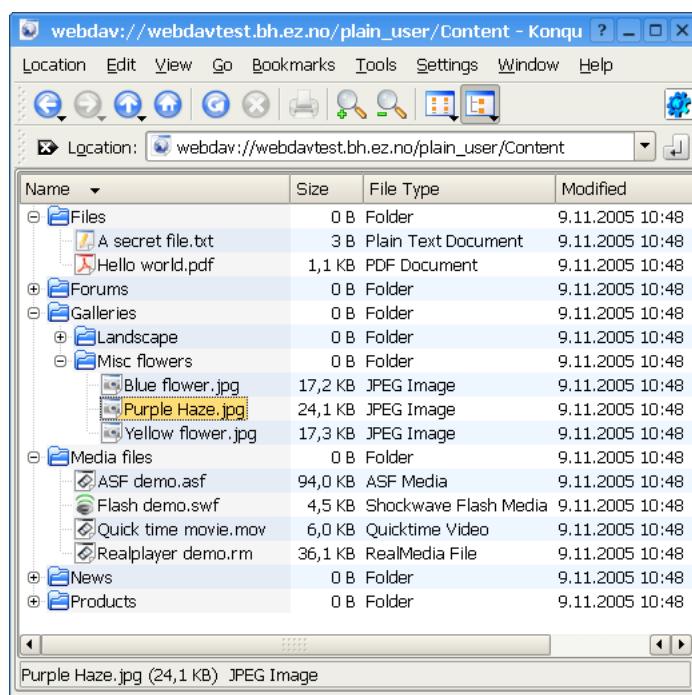
Figure 4.128: *WebDAV - Content node tree*

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.

4

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 "CreateSettings" 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

4.16.1   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 Web-DAV 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 4.129)
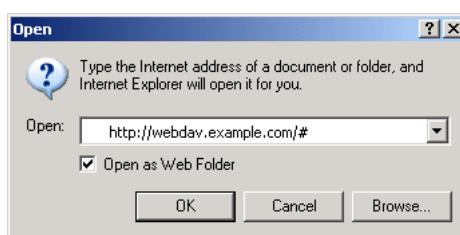
Figure 4.129: *WebDAV - IE open dialog*

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 4.130)
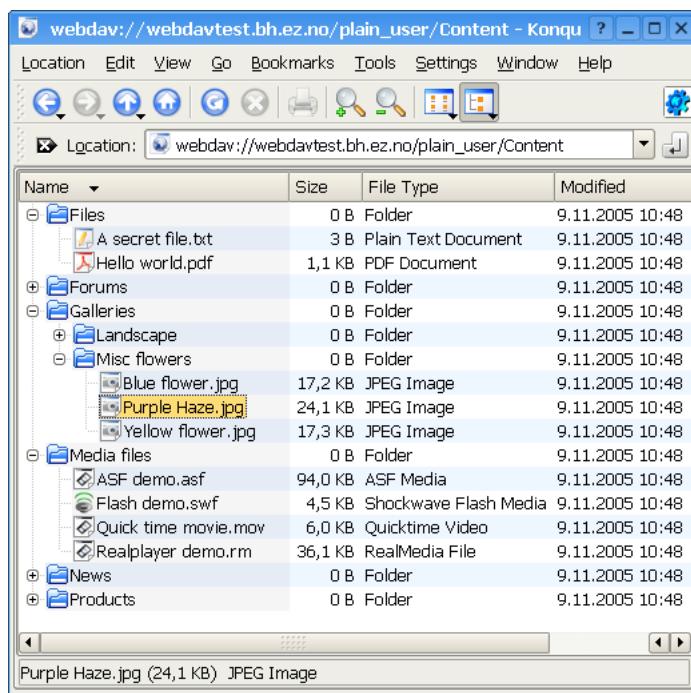
Figure 4.130: *WebDAV - Content node tree*

4.17   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 usergroup 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 4.131)

Figure 4.131:

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 by applied as default state.

(see figure 4.132)

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 4.133)

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 4.134)

4

Figure 4.132:

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.

(see figure 4.135)

This can also be done with an already published article:

Figure 4.133:

(see figure 4.136)

In the Administrator Interface, the state can be altered when you navigate to the "Content

Figure 4.134:

Figure 4.135:

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 4.137)

For another example of the use of object states, please read "eZ Publish Knowledge Series: Editorial workflow with Object States".

Figure 4.136:

Figure 4.137:

4.18 Language switcher

Since eZ Publish 4.1, there is a new way to create links that take the users to an other traslation 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

Chapter 5

Reference

This chapter provides reference information for developers. It covers the topics listed below. (Note that the eZ Publish kernel API documentation for PHP developers is not covered by this manual. You can find it here.)

\begin{itemize} \item{} Database schema\refer{labelf5b4f52c51f57d81f31849678540de20} \item{} Datatypes\refer{labelfbf63b97fc7db5f185527861bf2e37b0} \item{} Content classes\refer{labelee7be1c686d118fe47ba2fa34a5c8385} \item{} Modules\refer{label0dc2b3cc37e920d95213118ad0619602} \item{} Views\refer{labela91e57d683de7168ddebf265f87759eb} \item{} Objects\refer{labelele908ba8803955b3ddab6cd75eac49b6b} \item{} Workflow events\refer{labelf497f09f5938fba4f3a493047d40d2f5} \item{} Template operators\refer{label93c66bfd6d8e2a68b6ce731d7efeec6a} \item{} Template functions\refer{label1c320a2964295d9eaf65037b8f9aaf68} \item{} Template control structures\refer{label653bbe6182b4e1014f4d52a9c7dc970d} \item{} Template override conditions\refer{label86674db492fcf423539e0f296f758af4} \item{} Template fetch functions\refer{label537850e30950698b699fe351eb910059} \item{} Template PDF functions\refer{label563a47bb68632de65a957c9e3faf925c} \item{} Configuration files\refer{label6e45015029f6952ed62a834d440e56c5} \item{} Libraries\refer{labelele50bba9b52fc21b4844149d38484a40c} \item{} XML tags\refer{labelcfd626fee1279312c52622cb5480e2e0} \item{} Scripts\refer{label9bb96e62c81374df9a12a41776ebfb55} \end{itemize}

5.1 Database schema

Below please find a link to the eZ Publish database schema, the blueprint of the eZ Publish database. The database schema defines the tables of the eZ Publish database and the relationship between them.

The database schema documentation contains tabs that will show you the database tables from various "angles".
Please know that the schema consists of drill-down elements, so you can click down to the next level of information.

- eZ Publish 4.5.0 database schema

- eZ Publish 4.4.0 database schema

5

5.2   Datatypes

\begin{description} \item [Authors]\refer{label8e60f0973670da92c9ef1741d820ea2a}\ \\
Stores info about additional authors.

\item [Checkbox]\refer{label0be0ddcb7c682ac00a6b1c3b6cb2d55d}\ \\ Stores a binary value
(on or off).

\item [Country]\refer{labeldeadf26d24f003e1d223ca3890686c5b}\ \\ Stores a user country.
[Webshop]

\item [Date]\refer{label787251cf2174068999cc97935ed9e319}\ \\ Validates and stores a date
value.

\item [Date and time]\refer{label65445eb36cf41e48fa912302c1d38f1d}\ \\ Validates and
stores a date and a time value.

\item [E-mail]\refer{label1b66adbd94b366d2addc4db07b09de02}\ \\ Validates and stores an
E-mail address.

\item [Enum]\refer{label3c8fb1042607d9911a3491a545a383d6}\ \\ DEPRECATED

\item [File]\refer{label562cb9dc6082963ea28ab3aef6235b4b}\ \\ Stores any type of file.

\item [Float]\refer{label30ff29978fbbb5be77f0115414e9527e}\ \\ Validates and stores a deci-
mal value.

\item [Identifier]\refer{labelf90dac865f60ec45c0b4656102489235}\ \\ Generates a non-
editable identification string.

\item [Image]\refer{label992f30a9ed986177624cc7b02554cdb5}\ \\ Validates and stores a
digital image.

\item [Ini setting]\refer{labela38a4e2bc082daa164bd097f5436d268}\ \\ DEPRECATED

\item [Integer]\refer{label5db435c257545b32f2d844d5c287e07c}\ \\ Validates and stores an
integer value.

\item [ISBN]\refer{labeldb4f7eb2d3468114d35c3c312338958a}\ \\ Validates and stores an
ISBN value.

\item [ISBN (as of 4.5)]\refer{labelaf8525b207229511e34005e0993f9510}\ \\ Validates and
stores an ISBN value.

\item [Keywords]\refer{label687644f622ad6ee949dec213e0d7de18}\ \\ Stores keywords.

\item [Matrix]\refer{label77c8bf77a86d6c7a108520def3985c04}\ \\ Stores multiple rows and
columns of text.

\item [Media]\refer{label99fe8ee8434b061f146cb4df8c39e92e}\ \\ Stores a media file (Flash/
QT/Real/etc.).

\item [Multi-option]\refer{label5ffba4b2521e6bee929fcdc5b1cd11cc}\ \\ DEPRECATED

\item [Multi-option2]\refer{label25135aff71cceb22d8c93300a9dc2e62}\ \\ Allows option se-
lections (supports enabling/disabling of options, images, nested option sets and dependencies).

[Webshop]

\item [Multi-price]\refer{label795e255af0e08d33d0adc440c9863840}\ \\ Stores prices in different currencies (inc/ex VAT). [Webshop]

\item [Object relation]\refer{label95823fe400c6fed6025b9f43100a56ec}\ \\ Stores a relation to a content object.

\item [Object relations]\refer{labele52e3c0d800163542f47cfd2618eb533}\ \\ Stores relations to other content objects.

\item [Option]\refer{labele0700a0b58e29ea978427319508f80ba}\ \\ Allows an option selection. [Webshop]

\item [Package]\refer{label6871855549b78acefab6d0f20d67d975}\ \\ DEPRECATED

\item [Price]\refer{label58c7a584d77085e7bb24ff12cb830881}\ \\ Stores a price (inc/ex VAT). [Webshop]

\item [Product category]\refer{label4d265ad9e26af323ccb20d4ddb274222}\ \\ Stores a product category. [Webshop]

\item [Range option]\refer{labelbab6b7824dfa3ce0794ec9887a4cb0cd}\ \\ Allows an integer selection. [Webshop]

\item [Selection]\refer{labelbb02b01aa890c7abf55133e0b4feb353}\ \\ Stores single and multiple choices.

\item [Subtree subscription]\refer{labela951d99ea3eb3afc61142b6b3d5248a8}\ \\ DEPRECATED

\item [Text block]\refer{label6c05446c3af953a04dcb86b51d288e06}\ \\ Stores multiple lines of unformatted text.

\item [Text line]\refer{label0ba742c99599ce5a97057816587ece0a}\ \\ Stores a single line of unformatted text.

\item [Time]\refer{label9151fb878069e877958b2e70394cd77f}\ \\ Validates and stores a time value.

\item [URL]\refer{label92d752d6987b9e630a90b3c047a47990}\ \\ Validates and stores a URL / address.

\item [User account]\refer{labela997c72fe13f55baecdc85e970e27460}\ \\ Validates and stores info about a user.

\item [XML block]\refer{label0ea3b8fc99977d13e85d0f11f6175c41}\ \\ Validates and stores multiple lines of formatted text.

\end{description}

5

5.2.1 Authors

Summary

Stores info about additional authors.

Properties

Name	Internal name	Searchable	Information collector
Authors	ezauthor	Yes.	No.

Description

This datatype allows the validation, storage and retrieval of additional authors. For each author, it is capable of handling a name and an e-mail address. It is only the e-mail address that will be validated. It is typically useful when there is a need for storing information about additional authors who have written/created different parts of an object's contents. The following screenshot shows the class attribute edit interface for this datatype.

(see figure 5.1)

Figure 5.1: *Class attribute edit interface for the "Authors" datatype.*

As the screenshot indicates, the "Authors" datatype does not have any class specific configuration parameters.

Object attribute edit interface

The following screenshot shows the object attribute edit interface for this datatype.

(see figure 5.2)

When a new object is created, the attribute using this datatype will have its first row set to the name and the e-mail address of the user who created the object.

Figure 5.2: *Object attribute edit interface for the "Authors" datatype.*

Raw output

The ".content" of an ezcontentobjectattribute (page 1042) object using this datatype returns an ezauthor (page 1015) object.

5

5.2.2   Checkbox

Summary

Stores a binary value (on or off).

Properties

Name	Internal name	Searchable	Information collector
Checkbox	ezboolean	Yes.	Yes.

Description

This datatype allows the storage and retrieval of a binary value. It can be either on/true or off/false. The following screenshot shows the class attribute edit interface for this datatype.

(see figure 5.3)

Figure 5.3: *Class attribute edit interface for the "Checkbox" datatype.*

Object attribute edit interface

The following screenshot shows the object attribute edit interface for this datatype.

(see figure 5.4)

Figure 5.4: *Object attribute edit interface for the "Checkbox" datatype.*

5

Default value

The "Default value" parameter makes it possible to control the initial value of an attribute using this datatype when a new object is created.

Raw output

The ".content" of an ezcontentobjectattribute (page 1042) object using this datatype returns either "1" or "0" depending on the state of the checkbox (checked or unchecked).

5

5.2.3   Country

Summary

Stores a user country. [Webshop]

Properties

Name	Internal name	Searchable	Information collector
Country	ezcountry	Yes.	No.

Description

This datatype allows the storage of a user country and thus makes it possible to charge the value added taxes depending on the country the customer is from. This functionality is described in the "VAT charging system (page 375)" section of the "Features" chapter. The following screenshot shows the class attribute edit interface for this datatype.

(see figure 5.5)

Figure 5.5: *Class attribute edit interface for the "Country" datatype.*

Object attribute edit interface

The following screenshot shows the object attribute edit interface for this datatype.

(see figure 5.6)

Raw output

The ".content" of an ezcontentobjectattribute (page 1042) object using this datatype returns a hash that consists of the following elements:

Figure 5.6: *Object attribute edit interface for the "Country" datatype.*

Key	Type	Description
value	string	The actual name of the country (for example "Norway").

5

5.2.4   Date

Summary

Validates and stores a date value.

Properties

Name	Internal name	Searchable	Information collector
Date	ezdate	Yes.	No.

Description

This datatype allows the validation, storage and retrieval of dates consisting of a year, month and day value. The valid input range is 01.01.1970 - 19.01.2038. The following screenshot shows the class attribute edit interface of this datatype.

(see figure 5.7)

Figure 5.7: *Class attribute edit interface for the "Date" datatype.*

Default value

The "Default value" parameter has two options: "Empty" and "Current date". The default setting is "Empty", which means that when a new object is created, the attribute using this datatype will be empty. If the "Current date" setting is used, the current date will be set when a new object is created.

Object attribute edit interface

The following screenshot shows the object attribute edit interface for this datatye.

(see figure 5.8)

5

Figure 5.8: *Object attribute edit interface for the "Date" datatype.*

Raw output

The ".content" of an ezcontentobjectattribute object using this datatype returns an ezdate (page 1062) object.

5

5.2.5 Date and time

Summary

Validates and stores a date and a time value.

Properties

Name	Internal name	Searchable	Information collector
Date and time	ezdatetime	Yes.	No.

Description

This datatype allows the validation, storage and retrieval of a date/time value. It is capable of storing a date/time consisting of a year, month, day, hour and minute value. The following screenshot shows the class attribute edit interface for this datatype.

(see figure 5.9)

Figure 5.9: *Class attribute edit interface for the "Datetime" datatype.*

Default value

The "Default value" parameter can be used to control the initial value of an attribute using this datatype when a new object is created. There are three options:

- Empty

- Current datetime

- Adjusted current datetime

The default setting is "Empty", which means that when a new object is created, the attribute using this datatype will be empty. If the "Current datetime" option is used, the current date and time will be set. If the "Adjusted datetime" is used, an adjusted value of the current date and time will be set. How much the current date and time should be adjusted must be specified using the fields within the "Current datetime adjusted by" group.

Current datetime adjusted by

The fields within this group can be used to specify the desired date/time adjustment when the ("Adjusted current datetime") option is used. Both positive and negative numerical values are allowed. If the values given in the example above are used and an object is created at 00:00 on the first of January 2005, the initial value of the attribute will be set to 04:05, third of February, 2006.

Object attribute edit interface

The following screenshot shows the object attribute interface for this datatype.

(see figure 5.10)

Figure 5.10: *Object attribute edit interface for the "Date and time" datatype.*

Raw output

The ".content" of an ezcontentobjectattribute (page 1042) object using this datatype returns an ezdatetime (page 1063) object.

5.2.6   E-mail

Summary

Validates and stores an E-mail address.

Properties

Name	Internal name	Searchable	Information collector
E-mail	ezemail	Yes.	Yes.

Description

This datatype allows the validation, storage and retrieval of an electronic mail address. The following screenshot shows the class attribute edit interface for this datatype.

(see figure 5.11)

Figure 5.11: *Class attribute edit interface for the "Email" datatype.*

Object attribute edit interface

The following screenshot shows the object attribute edit interface for this datatype.

(see figure 5.12)

Figure 5.12: *Object attribute edit interface for the "E-mail" datatype.*

Raw output

The ".content" of an ezcontentobjectattribute (page 1042) object returns the actual E-mail address.

5.2.7   Enum

Summary

DEPRECATED

Properties

Name	Internal name	Searchable	Information collector
Enum	ezenum	Yes.	No.

Description

This datatype should not be used any more because it is slow. It has been substituted by the
"Selection" (page 588) datatype.

5

5.2.8 File

Summary

Stores any type of file.

Properties

Name	Internal name	Searchable	Information collector
File	ezbinaryfile	Yes.	No.

Description

This datatype allows the storage and retrieval of a single file. It is capable of handling virtually any file type and is typically used for storing legacy document types such as PDF files, Word documents, spreadsheets, etc. The maximum allowed file size is determined by the "Max file size" class attribute edit parameter and the "upload_max_filesize" directive in the main PHP configuration file ("php.ini"). The following screenshot shows the class attribute edit interface for this datatype.

(see figure 5.13)

Figure 5.13: *Class attribute edit interface for the "File" datatype.*

Max file size

The "Max file size" parameter makes it possible to set the highest size (in megabytes) that the system will allow. By default, this parameters is zero, which means that eZ publish will not do any size checking when files are uploaded. In the example above, the "Max file size" parameter is set to 16 MB, which means that the object edit interface will not allow the upload of files that are larger than 16 megabytes. However, if the value of the "upload_max_filesize" PHP setting is lower than 16 megabytes, the underlying system will cancel the upload.

Object attribute edit interface

The following screenshot shows the object attribute edit interface when an attribute using this datatype does not contain any file.

(see figure 5.14)

Figure 5.14: *Object attribute edit interface for the "File" datatype.*

The following screenshot shows the object attribute edit interface when an attribute using this datatype contains a file. The interface reveals the name of the file that was uploaded ("BD-Scratch.wav"), the MIME type ("audio/wav") and the size (1.15 MB).

(see figure 5.15)

Figure 5.15: *Object attribute edit interface for the "File" datatype.*

MIME types

The MIME type will be automatically set based on the extension of the uploaded file's name. If the extension is unknown, the default MIME type will be used ("application/octet-stream"). The MIME types can be configured at the end of the "/lib/ezutils/classes/ezmimetype.php" file.

Storage

The uploaded files 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 and it dramatically decreases the size of the database.

All files uploaded through an attribute that makes use of the file datatype will be stored below "storage/original" within the directory specified by the "VarDir" directive in a configuration override for "site.ini". A new sub-directory will be created for every MIME type. For example, if an executable (.exe) file is uploaded, a directory called "application" will be created; if a text file is uploaded then a directory called "text" will be created, and so on. The uploaded files will be put in the different MIME type directories. Instead of re-using the original file names, eZ publish will create a hash for every file. The following illustration shows the location of two uploaded files (an .exe and a .txt file) when the var directory is set to "my_site".

(see figure 5.16)
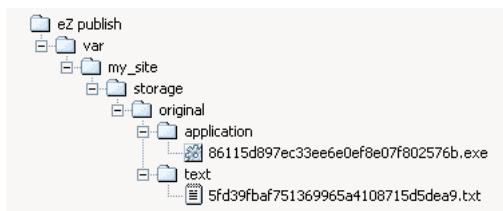
Figure 5.16: *Complete directory structure with uploaded files.*

The system keeps track of the files using a database table called "ezbinaryfile" consisting of the following fields:

Field	Description
content_object_attribute_id	The identification of the content object attribute.
download_count	The number of times the file has been downloaded.
filename	The name of the file on the filesystem (for example "5fd39fbaf751369965a4108715d5dea9.txt").
mime_type	The MIME type of the file (for example "text/plain").
original_filename	The original name of the uploaded file (for example "readme.txt").
version	The version of the object that the file belongs to.

Binary file indexing

eZ Publish is capable of indexing the actual contents of uploaded files. This feature makes it possible to use the built-in search engine to search for something that is inside a file; for example the contents of a PDF file or a spreadsheet. By default, the system is only capable of indexing the

contents of plain text/ASCII files. However, by making use of external programs, it is capable of indexing the contents of virtually any file type (as long as there is a program that goes through the file and returns keywords/contents as plain text). The external handlers can be set up in a configuration override for the "binaryfile.ini" file.

Raw output

The ".content" of an ezcontentobjectattribute (page 1042) object using this datatype returns an ezbinaryfile (page 1019) object.

5

5.2.9   Float

Summary

Validates and stores a decimal value.

Properties

Name	Internal name	Searchable	Information collector
Float	ezfloat	No.	No.

Description

This datatype makes it possible to validate, store and retrieve a single decimal value. It is capable of handling both positive and negative numbers ranging from -3.402823466E+38 to 3.402823466E+38. Please note that these numbers may vary depending on the platform and the database that is used. The following screenshot shows the class edit interface for this datatype.

(see figure 5.17)

Figure 5.17: *Class edit interface for the "Float" datatype.*

5

Default value

The "Default value" parameter makes it possible to set a default decimal value. When the parameter is used and a new object is created, the contents of the attribute using this datatype will be preset to the given value. In the example above, the "Temperature" attribute of new objects will be set to 24.13.

Min integer value

The "Min float value" parameter makes it possible to set the lowest value that the input interface will allow. The default value of this parameter is empty, which means that the system will allow the lowest possible value (-3.402823466E+38). In the example above, the parameter is set to "-40.00". This means that the input interface will not allow the storage of numbers with values less than -40.00.

Max float value

The "Max float value" parameter makes it possible to set the highest value that the input interface will allow. The default value of this parameter is empty, which means that the system will allow the highest possible value (3.402823466E+38). In the example above, the parameter is set to "120.00". This means that the input interface will not allow the storage of numbers with values as high as 120.00.

Object attribute edit interface

The following screenshot shows the object attribute edit interface for this datatype.

(see figure 5.18)

Temperature:
18.50

Figure 5.18: *Object attribute edit interface for the "Float" datatype.*

Raw output

The ".content" of an ezcontentobjectattribute (page 1042) object using this datatype returns a string containing the actual decimal value.

5.2.10   Identifier

Summary

Generates a non-editable identification string.

Properties

Name	Internal name	Searchable	Information collector
Identifier	ezidentifier	Yes.	No.

Description

This datatype allows the automatic generation of unique identification strings for objects. An instance of a class that makes use of this datatype will have a unique identification string generated whenever the object is published for the very first time. The identification string is system-wide and may consist of the following elements:

- User configurable start-text

- Automatically generated identification number (configurable start value and number of digits)

- User configurable end-text

The system will increment the actual counter(s) whenever a new object is published. It will not decrement and reorganize the identification strings when an object is removed. The identification strings are generated and maintained by the system and thus they can not be modified using the object edit interface. The identification strings are generated based on the class type and the identification number of the class attribute. In other words, the identifiers do not depend on the objects' locations within the content node tree. In addition, the datatype may be used to represent several attributes within the same class. The following screenshot shows the class attribute edit interface for this datatype.

(see figure 5.19)

5

Pretext

The "Pretext" parameter can be used to specify a desired start-text (all characters are allowed) that should appear before the automatically generated identification number. In the example above, the text "ABC" is used. This means that the identification strings generated for instances of this class will start with the letters "ABC".

Figure 5.19: *Class attribute edit interface for the "Identifier" datatype.*

Posttext

The "Posttext" parameter can be used to specify a desired end-text (all characters are allowed) that should appear after the automatically generated identification number. In the example above, the text "XYZ" is used. This means that the identification strings generated for instanced of this class will end with the letters "XYZ".

Digits

The "Digits" parameter makes it possible to insert additional zeros in order to generate equally long identification strings. In the example above, the digits parameter is set to "2". This means that numbers below 10 will appear with a prepended zero: "01", "02", "03"..."09". If the parameter was set to "3", the system would generate "001", "002" and so on. The "Digits" parameter can not be used to set the actual range / stop value. It only makes it possible to prepend the actual number with zeros.

Start value

The "Start value" parameter can be used to specify a desired start value for the counter; the default value is zero. In the example above, the value "1" is used. This means that the counter will start at "1". Given all the parameters above, the identification string generated for the very first object will be "ABC01XYZ".

Raw output

The ".content" of an ezcontentobjectattribute (page 1042) object using this datatype returns a string containing the actual identification value.

5.2.11   Image

Summary

Validates and stores a digital image.

Properties

Name	Internal name	Searchable	Information collector
Image	ezimage	No.	No.

Description

This datatype allows the storage of digital images. It is capable of handling virtually any image type. The maximum allowed file size is determined by the "Max file size" class attribute edit parameter and the "upload_max_filesize" directive in the main PHP configuration file ("php.ini"). The following screenshot shows the class attribute edit interface for this datatype.

(see figure 5.20)

Figure 5.20: *Class attribute edit interface for the "Image" datatype.*

Max file size

The "Max file size" parameter makes it possible to set the highest size (in megabytes) that the system will allow. By default, this parameters is zero, which means that eZ Publish will not do any size checking when image files are uploaded. In the example above, the "Max file size" parameter is set to 16 MB, which means that the object edit interface will not allow the upload of images that are larger than 16 megabytes. However, if the value of the "upload_max_filesize" PHP setting is lower than 16 megabytes, the underlying system will cancel the upload.

5

Object attribute edit interface

The following screenshot shows the object attribute edit interface when an attribute using this datatype does not contain an image.

(see figure 5.21)

Figure 5.21: *Object attribute edit interface for the "Image" datatype.*

The following screenshot shows the object attribute edit interface when an attribute using this datatype contains an actual image. The interface reveals the image itself, the name of the file that was uploaded ("dj_cat.jpg"), the MIME type ("image/jpeg") and the size (29.02 kB).

(see figure 5.22)

Figure 5.22: *Object attribute edit interface for the "Image" datatype.*

Storage

The uploaded images are stored on the filesystem. The main reason for this is because the filesystem is much faster than the database when it comes to the storage and retrieval of large data chunks. Having the images on the filesystem allows the webserver 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 the images and it dramatically decreases the size of the database.

All images uploaded through an attribute that makes use of the image datatype will be stored below "storage/images" within the directory specified by the "VarDir" directive in a configuration override for "site.ini". A directory structure is generated for each object that stores an image. The structure will be an exact copy of the actual node path (consisting of the object names) from the root to the main node that references the object which contains images. The following illustration shows the path to a directory on the filesystem in which the different images of an object are stored.

(see figure 5.23)
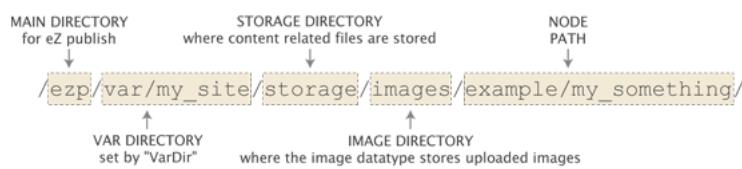
Figure 5.23: *Example of image path on the filesystem.*

In the example above, an image has been stored by an object called "My Something". The main node of this object is a child of a node called "Example". The var directory used by the siteaccess is "var/my_site". The images stored by the "My Something" object will be located in this directory. For each image, version and translation, a new directory will be created. The following illustration shows the naming convention of the directories under which the actual images are stored.

(see figure 5.24)
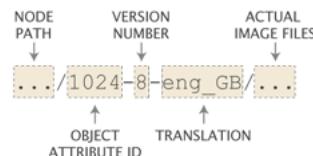
Figure 5.24: *Example of an image subdirectory.*

The example above shows a directory for an image that has been stored by the 1024th object attribute. The image belongs to the British translation for the eight version of the content object. The following illustration shows a complete directory structure with actual image files.

(see figure 5.25)

As the illustration indicates, the image stored by the 1024th attribute (that belongs to the "My Something" object) is represented by multiple files. The original image is "test.png", the rest of the images are the different image variations.

Figure 5.25: *Complete directory structure with uploaded image and generated variations.*

Image variations

With the help of an external application (ImageMagick) or the PHP image library (GD), eZ Publish is capable of generating different variations of an uploaded image. This feature is typically useful when there is a need to show the same image in different ways (for example different sizes). The image variations are controlled by different configuration directives in image.ini. The following table shows the default image variations.

Variation	Width	Height
reference	max 600 pixels	max 600 pixels
small	max 100 pixels	max 100 pixels
medium	max 200 pixels	max 200 pixels
large	max 300 pixels	max 300 pixels
rss	max 88 pixels	max 31 pixels

When an image variation is created (and the default settings are used), eZ Publish will generate a reference image using the file that was uploaded. The reference image is then used to generate the different image variations (small, medium, large and rss) on-demand using the constraints specified in the table above. The generated variations will be cached on the filesystem.

Depending on the image system used, eZ Publish is capable of generating image variations that consist of much more than just scaling. For example, it is possible to generate sharpened grayscale images for the thumbnails of a photo album. Please refer to the documentation of the image.ini configuration file for more information about the different possibilities.

Raw output

The ".content" of an ezcontentobjectattribute (page 1042) object using this datatype returns an ezimagealiashandler (page 1064) object.

5.2.12 Ini setting

Summary

DEPRECATED

Properties

Name	Internal name	Searchable	Information collector
Ini setting	ezinisetting	Yes.	No.

Description

This datatype should no longer be used.

5

5.2.13 Integer

Summary

Validates and stores an integer value.

Properties

Name	Internal name	Searchable	Information collector
Integer	ezinteger	Yes.	No.

Description

This datatype makes it possible to validate, store and retrieve a single integer value. It is capable of handling both positive and negative numbers ranging from -2,147,483,648 to 2,147,483,647. The following screenshot shows the class edit interface for this datatype.

(see figure 5.26)
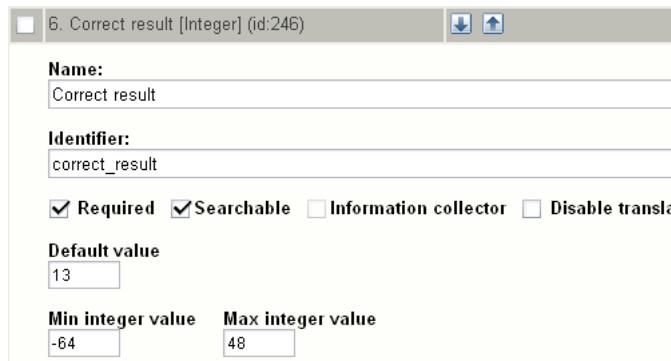
Figure 5.26: *Class edit interface for the "Integer" datatype.*

Default value

The "Default value" parameter makes it possible to set a default integer. When the parameter is used and a new object is created, the contents of the attribute using this datatype will be preset to the given value. In the example above, the "Correct result" attribute of new objects will be set to 13.

Min integer value

The "Min integer value" parameter makes it possible to set the lowest value that the input interface will allow. The default value of this parameter is empty, which means that the system will

allow the lowest possible value, which is -2,147,483,648. In the example above, the parameter is set to "-64". This means that the input interface will not allow the storage of numbers with values less than -64.

Max integer value

The "Max integer value" parameter makes it possible to set the highest value that the input interface will allow. The default value of this parameter is empty, which means that the system will allow the highest possible value, which is 2,147,483,647. In the example above, the parameter is set to "48". This means that the input interface will not allow the storage of numbers with values as high as 48.

Object attribute edit interface

The following screenshot shows the object attribute interface for this datatype.

(see figure 5.27)

Figure 5.27: *Object attribute edit interface for the "Integer" datatype.*

Raw output

The ".content" of an ezcontentobjectattribute (page 1042) object using this datatype returns a string containing the actual integer.

5.2.14 ISBN

Summary

Validates and stores an ISBN value.

Properties

Name	Internal name	Searchable	Information collector
ISBN	ezisbn	Yes.	No.

Description

The "ISBN" datatype allows the validation, storage and retrieval of a single ISBN (International Standard Book Number) code. From version 3.10 of eZ Publish, this datatype uses the new ISBN-13 format by default.

Note that from the 1st of January, 2007, the book industry started using 13 digit ISBN codes (ISBN-13) to identify all books in their supply chains.

According to the ISBN-13 standard, the 13-digit number consists of five parts. The parts are separated by dashes. The following table describes each of the five parts using the ISBN-13 code "978-0-14026-690-0" as an example.

#	Name	Description	Example
1	Prefix	A three digit prefix that can be either 978 or 979. Note that the new 979 prefix will not be utilized until all 978 prefixes have been assigned.	978
2	Registration group	A variable set of digits that identify the country/region in which the ISBN code was assigned. Zero ("0") means that the book belongs to an English speaking area.	0
3	Registrant	A variable set of digits that identify the publisher to whom the ISBN code was	14026

			originally allocated.	
4		Publication	A variable set of digits that identify the title of the book.	690
5		Check digit	A one digit control number that makes it possible to validate the ISBN code. The check digit is calculated using a formula that makes use of the preceding parts (Prefix, Registration group, Registrant and Publication).	0

While the "Prefix" and "Check digit" elements have a fixed length (3 and 1 digits accordingly), the length of the remaining elements may vary. However, the total length of an ISBN-13 is always 13 digits. The length of the "Registration group", "Registrant" and "Publication" elements can be calculated according to the rules set by the International ISBN Agency. The calculation of these elements is based on the distribution of ranges for the different regions.

In eZ Publish, the ISBN codes are stored as text. The digits are separated by dashes. The different ranges are stored in the database, allowing the system to fix up incorrectly entered ISBN-13 codes (codes with misplaced or missing dashes). When a new version of eZ Publish is released, the latest information about the different ranges is downloaded from http://www.isbn-international.org and stored in the "db_data.dba" file located in the "kernel/classes/datatypes/ ezisbn/share/" directory. During the installation process, the setup wizard imports this information to the database. If the information about ranges is missing from the database, the system will display a warning at the class attribute edit interface for this datatype (see below).

Note that the calculation rules can be changed on a monthly basis and thus you should run the "updateisbn13.php" script regularly in order to keep the range-information in the database up to date. The following example shows how to get the latest ranges from the ISBN International website:

1. Download a ".js" file containing the list of ranges using the following link: http://www.isbn-international.org/converter/ranges.js

2. Navigate into the eZ Publish directory and copy the downloaded file there.

3. Run the "updateisbn13.php" script using the following shell command:

```
php bin/php/updateisbn13.php --file=ranges.js
```

The script will import information from the "range.js" file to the database.

The following screenshot shows the class attribute edit interface for this datatype.

5

(see figure 5.28)

Figure 5.28: *Class attribute edit interface for the "ISBN" datatype.*

The "ISBN-13 format" checkbox makes it possible to configure the datatype so that it validates and stores ISBN codes according to the ISBN-13 standard. From version 3.10, this checkbox is checked by default when a new attribute using the ISBN datatype is created. Note that the object edit interface will appear differently (four input fields instead of one) when the checkbox is unchecked (see below).

The following screenshot shows how the class attribute edit interface for the ISBN datatype looks like if the information about ranges is missing from the database.

(see figure 5.29)
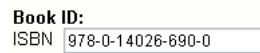
Figure 5.29: *Class attribute edit interface for the "ISBN" datatype if the range data for ISBN-13 was not imported.*

Click the "Import ISBN range data" button to import ranges to the database.

Object attribute edit interface

The following screenshot shows the default object attribute edit interface for this datatype.

(see figure 5.30)

Figure 5.30: *Object attribute edit interface for the "ISBN" datatype.*

As the screenshot shows, an ISBN-13 code consists of five parts separated by dashes.

The system will automatically add dashes in case they were omitted during input (for example, "9780140266900" will be converted to "978-0-14026-690-0"). The system will also automatically fix incorrectly placed dashes (for example, "978-0-140266-90-0" will be converted to "978-0-14026-690-0"). If an ISBN-10 code is entered, the system will automatically convert it into a valid ISBN-13 code unless the "ISBN-13 format" checkbox is disabled at the class level. Note that there is no support for automatic conversion of ISBN-13 codes to ISBN-10 codes since the ISBN-10 format is considered to be obsolete.

In eZ Publish 3.10, a feature for checking the validity of ISBN codes has been introduced. By default, the system will check all elements of an ISBN-13 code and display a warning if one or more of the entered elements are invalid. For example, "978-0-14026-690-6" is considered to be an invalid code; according to the standard, the last digit should have been a zero ("0"). The validity checking feature can be turned on (default) or off. It is controlled by the "StrictValidation (page 1740)" setting located in the [ISBNSettings] section of the "settings/content.ini" configuration file. Note that the "Prefix" and "Check digit" parts will be validated regardless if the "StrictValidation" setting is enabled or not.

The following screenshot shows the object attribute edit interface for this datatype when ISBN-13 support is disabled.

(see figure 5.31)

Figure 5.31: *Object attribute edit interface for the "ISBN" datatype when in ISBN-10 mode.*

Raw output

The ".content" of an ezcontentobjectattribute (page 1042) object using this datatype returns a hash containing eight elements. The hash-structure of the elements is given in the table below.

Key	Type	Description
prefix	string	The "Prefix" part of the ISBN

		code (for example 978).
field1	string	The "Registration group" part of the ISBN code (for example 0).
field2	string	The "Registrant" part of the ISBN code (for example 14026).
field3	string	The "Publication group" part of the ISBN code (for example 690).
field4	string	The "Check digit" part of the ISBN code (for example 0).
value	string	The entire ISBN code with dashes (for example 978-0-14026-690-0).
value_without_hyphens	string	The entire ISBN code without dashes (for example 9780140266900).
value_with_spaces	string	The entire ISBN code with spaces (for example 978 0 14026 690 0).

5

5.2.15   ISBN (as of 4.5)

Summary

Validates and stores an ISBN value.

Properties

Name	Internal name	Searchable	Information collector
ISBN (as of 4.5)	ezisbn	Yes.	No.

Description

The "ISBN" datatype allows the validation, storage and retrieval of a single ISBN (International Standard Book Number) code. From version 3.10 of eZ Publish, this datatype uses the new ISBN-13 format by default.

Note that from the 1st of January, 2007, the book industry started using 13 digit ISBN codes (ISBN-13) to identify all books in their supply chains.

According to the ISBN-13 standard, the 13-digit number consists of five parts. The parts are separated by dashes. The following table describes each of the five parts using the ISBN-13 code "978-0-14026-690-0" as an example.

#	Name	Description	Example
1	Prefix	A three digit prefix that can be either 978 or 979. Note that the new 979 prefix will not be utilized until all 978 prefixes have been assigned.	978
2	Registration group	A variable set of digits that identify the country/region in which the ISBN code was assigned. Zero ("0") means that the book belongs to an English speaking area.	0
3	Registrant	A variable set of digits that identify the publisher to whom the ISBN code was	14026

		originally allocated.	
4	Publication	A variable set of digits that identify the title of the book.	690
5	Check digit	A one digit control number that makes it possible to validate the ISBN code. The check digit is calculated using a formula that makes use of the preceding parts (Prefix, Registration group, Registrant and Publication).	0

While the "Prefix" and "Check digit" elements have a fixed length (3 and 1 digits accordingly), the length of the remaining elements may vary. However, the total length of an ISBN-13 is always 13 digits. The length of the "Registration group", "Registrant" and "Publication" elements can be calculated according to the rules set by the International ISBN Agency. The calculation of these elements is based on the distribution of ranges for the different regions.

In eZ Publish, the ISBN codes are stored as text. The digits are separated by dashes. The different ranges are stored in the database, allowing the system to fix up incorrectly entered ISBN-13 codes (codes with misplaced or missing dashes). When a new version of eZ Publish is released, the latest information about the different ranges is downloaded from http://www.isbn-international.org and stored in the "db_data.dba" file located in the "kernel/classes/datatypes/ ezisbn/share/" directory. During the installation process, the setup wizard imports this information to the database. If the information about ranges is missing from the database, the system will display a warning at the class attribute edit interface for this datatype (see below).

Note that the calculation rules can be changed on a monthly basis and thus you should run the "updateisbn13.php" script regularly in order to keep the range-information in the database up to date. The following example shows how to get the latest ranges from the ISBN International website:

Run the "updateisbn13.php" script using the following shell command:

```
php bin/php/updateisbn13.php --file=ranges.js
```

The script will import information from the isbn international website to the database.

The following screenshot shows the class attribute edit interface for this datatype.

(see figure 5.32)

The "ISBN-13 format" checkbox makes it possible to configure the datatype so that it validates and stores ISBN codes according to the ISBN-13 standard. From version 3.10, this checkbox is checked by default when a new attribute using the ISBN datatype is created. Note that the

Figure 5.32: *Class attribute edit interface for the "ISBN" datatype.*

object edit interface will appear differently (four input fields instead of one) when the checkbox is unchecked (see below).

The following screenshot shows how the class attribute edit interface for the ISBN datatype looks like if the information about ranges is missing from the database.

(see figure 5.33)

Figure 5.33: *Class attribute edit interface for the "ISBN" datatype if the range data for ISBN-13 was not imported.*

Click the "Import ISBN range data" button to import ranges to the database.

Object attribute edit interface

The following screenshot shows the default object attribute edit interface for this datatype.

(see figure 5.34)

As the screenshot shows, an ISBN-13 code consists of five parts separated by dashes.

Book ID:
ISBN 978-0-14026-690-0

Figure 5.34: *Object attribute edit interface for the "ISBN" datatype.*

The system will automatically add dashes in case they were omitted during input (for example, "9780140266900" will be converted to "978-0-14026-690-0"). The system will also automatically fix incorrectly placed dashes (for example, "978-0-140266-90-0" will be converted to "978-0-14026-690-0"). If an ISBN-10 code is entered, the system will automatically convert it into a valid ISBN-13 code unless the "ISBN-13 format" check-box is disabled at the class level. Note that there is no support for automatic conversion of ISBN-13 codes to ISBN-10 codes since the ISBN-10 format is considered to be obsolete.

In eZ Publish 3.10, a feature for checking the validity of ISBN codes has been introduced. By default, the system will check all elements of an ISBN-13 code and display a warning if one or more of the entered elements are invalid. For example, "978-0-14026-690-6" is considered to be an invalid code; according to the standard, the last digit should have been a zero ("0"). The validity checking feature can be turned on (default) or off. It is controlled by the "StrictValidation (page 1740)" setting located in the [ISBNSettings] section of the "settings/content.ini" configuration file. Note that the "Prefix" and "Check digit" parts will be validated regardless if the "StrictValidation" setting is enabled or not.

The following screenshot shows the object attribute edit interface for this datatype when ISBN-13 support is disabled.

(see figure 5.35)

Book ID:
ISBN 0 - 306 - 40615 - 2

Figure 5.35: *Object attribute edit interface for the "ISBN" datatype when in ISBN-10 mode.*

Raw output

The ".content" of an ezcontentobjectattribute (page 1042) object using this datatype returns a hash containing eight elements. The hash-structure of the elements is given in the table below.

Key	Type	Description
prefix	string	The "Prefix" part of the ISBN code (for example 978).
field1	string	The "Registration group" part of the ISBN code (for example 0).
field2	string	The "Registrant" part of the ISBN code

5

		(for example 14026).
field3	string	The "Publication group" part of the ISBN code (for example 690).
field4	string	The "Check digit" part of the ISBN code (for example 0).
value	string	The entire ISBN code with dashes (for example 978-0-14026-690-0).
value_without_hyphens	string	The entire ISBN code without dashes (for example 9780140266900).
value_with_spaces	string	The entire ISBN code with spaces (for example 978 0 14026 690 0).

5

5.2.16   Keywords

Summary

Stores keywords.

Properties

Name	Internal name	Searchable	Information collector
Keywords	keywords	Yes.	No.

Description

This datatype allows the storage of keywords for an object. The keywords must be specified as a comma separated list of words and/or phrases. This datatype can be used to connect objects of the same type based on their keywords. It is typically useful when it comes to creating interfaces that allow the user to quickly browse other pages with related content (for example "see also" or "related pages" list of hyperlinks). If two objects share at least one common keyword, the objects will be connected behind the scenes. The system will create the necessary entries in the "ezkeyword" and the "ezkeyword_attribute_link" database tables. Please note that this datatype does not generate object relations, it simply uses its own tables in the database to connect the objects. The following screenshot shows the class attribute edit interface for this datatype.

(see figure 5.36)

Figure 5.36: *Class attribute edit interface for the "Keywords" datatype.*

Object attribute edit interface

The following screenshot shows the object attribute edit interface for this datatype.

(see figure 5.37)

Figure 5.37: *Object attribute edit interface for the "Keywords" datatype.*

Raw output

The ".content" of an ezcontentobjectattribute (page 1042) object using this datatype returns an ezkeyword (page 1075) object.

5

5.2.17 Matrix

Summary

Stores multiple rows and columns of text.

Properties

Name	Internal name	Searchable	Information collector
Matrix	ezmatrix	Yes.	No.

Description

This datatype allows the storage and retrieval of information structured in a table. The following screenshot shows the class attribute edit interface for this datatype.

(see figure 5.38)

Figure 5.38: *Class attribute edit interface for the "Matrix" datatype.*

Object attribute edit interface

The following screenshot shows the object attribute edit interface for this datatype.

(see figure 5.39)

Figure 5.39: *Object attribute edit interface for the "Matrix" datatype.*

Raw output

The ".content" of an ezcontentobjectattribute (page 1042) object using this datatype returns an ezmatrix (page 1080) object.

5

5.2.18 Media

Summary

Stores a media file (Flash/QT/Real/etc.).

Properties

Name	Internal name	Searchable	Information collector
Media	ezmedia	No.	No.

Description

This datatype allows the storage and playback of a video file. It is capable of handling Apple QuickTime, Macromedia Flash, Microsoft Windows Media, Real Media, Silverlight*, HTML5 Video** and HTML Audio** files. The maximum allowed file size is determined by the "Max file size" class attribute edit parameter and the "upload_max_filesize" directive in the main PHP configuration file ("php.ini").

*available from 4.1 onwards
 **available from 4.4 onwards

The following screenshot shows the class attribute edit interface for this datatype.

(see figure 5.40)

Figure 5.40: *Class attribute edit interface for the "Media" datatype.*

5

Media player type

The "Media player type" parameter controls the way the video file will be played back on the client side. The following options are available:

- Flash

- QuickTime

- RealPlayer

- Windows media player

- Silverlight - available from 4.1 onwards

- HTML5 Video - available from 4.4 onwards

- HTML5 Audio - available from 4.4 onwards

Regarding Html5 video and Html5 audio: As the name implies they use the native video and audio tag as added in html5, this means that as of the time of writing it's up to the browser to support the video/audio format you upload. For instance for video Apple Safari (incl IPhone & IPad) supports mpeg4(h.264) and Mozilla Firefox supports ogv (Theora). For an up-to-date list of supported formats in different browsers, see more info on html5 video and audio.

Media players and supported formats

In the table below you will find the media players supported by eZ Publish along with the supported formats out-of-the-box:

Media player	Formats
Flash	**.swf**, **.flv**, **.avi**
QuickTime	**.mov**, **.avi**, **.mp4**
RealPlayer	**.rm**
Windows Media Player	**.wmv**, **.avi**,** .mpeg**,** .mp4**
Silverlight	**.xap**
HTML5 Audio/Video	**.mp4**, **.m4v**, **.mov**, **.flv**, **.ogv**, **.avi**, **.webm**

Find more information on the pre-defined media content classes under "Media (page 635)".

Max file size

The "Max file size" parameter makes it possible to set the highest size (in megabytes) that the system will allow. By default, this parameters is zero, which means that eZ Publish will not do any size checking when files are uploaded. In the example above, the "Max file size" parameter is set to 32 MB, which means that the object edit interface will not allow the upload of files that are larger than 32 megabytes. However, if the value of the "upload max filesize" PHP setting is lower than 32 megabytes, the underlying system will cancel the upload.

Object attribute edit interface

The object attribute edit interface for the "Media" datatype depends on the selected "Media player type" (in class edit mode). The following screenshots show the different object attribute edit interfaces for the supported media types.

5

(see figure 5.41)

Figure 5.41: *Object attribute edit interface for the "Media" datatype (Flash).*

(see figure 5.42)

Figure 5.42: *Object attribute edit interface for the "Media" datatype (QuickTime).*

(see figure 5.43)

(see figure 5.44)

Storage

Files that have been uploaded through the "Media" datatype are stored in the same way as files that are uploaded using the "File" (page 528) datatype. However, the system keeps track of the media files using the "ezmedia" table instead of the "ezbinaryfile" table.
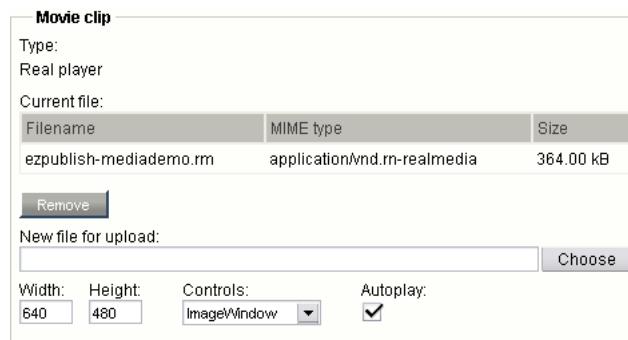
Figure 5.43: *Object attribute edit interface for the "Media" datatype (Real Media)*.

Figure 5.44: *Object attribute edit interface for the "Media" datatype (Windows media)*.

Raw output

The ".content" of an ezcontentobjectattribute (page 1042) object using this datatype returns an ezmedia (page 1083) object.

5.2.19   Multi-option

Summary

DEPRECATED

Properties

Name	Internal name	Searchable	Information collector
Multi-option	ezmultioption	Yes.	No.

Description

This datatype should no longer be used. It was deprecated in eZ Publish 3.10 and is replaced by the "Multi-option2 (page 563)" datatype.

This 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" (page 579) 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. The following screenshot shows the class attribute edit interface for this datatype.

(see figure 5.45)

Figure 5.45: *Class attribute edit interface for the "Multi-option" datatype.*

Object attribute edit interface

The following screenshot shows the object attribute edit interface for this datatype.

(see figure 5.46)

Raw output

The ".content" of an ezcontentobjectattribute (page 1042) object using this datatype returns an ezmultioption (page 1085) object.

Figure 5.46: *Object attribute edit interface for the "Multi-option" datatype.*

5

5.2.20   Multi-option2

Summary

Allows option selections (supports enabling/disabling of options, images, nested option sets and dependencies). [Webshop]

Properties

Name	Internal name	Searchable	Information collector
Multi-option2	ezmultioption2	Yes.	No.

Description

Note: "This datatype is a replacement for the deprecated "Multi-option" datatype. Refer to the 3.10 upgrading manuals for information about how to convert the data.

This datatype makes it possible to create multiple and distinctive groups of multi-options for each content object. The multi-options can be nested (there is a limit, see below). 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 possible to set up rules for allowing/disallowing certain combinations of options.

All configuration is done at the object level. This provides great flexibility. Every object using the "Multi-option2" datatype can be different since there is no class-level configuration to be followed. In other words, if you need to represent different types of products, you can just create a simple and generic "Product" class that uses this datatype. The products (objects) themselves can be of completely different types with different traits (options). For example, you can create an object representing a computer with different options (memory, processor, etc.). At the same time, you can create another object that represents a couch; again with different options (color, finish, size, softness, etc.) and so on. All options, additional prices, rules / dependencies and so on can be defined at the object level.

The capabilities and practical usage of this datatype will be explained using an example of cars being sold through the webshop. Note that there will only be one car object. However, the options, rules and so on will make it possible to sell different versions (custom configurations) of the product. The price of the car will vary depending on the selected options. Note that some options will be default while others must be selected (the user will not be allowed to continue until a selection is made). In addition, some combination of options will not be allowed. Note that this particular feature requires JavaScript support in the browser because the validation will be done on-the-fly while the user is tweaking the options. However, there is a fallback mechanism for non-JavaScript clients. It will take care of the validation when the product is added to the basket. In this case, the product will not be added to the basket as long as an invalid set of selections is provided.

The product (car) in this example will have three different traits: body, color and seats. We will use three multi-options called "Body", "Color" and "Seats" to represent the possible combinations/variations.

The following table shows the available options for this example.

Body	Color	Seats
• Sedan • Combi • Hatchback	• Black • Red	• Cloth • Leather

Each option can be assigned a short text and an additional price. It is also possible to add images for options. In the example above, images can be used to demonstrate the colors that the vehicle is available in and/or show how the different bodies look like. An option can be disabled. Disabling an option will make it non-selectable when the object is viewed. For each multi-option, it is possible to specify a default selection. If a disabled option is chosen as the default selection, users will be forced to select one of the enabled options instead. The following table shows how the default and disabled options are set for the example described above.

Body	Color	Seats
• Sedan (default) • Combi • Hatchback	• Black • Red (default)	• Make your choice (default, disabled) • Cloth • Leather

As the table shows, the default choice for body style is "Sedan", the default color is red. The "Seats" multi-option contains a new option called "Make your choice", which will be selected by default. Since this option is disabled, the user will have to select either "Cloth" or "Leather" in order to buy a car ("Make your choice" will be treated as an invalid selection).

This datatype supports hierarchical organization of the multi-options for each content object. This means that it is possible to create new multi-options beneath existing ones, thus allowing the site administrator to build complex multi-level structures. In the car example, two new multi-options called "Bumpers" and "Mirrors" added beneath the "Color" multi-option could represent the available colors of bumpers and mirrors (the "Color" multi-option represents available body colors). The following table shows the resulting two-level structure.

5

Body	Color	Seats	
• Sedan • Combi • Hatchback	• Black • Red	• *Make your choice* • Cloth • Leather	
	Bumpers	**Mirrors**	
	• Same as body • Non-painted	• Silver • Same as body	

		• Non-painted	

It is possible to specify dependency rules for options located in nested groups. Let's say that for example red cars can not have their bumpers painted in the same color as the body. This situation can be described using the following table where each column corresponds to one of the options of the outer group ("Color"), while the options of the nested group ("Bumpers") are represented by the rows.

Bumpers	**Color of the body**	
Black	Red	
Same as body	Available	Not available
Non-painted	Available	Available

In a similar way, the following table demonstrates the dependency rules between the "Color" and "Mirrors" groups of options.

Mirrors	**Color of the body**	
Black	Red	
Silver	Not available	Available
Same as body	Available	Available
Non-painted	Available	Not available

As the table shows, silver mirrors are not available for black cars and red cars can not have non-painted mirrors. If the dependency rules are specified as shown in the tables above, the system will make the specified options of the inner groups disabled depending on the options that are selected in the outer group. This is achieved by making use of JavaScript in the resulting HTML and thus the client browser must support JavaScript for it to work. If JavaScript is not supported, the interface will allow invalid selections while the visitor/customer plays around with the different options. However, the product will not be added to the basket as long as an invalid set of selections is provided. As previously mentioned, the combination of options will be validated when the user attempts to add the object to the shopping basket.

This datatype supports grouping of multi-options. A multi-option located at the top level of the structure always belongs to a group. Multi-options located beneath it (second level, third level and so on) will be considered to be in the same group. The following table shows the situation with two groups called "Exterior" and "Interior".

EXTERIOR	**INTERIOR**		
Body	**Color**	**Seats**	
• Sedan • Combi • Hatchback	• Black • Red	• *Make your choice* • Cloth • Leather	
	Bumpers	**Mirrors**	
	• Same as body • Non-painted	• Silver • Same as body • Non-painted	

In the example above, while the "Body" and "Color" multi-options belong to the "Exterior" group, the "Seats" multi-option belongs to another group called "Interior". In other words, the multi-options located at the top level are grouped. The "Bumpers" and "Mirrors" multi-options located at the second level belong to the same group as "Color".

Note that both grouping and nesting of multi-options are defined at the time of creation. It is not possible to move an existing multi-option to another group or place it beneath a different multi-option. This is why the multi-level structure must be carefully thought out beforehand. It is not recommended to create structures containing more than ten levels. The maximum number of levels is limited to 15. Refer to the source code if you need to change this limitation.

Removal of a multi-option will also remove all its sub items (multi-options located beneath it). By removing all multi-options that belong to a group, you can remove the the group itself. Note that this is the only way of removing a group.

The following screenshot shows the class attribute edit interface for this datatype.

(see figure 5.47)

Figure 5.47: *Class attribute edit interface for the "Multi-option2" datatype.*

Object attribute edit interface

The following screenshot shows the object attribute edit interface for this datatype.

(see figure 5.48)

As you can see from the screenshot above, options are represented as lines in the table (one table for each multi-option). The "Option" field contains the name of the option. The value specified in the "Additional price" column will be added to the product price in case the corresponding option is selected. The "Image" column can be used to add images to options. When viewing a product, users will see a thumbnail/icon that links to the original image. In this case, the system will display options as a list with radio buttons (if no images are added, the options are shown using a drop-down list). Note that it is possible to add other types of related objects instead of images, but the default templates are for images only.

While the "Def" column with radio buttons can be used to specify the default selection, the "Dis" column contains checkboxes for disabling the options. The "Add option" button can be used to add a new empty line to the table. To remove one or more options, select them using the checkboxes located in the first unnamed column and click the "Remove selected" button.
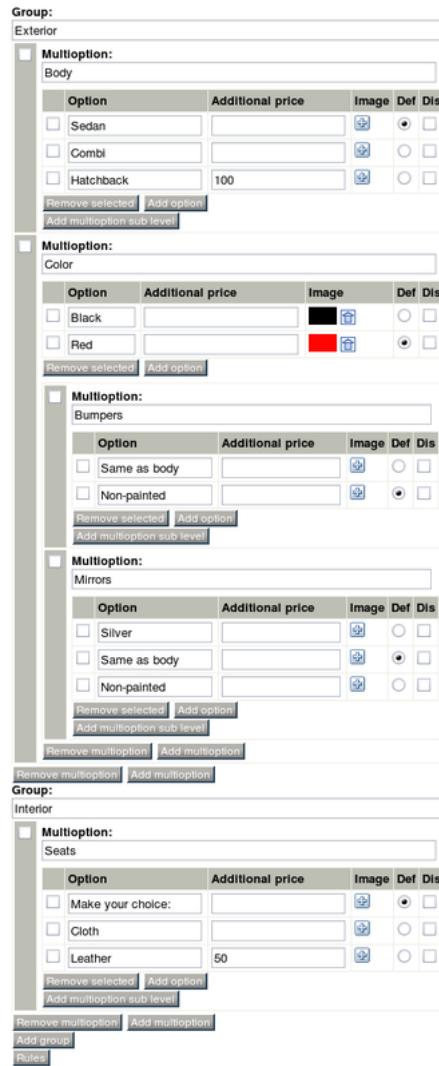
Figure 5.48: *Object attribute edit interface for the "Multi-option2" datatype.*

To add a new multi-option beneath an existing one, click the "Add multioption sub level" button. Note that this button will not be shown for multi-options that already have sub items (multi-options located beneath them). In this case, the system will show a list of the sub items with checkboxes on the left side plus two new buttons called "Remove multioption" and "Add multi-option" below. These buttons are present at each level of the structure, allowing you to add and remove multi-options located at the different levels. Note that removing a multi-option will also remove all its sub items.

Adding a new group of multi-options

The "Add group" button makes it possible to add a new group of multi-options. The following screenshot shows a part of the object attribute edit interface that corresponds to a newly created

empty group of multi-options.

(see figure 5.49)

Figure 5.49: *A part of the object attribute edit interface for the "Multi-option2" datatype.*

As the screenshot shows, a newly created group consists of one multi-option containing only one option (all the fields are empty). At this stage, you should enter the name of the group and the name of multi-option, plus create the necessary options. Nested multi-options can be created using the "Add multioption sub level" button.

The "Add multioption" button allows you to add new multi-options to the top level within the group. The system will show all multi-options that belong to the group with checkboxes on the left hand side. The "Remove multioption" button can be used to remove selected multi-options. The group itself is automatically removed when all multi-options that belong to it are removed.

Adding dependency rules

The "Rules" button located at the bottom makes it possible to specify dependency rules for nested multi-options. When this button is clicked, the object attribute edit interface will change. Instead of showing an interface for editing the options, the system will show an interface that can be used to define the rules. Note that this behavior isn't common when it comes to other datatypes. The following screenshot shows what the object attribute edit interface will change into when the "Rules" button is clicked.

(see figure 5.50)

As the screenshot shows, every nested multi-option ("Bumpers" and "Mirrors") has a table with checkboxes that make it possible to specify which options are available for each possible value of the parent multi-option ("Color"). By default, all checkboxes are selected, meaning that all options are available. The screenshot above shows a situation which follows the rules that were discussed earlier. In other words, the following limitations are introduced:

- Red cars can not have bumpers painted in the same color as the body.

- Black cars can not have silver mirrors.

- Red cars can not have non-painted mirrors.

Figure 5.50: *Dependency rules in the object attribute edit interface for the "Multi-option2" datatype.*

The "Reset rules" button located at the bottom of the interface makes it possible to restore the default configuration (allowing all possible combinations). Note that pressing this button will wipe out the custom rules. Clicking the "Options" button will bring you back to the interface for editing options. In other words, the "Rules" and the "Options" buttons make it possible to switch between the two object edit interfaces of this datatype.

Raw output

The ".content" of an ezcontentobjectattribute (page 1042) object using this datatype returns an ezmultioption2 (page 1087) object.

5.2.21   Multi-price

Summary

Stores prices in different currencies (inc/ex VAT). [Webshop]

Properties

Name	Internal name	Searchable	Information collector
Multi-price	ezmultiprice	Yes.	No.

Description

This datatype allows the storage of prices in multiple currencies and thus makes it possible to connect content objects with the e-commerce subsystem. The e-commerce features of eZ Publish are described in the "Webshop (page 162)" section of the "Concepts and basics" chapter and in the "Multi-currency (page 417)" section of the "Features" chapter. The following screenshot shows the class attribute edit interface for this datatype.

(see figure 5.51)

Figure 5.51: *Class attribute edit interface for the "Multi-price" datatype.*

Parameters and usage

This is a special datatype which plugs more deeply into the system. Instances of any class containing the multi-price datatype will automatically be treated as products. This datatype works with VATs in the same way as the "Price (page 582)" datatype. (Please note that price and multi-price datatypes are incompatible. A content class can only contain one price attribute or one multi-price attribute.)

It is possible to set one of the predefined currencies as "default currency" on the class level so that the system will create a custom price (page 418) in this currency and auto prices (page 418) in all other currencies when a new object is created. Note that a custom price can be always modified or removed for each individual product / object.

Object attribute edit interface

The following screenshot shows the object attribute edit interface for this datatype.

(see figure 5.52)

Figure 5.52: *Object attribute edit interface for the "Multi-price" datatype.*

Raw output

The ".content" of an ezcontentobjectattribute (page 1042) object using this datatype returns an ezmultiprice (page 1092) object.

5

5.2.22   Object relation

Summary

Stores a relation to a content object.

Properties

Name	Internal name	Searchable	Information collector
Object relation	ezobjectrelation	Yes.	No.

Description

This datatype allows the relation of a single object. The following screenshot shows the class attribute edit interface for this datatype.

(see figure 5.53)

Figure 5.53: *Class attribute edit interface for the "Object relation" datatype.*

Object attribute edit interface

The following screenshot shows the object attribute edit interface for this datatype.

(see figure 5.54)

Figure 5.54: *Object attribute edit interface for the "Object relation" datatype.*

Raw output

The ".content" of an ezcontentobjectattribute (page 1042) object using this datatype returns either FALSE (if there is no relation) or an ezcontentobject (page 1032) object.

5.2.23 Object relations

Summary

Stores relations to other content objects.

Properties

Name	Internal name	Searchable	Information collector
Object relations	ezobjectrelationlist	Yes.	No.

Description

This datatype makes it possible to store relations to multiple objects. It allows users to browse the node tree and select objects that should be related; in addition, it makes it possible to create new objects and automatically relate them to the one that is being edited. The following screenshot shows the class attribute edit interface for this datatype.

(see figure 5.55)

Figure 5.55: *Class attribute edit interface for the "Object relations" datatype.*

The "Selection method" dropdown list makes it possible to set which browse interface should be

used when the node tree is being explored by users (from within the object edit interface). There are seven selection methods in total. Three of them allow the selection of one object at a time. The remaining four make it possible to select multiple objects at the same time.

The following selection methods only allow the addition of one related object at a time:

- Dropdown list

- List with radio buttons

- Template based, single

The following selection methods allow the addition of multiple related objects at the same time:

- Browse (default)

- List with check boxes

- Multiple selection list

- Template based, multi

By default, the "Browse" selection method is used, which means that the system will bring up a standard browse interface for adding related objects. It is possible to specify the start-location for browsing. This is controlled by the "Default location" section. If any other selection method is used, the system will display a list of nodes as part of the object edit interface. In these cases, the list is automatically limited to one subtree, according to the specified default location. The template based selection methods ("Template based, multi" and "Template based, single") make use of the "objectrelationlist.tpl" template located in the "templates/node/view/" directory of the standard design. By creating a template override, you can customize the look and feel for these selection methods.

Note that if a related object is moved to the trash, the relation will still exist. The related object itself will not be shown when the object which it is related to is displayed. This behavior can be changed by overriding the default templates. If the "Browse" selection method is used in the object edit interface, the "Published" column of the corresponding related object will read "No".

The "Type" dropdown list is no longer in use. It is only available if the backwards compatibility mode is enabled. This is controlled by the "AdvancedObjectRelationList" setting located in the "[BackwardCompatibilitySettings]" section of the "settings/site.ini" configuration file or an override. Due to some problems with the old implementation, this setting is disabled by default. This documentation page provides the description of the datatype assuming that the "AdvancedObjectRelationList" setting is disabled.

The "Allowed classes" menu makes it possible to control which types of objects that users should be allowed to select while browsing the node tree.

In addition to allowing relations to existing objects, this datatype makes it possible to create new objects and automatically relate them to the one that is being edited. Note that this currently works with all selection methods except the "Browse" method. In order to use this feature, you

5

need to specify which type of object that should be created and where the newly created objects should be placed. This can be done by making use of the "Object class" dropdown list in the "New objects" section and the "Default location" section. If the "Object class" dropdown is set to "none", it will not be possible to create new objects from within the object edit interface.

As previously mentioned, the "Default location" section can be used to limit the list of nodes to the specified subtree (for any selection method except "Browse"). For example, if you use the "Media" folder as the default location and specify "List with check boxes" as the selection method in the class edit interface, the system will display nodes which are located under the "Media" node (only the ones which are directly below it) along with checkboxes for making selections right in the object edit interface. If no default location is specified, the list of nodes will only contain the top level nodes.

Object attribute edit interface

The look and feel of the object attribute edit interface for this datatype varies depending on which selection method that is specified at the class level. Note that the "Create new" button will not be available unless a default location and a class for newly created objects are specified.

The following screenshot shows the object attribute edit interface for this datatype in a case where the "Browse" selection method is used.

(see figure 5.56)

Figure 5.56: *Object attribute edit interface for the "Object relations" datatype (default selection method)*.

The following screenshot shows the object attribute edit interface in a case where the "List with check boxes" selection method is used while both the default location (in this case the "Media" folder) and the type of newly created objects are specified.

(see figure 5.57)

The following screenshot shows the object attribute edit interface in a case where the "Multiple selection list" selection method is used and the "Media" folder is specified as the default location.

(see figure 5.58)

The following screenshot shows the object attribute edit interface in a case where the "Dropdown list" selection method is used and the "Media" folder is specified as the default location.

(see figure 5.59)

5

Figure 5.57: *Object attribute edit interface for the "Object relations" datatype (list with checkboxes).*
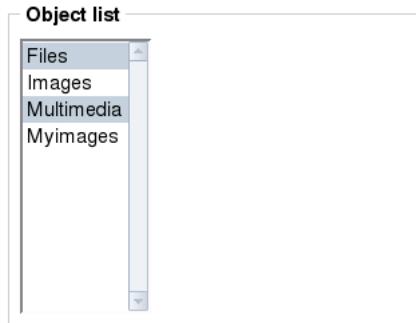
Figure 5.58: *Object attribute edit interface for the "Object relations" datatype (multiple selection list).*

Figure 5.59: *Object attribute edit interface for the "Object relations" datatype (dropdown list).*

The following screenshot shows the object attribute edit interface in a case where the "List with radio buttons" selection method is used and the "Media" folder is specified as the default location.

(see figure 5.60)
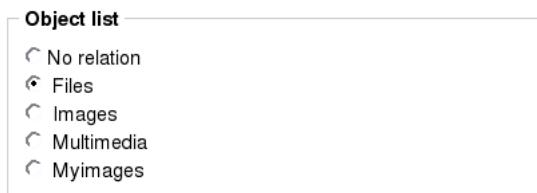
Figure 5.60: *Object attribute edit interface for the "Object relations" datatype (list with radio buttons).*

Raw output

The ".content" of an ezcontentobjectattribute (page 1042) object using this datatype returns a hash containing one element called "relation_list". This element can be either an empty array (if there are no relations) or an array of hashes. If there are related objects, each element will contain information about a specific object. The hash-structure of the elements is given in the table below.

Key	Type	Description
priority	string	The node's priority (positive or negative integer).
contentobject_id	string	The ID number of the content object.
contentobject_version	string	The number of the published version.
node_id	string	The ID number of the node.
parent_node_id	string	The ID number of the parent node.
contentclass_id	string	The ID number of the content class which the object is an instance of.
contentobject_remote_id	string	The remote id of the related content object.
identifier	boolean	Reserved for future use.
is_modified	boolean	Not in use (needed for backwards compatibility).

Changing/Deleting a related-object main location/node

When relating an object using a relation_list, if the related-object main location is changed and then deleted, the relation_list node_id-reference is not updated to reflect the related-objects changed main location/node.

Since the information provided in relation_list is the original one, you might have to use an object-fetch method to retrieve up-to-date information.

Example:

```
{foreach $item.data_map.authors.content.relation_list as $author_relation}
{set $person=fetch(content,object, hash(object_id, $author_relation.contentobject_id))}
{/foreach}
```

5

5.2.24   Option

Summary

Allows an option selection. [Webshop]

Properties

Name	Internal name	Searchable	Information collector
Option	ezoption	No.	Yes.

Description

This 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. The following screenshot shows the class attribute edit interface for this datatype.

(see figure 5.61)

Figure 5.61: *Class attribute edit interface for the "Option" datatype.*

The "Default name" parameter can be used to specify a name which will be used (as the name for the option) every time a new object is created.

Object attribute edit interface

The following screenshot shows the object attribute edit interface for this datatype.

(see figure 5.62)

Raw output

The ".content" of an ezcontentobjectattribute (page 1042) object using this datatype returns an ezoption (page 1096) object.

5

Figure 5.62: *Object attribute edit interface for the "Option" datatype.*

5

5.2.25 Package

Summary

DEPRECATED

Properties

Name	Internal name	Searchable	Information collector
Package	ezpackage	No.	No.

Description

This datatype should no longer be used.

5

5.2.26   Price

Summary

Stores a price (inc/ex VAT). [Webshop]

Properties

Name	Internal name	Searchable	Information collector
Price	ezprice	No.	No.

Description

This datatype allows the storage of a price and thus makes it possible to connect content objects with the e-commerce subsystem. The e-commerce features of eZ Publish are described in the "Webshop (page 162)" section of the "Concepts and basics" chapter. The following screenshot shows the class attribute edit interface for this datatype.

(see figure 5.63)

Figure 5.63: *Class attribute edit interface for the "Price" datatype.*

Parameters and usage

This is a special datatype which plugs more deeply into the system. Instances of any class containing the price datatype will automatically be treated as products. This datatype does not support prices in multiple currencies. It makes possible to set only one price value for each product (the system will use your locale currency for this price). However, you can use the "Multi-price (page 570)" datatype for multi-currency support. (Please note that price and multi-price datatypes are incompatible. A content class can only contain one price attribute or one multi-price attribute.)

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). The VAT parameters on the class level only control the default VAT settings that will be used when a new object is created. In other words, the VAT settings may be modified for each individual product / object.

Object attribute edit interface

The following screenshot shows the object attribute edit interface for this datatype.

(see figure 5.64)

Figure 5.64: *Object attribute edit interface for the "Price" datatype.*

Raw output

The ".content" of an ezcontentobjectattribute (page 1042) object using this datatype returns an ezprice (page 1104) object.

5.2.27 Product category

Summary

Stores a product category. [Webshop]

Properties

Name	Internal name	Searchable	Information collector
Product category	ezproductcategory	Yes.	No.

Description

This datatype allows the storage of a product category and thus makes it possible to charge different value added taxes depending on the product category. This functionality is described in the "VAT charging system (page 375)" section of the "Features" chapter. The following screenshot shows the class attribute edit interface for this datatype.

(see figure 5.65)

Figure 5.65: *Class attribute edit interface for the "Product category" datatype.*

Object attribute edit interface

The following screenshot shows the object attribute edit interface for this datatype.

(see figure 5.66)

Figure 5.66: *Object attribute edit interface for the "Product category" datatype.*

Raw output

The ".content" of an ezcontentobjectattribute (page 1042) object using this datatype returns an ezproductcategory (page 1105) object.

5

5.2.28   Range option

Summary

Allows an integer selection. [Webshop]

Properties

Name	Internal name	Searchable	Information collector
Range option	ezrangeoption	No.	No.

Description

This 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). The following screenshot shows the class attribute edit interface for this datatype.

(see figure 5.67)

Figure 5.67: *Class attribute edit interface for the "Range option" datatype.*

The "Default name" parameter can be used to specify a name which will be used (as the name for the option) every time a new object is created.

Object attribute edit interface

The following screenshot shows the object attribute edit interface for this datatype.

(see figure 5.68)

5

Figure 5.68: *Object attribute edit interface for the "Range option" datatype.*

Raw output

The ".content" of an ezcontentobjectattribute (page 1042) object using this datatype returns an ezrangeoption (page 1107) object.

5

5.2.29   Selection

Summary

Stores single and multiple choices.

Properties

Name	Internal name	Searchable	Information collector
Selection	ezselection	Yes.	No.

Description

This datatype allows the storage of single or multiple option selections. The options must be
defied on the class level. This datatype can for example be used to define different categories
for news articles (as shown in the example below: "Local", "National", etc.). The following
screenshot shows the class attribute edit interface for this datatype.

(see figure 5.69)

Figure 5.69: *Class attribute edit interface for the "Selection" datatype.*

5

Style

The "Style" parameter controls the behavior of the object attribute edit interface. It can be either
"Single" or "Multiple". While "Single" means that the object attribute edit interface will only

allow a single selection, the "Multiple" setting allows the selection of multiple options.

Options

The "Options" interface allows the specification of the options that should be available for selection when an object is edited. The options are identified by an identification number. The identification number of the first option is zero. Please note that the system does not give any warning if an option that is used is removed. In other words, removing options may result in an inconsistent database.

Object attribute edit interface

The following screenshot shows the object attribute edit interface for this datatype.

(see figure 5.70)

Figure 5.70: *Object attribute interface for the "Selection" datatype.*

Raw output

The ".content" of an ezcontentobjectattribute (page 1042) object using this datatype returns an array of the identification numbers (as strings) of the selected options.

5.2.30 Subtree subscription

Summary

DEPRECATED

Properties

Name	Internal name	Searchable	Information collector
Subtree subscription	ezsubtreesubscription	No.	No.

Description

This datatype should no longer be used. It has been replaced by the "My notification settings" interface of the "My account" part in the administration interface.

5

5.2.31   Text block

Summary

Stores multiple lines of unformatted text.

Properties

Name	Internal name	Searchable	Information collector
Text block	eztext	Yes.	Yes.

Description

This datatype allows the storage and retrieval of multiple lines of unformatted text. It is capable of handling up to 16,777,216 characters. The following screenshot shows the class edit interface for this datatype.

(see figure 5.71)

Figure 5.71: *Class edit interface for the "Text block" datatype.*

Preferred number of rows

The "Preferred number of rows" parameter makes it possible to control the height of the input field that is displayed when an object is being edited. The following options are available: 2, 5, 10, 15, 20 and 25.

Object attribute edit interface

The following screenshot shows the object attribute edit interface for this datatype.

(see figure 5.72)

5
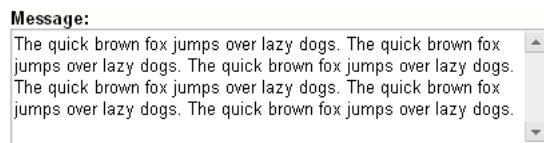
Figure 5.72: *Object attribute edit interface for the "Text block" datatype.*

Raw output

The ".content" of an ezcontentobjectattribute (page 1042) object using this datatype returns a string containing the actual text.

5

5.2.32   Text line

Summary

Stores a single line of unformatted text.

Properties

Name	Internal name	Searchable	Information collector
Text line	ezstring	Yes.	Yes.

Description

This datatype makes it possible to store and retrieve a single line of unformatted text. It is capable of handling up to 255 number of characters. The following screenshot shows the class edit interface for this datatype.

(see figure 5.73)

Figure 5.73: *Class edit interface for the "Text line" datatype.*

Default value

The "Default value" parameter makes it possible to set a default text. When the parameter is used and a new object is created, the contents of the attribute using this datatype will be preset to the given text. In the example above, the "Summary" attribute of new objects will be set to "The summary has not yet been added.".

Max string length

The "Max string length" parameter makes it possible to control the maximum number of characters that the input interface should allow. By default, this parameter is empty - which means

that the system will allow the maximum number of characters (255). In the example above, the parameter is set to 64. This means that the input interface will not allow the storage of strings that exceed 64 characters.

Object attribute edit interface

The following screenshot shows the object attribute edit interface for this datatype.

(see figure 5.74)

Figure 5.74: *Object attribute interface for the "Text line" datatype.*

Raw output

The ".content" of an ezcontentobjectattribute (page 1042) object using this datatype returns a string containing the actual text.

5.2.33   Time

Summary

Validates and stores a time value.

Properties

Name	Internal name	Searchable	Information collector
Time	eztime	No.	No.

Description

This datatype allows the validation, storage and retrieval of a time value consisting of hour and minute. The following screenshot shows the class attribute edit interface for this datatype.

(see figure 5.75)

Figure 5.75: *Class attribute edit interface for the "Time" datatype.*

Object attribute edit interface

The following screenshot shows the object attribute edit interface for this datatype.

(see figure 5.76)

Figure 5.76: *Object attribute edit interface for the "Time" datatype.*

Raw output

The ".content" of an ezcontentobjectattribute (page 1042) object using this datatype returns an eztime (page 1115) object.

5

5.2.34   URL

Summary

Validates and stores a URL / address.

Properties

Name	Internal name	Searchable	Information collector
URL	ezurl	No.	No.

Description

This datatype allows the validation and storage of a hyperlink. The following screenshot shows the class attribute edit interface for this datatype.

(see figure 5.77)

Figure 5.77: *Class attribute edit interface for the "URL" datatype.*

Object attribute edit interface

For each link, an address and an additional text may be stored. The following screenshot shows the object attribute edit interface for this datatype.

(see figure 5.78)

Figure 5.78: *Object attribute edit interface for the "URL" datatype.*

URL storage and checking

URLs that are input, either using the URL or the XML block datatype, are stored and handled in a special way. This solution makes it possible to view, edit and check if the URLs actually work on a massive scale.

Important note

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

Please refer to the "URL storage" (page 134) section of the "Concepts and basics" (page 93) chapter for more information about this feature.

Raw output

The ".content" of an ezcontentobjectattribute (page 1042) object using this datatype returns a string containing the actual address.

5

5.2.35   User account

Summary

Validates and stores info about a user.

Properties

Name	Internal name	Searchable	Information collector
User account	ezuser	Yes.	No.

Description

This datatype allows storage and retrieval of a single user account. Whenever an object that uses this datatype is published for the first time, the system will create a new user. The user's ID number will be the identification number of the object. The following screenshot shows the class attribute edit interface for this datatype.

(see figure 5.79)

Figure 5.79: *Class attribute edit interface for the "User account" datatype.*

Object attribute edit interface

For each user, the "User account" datatype stores the following information:

- A username

- A password

- An E-mail address

The username can only be modified before the object is published for the first time. The following screenshot shows the object attribute edit interface for this datatype.

(see figure 5.80)

Figure 5.80: *Object attribute edit interface for the "User account" datatype.*

User account settings

The properties of a user account can be tweaked by the administrator. This can be done by following the "Change user account settings" link from within the view of a user account object in the administration interface. Currently it is only possible to change whether the user account should be enabled or disabled (whether the user is allowed to log in to the system or not). The following screenshot shows the user account settings panel.

(see figure 5.81)

Figure 5.81: *Settings interface for the "User account" datatype.*

Raw output

The ".content" of an ezcontentobjectattribute (page 1042) object using this datatype returns an ezuser (page 1117) object.

5

5.2.36   XML block

Summary

Validates and stores multiple lines of formatted text.

Properties

Name	Internal name	Searchable	Information collector
XML block	ezxmltext	Yes.	No.

Description

Although there are no immediate visual clues, this datatype behaves quite differently compared to the regular "Text block" datatype. In particular, it is capable of validating and storing multiple lines of formatted text instead of just plain text. The text in an XML block must be formatted using a collection of predefined tags. The tags control the actual HTML markup of the content. eZ Publish comes with a collection of tags that cover the needs of typical everyday tasks. In addition, it is also possible to extend the system by creating custom tags for special needs.

Please note that the "<" sign means the beginning of an XML tag. If you need to insert a less-than sign in the normal text flow (for example, "3<5"), you will have to use the corresponding XML entity as shown below:

```
3&lt;5
```

From version 3.9, it is possible to input numerical entities and have them translated to their corresponding characters/symbols when the text is being rendered. For this to work, you need to enable the "AllowNumericEntities (page 1849)" configuration setting within the "[InputSettings]" block of an override for "ezxml.ini".

By default, the datatype supports the following XML tags:

- Headings
- Bold text
- Italic text
- Unformatted text
- Lists
- Tables
- Hyperlinks
- Anchors
- Object embedding
- Custom tags
- Paragraphs

The following screenshot shows the class attribute edit interface for this datatype.

(see figure 5.82)

5

Figure 5.82: *Class attribute edit interface for the "XML block" datatype.*

Object attribute edit interface

The following screenshot shows the object attribute edit interface for this datatype.

(see figure 5.83)

Figure 5.83: *Object attribute edit interface for the "XML block" datatype.*

Raw output

The ".content" of an ezcontentobjectattribute (page 1042) object using this datatype returns an ezxmltext (page 1126) object.

5

Headings

Headings/titles can be added by making use of either the "h" or the "header" tag. The "level" parameter controls the size/level of the heading, it must be a number between 1 and 6. The optional "class" parameter allows the use of a desired CSS class. The optional "anchor_name" parameter makes it possible to add an anchor to the heading. Usage:

```
<h [level=""] [class=""] [anchor_name=""] [custom_parameter="" [...]
]>Example</
h>
```

or

```
<header [level=""] [class=""] [anchor_name=""] [custom_parameter="" [...]
]>Example</
header>
```

The custom parameters are optional. Their names must be specified using the "CustomAttributes[] (page 1744)" array in the [header] block within an override for the "content.ini" configuration file. When used, a custom parameter will be available as a template variable with the same name as it was specified in the tag itself.

By default, the specified levels are increased by one. In other words, a level 1 header in the XML block will become a level 2 header (H2) in the resulting HTML. The reason for this is because the H1 tag is reserved for the name / main title of the content object. The headings inside the XML block will thus become subheadings of the main title. This behavior can be changed by creating an override template for the "/content/datatype/view/ezxmltags/header.tpl" template (it can not be controlled from within an configuration file).

5

Bold text

Bold text can be achieved by using one of the following tags: "b", "bold" or "strong". The optional "class" parameter allows the use of a desired CSS class. Usage:

```
<b [class=""] [custom_parameter="" [...] ]>Bold text.</b>
```

or

```
<bold [class=""] [custom_parameter="" [...] ]>Bold text.</bold>
```

or

```
<strong [class=""] [custom_parameter="" [...] ]>Bold text.</strong>
```

The custom parameters are optional. Their names must be specified using the "CustomAttributes[] (page 1744)" array in the [strong] block within an override for the "content.ini" configuration file. When used, a custom parameter will be available as a template variable with the same name as it was specified in the tag itself.

5

Italic text

Italic/emphasized text can be achieved by using one of the following tags: "i", "em" or "emphasize". The optional "class" parameter allows the use of a desired CSS class. Usage:

```
<i [class=""] [custom_parameter="" [...] ]>Emphasized text.</i>
```

or

```
<em [class=""] [custom_parameter="" [...] ]>Emphasized text.</em>
```

or

```
<emphasize [class=""] [custom_parameter="" [...] ]>Emphasized text.</emphasize>
```

The custom parameters are optional. Their names must be specified using the "CustomAttributes[] (page 1744)" array in the [emphasize] block within an override for the "content.ini" configuration file. When used, a custom parameter will be available as a template variable with the same name as it was specified in the tag itself.

5

Unformatted text

The "literal" tag can be used to output unformatted text, for example program source code, HTML code, XML content, etc. Everything that is inside an a literal block will be rendered in the same way (character by character) as it is within the literal tags (the text will be output using the HTML PRE tags). The optional "class" parameter allows the use of a desired CSS class. Usage:

```
<literal [class=""] [custom_parameter="" [...] ]>Example<\literal>
```

Please note that in the example above, the slash is in the wrong way within the tag that terminates the literal block. This was done in order to make the tag appear on the documentation page (since we're using literal tags to make code blocks). In other words, it should be terminated with a frontslash instead of a backslash.

The custom parameters are optional. Their names must be specified using the "CustomAttributes[] (page 1744)" array in the [literal] block within an override for the "content.ini" configuration file. When used, a custom parameter will be available as a template variable with the same name as it was specified in the tag itself.

5

Lists

It is possible to create lists in the same way as in HTML by making use of the "ol", "ul" and "li" tags. The lists can be nested. The optional "class" parameter allows the use of a desired CSS class. The following examples demonstrate the usage of ordered and unordered lists.

Ordered lists

```
<ol [class=""] [custom_parameter="" [...] ]>
    <li [class=""] [custom_parameter="" [...] ]>Element 1</li>
    <li [class=""] [custom_parameter="" [...] ]>Element 2</li>
    <li [class=""] [custom_parameter="" [...] ]>Element 3</li>
</ol>
```

Unordered lists

```
<ul [class=""] [custom_parameter="" [...] ]>
    <li [class=""] [custom_parameter="" [...] ]>Element 1</li>
    <li [class=""] [custom_parameter="" [...] ]>Element 2</li>
    <li [class=""] [custom_parameter="" [...] ]>Element 3</li>
</ul>
```

The custom parameters are optional. In order to use them, their names must be specified using the "CustomAttributes[] (page 1744)" array in the [ol], [ul], [li] blocks within an override for the "content.ini" configuration file. When used, a custom parameter will be available as a template variable with the same name as it was specified in the tag itself.

5

Tables

Tables can be created in the same way as in HTML using "table", "tr", "th" and "td" tags. The tables can be nested. Usage:

```
<table [class=""] [border=""] [width=""] [custom_parameter="" [...] ]>
...
</table>
```

The "class", "border" and "width" parameters are optional. The "class" parameter can be used to assign a desired CSS class. The "border" parameter can be used to set a border (number of pixels). The "width" parameter can be used to control the table width (either 0-100% or number of pixels). Table content should be written according to normal HTML table syntax with "tr", "th" and "td" tags, see below.

Table rows

Table rows can be created in the same way as in HTML:

```
<tr [class=""] [custom_parameter="" [...] ]>Table row content goes here.</tr>
```

The "class" parameter can be used to set the desired CSS class.

Table headers

Table headers can be created in the same way as in HTML:

```
<th [class=""] [width=""] [rowspan=""] [colspan=""] [custom_parameter="" [...]
]>Example.</
th>
```

All parameters are optional. The "class" parameter can be used to set the desired CSS class. The "width" parameter can be used to set the width (either as percentage or number of pixels). The "rowspan" and "colspan" parameters are the same as in HTML.

Table data/cell

Table data/cells can be created in the same way as in HTML:

```
<td [class=""] [width=""] [rowspan=""] [colspan=""] [custom_parameter="" [...]
]>Example.</
td>
```

5

All parameters are optional. The "class" parameter can be used to set the desired CSS class. The "width" parameter can be used to set the width (either as percentage or number of pixels). The "rowspan" and "colspan" parameters are the same as in HTML.

Note that all custom parameters mentioned in the usage examples are optional as well. In order to use them, their names must be specified using the "CustomAttributes[] (page 1744)" array in the [table], [tr], [th], [td] blocks within an override for the "content.ini" configuration file. When used, a custom parameter will be available as a template variable with the same name as it was specified in the tag itself.

5

Hyperlinks

Hyperlinks can be inserted by making use of the "a" or the "link" tags. Usage:

```
<a href="" [view=""] [target=""] [ class=""] [title=""] [id=""]
[custom_parameter="" [...] ]>Example.</
a>
```

or

```
<link href="" [view=""] [target=""] [ class=""] [title=""] [id=""]
[custom_parameter="" [...] ]>Example.</
link>
```

The "href" parameter is required and it must be set to a valid address (either external or internal).

The "view" parameter will only have effect when used together with internal links (see below). This parameter makes it possible to specify the view mode that will be used when the node (or object) which the internal link points to is shown. By default, the system always uses the "full" view mode to show the contents of internal links.

The "target" parameter can be used to control how the target URL should be opened (inside the existing/active browser window/tab or within a new window/tab). The "class" parameter can be used to specify a CSS class that should be used when the link is rendered. The "title" parameter can be used to specify a short title text (will be shown when the pointer is hovering over the link). The "id" parameter is for assigning unique identifiers.

The custom parameters are optional. Their names must be specified using the "CustomAttributes[] (page 1744)" array in the [link] block within an override for the "content.ini" configuration file. When used, a custom parameter will be available as a template variable with the same name as it was specified in the tag itself.

Internal links

It is possible to create internal links (to other nodes and objects) by making use of the "eznode://" and the "ezobject://" notation. The internal links will be created dynamically based on the node/object ID numbers. In other words, if a node is moved, the link(s) will point to the new location(s) and thus they will not be broken.

5

Link to a node

A link to a node can be created either by specifying the target node's ID number or the node path. The following examples demonstrate how an internal link to node number 128 can be created.

```
<a href="eznode://128">Example.</a>
```

or

```
<link href="eznode://128">Example.</link>
```

The following examples demonstrate how an internal link to a node located at "products/ computers/example" can be created.

```
<a href="eznode://products/computers/example">Example.</a>
```

or

```
<link href="eznode://products/computers/example">Example.</link>
```

Link to an object

The following examples demonstrate how an internal link to object number 1024 can be created.

```
<a href="ezobject://1024">Example.</a>
```

or

```
<link href="ezobject://1024">Example.</link>
```

When object linking is used, the destination address will be generated using the main node assignment of the target object.

Alternate view modes

The "view" parameter can be used together with both "eznode://" and "ezobject://" links. It makes it possible to display the specified node (if object, its main node will be used) using a specific view mode instead of the default "full" view mode. The following example shows how to create an internal link that (when clicked/followed) brings up node number 1024 using the "line" view mode.

```
<a href="eznode://1024" view="line">Example (as line).</a>
```

5

Anchors

The "anchor" tag makes it possible to insert HTML anchors inside the XML block. The inserted anchors will work like standard HTML anchors. Usage:

```
<anchor name="" [custom_parameter="" [...] ] />
```

The "name" parameter must be set to a unique identifier for the anchor. Anchors can be reached by appending the hash character (#) followed directly by the name of the anchor that the browser should jump to. Example: http://www.example.com/hobbies#music

The custom parameters are optional. Their names must be specified using the "CustomAttributes[] (page 1744)" array in the [anchor] block within an override for the "content.ini" configuration file. When used, a custom parameter will be available as a template variable with the same name as it was specified in the tag itself.

5

Object embedding

The "embed" tag makes it possible to insert an arbitrary content object directly in the XML block. It can for example be used to embed images. Usage:

```
<embed href="" [class=""] [view=""] [align=""] [target=""] [size=""] [id=""]
[custom_parameter="" [...] ] /
>
```

With this tag, embedded objects are inserted as "block elements". Such an element always begins on a new line when it is displayed. It exists in its own virtual box and is always followed by a carriage return (like if someone hit the "Enter" key after inserting the object). This means that, for example, inserting an image using the "embed" tag will break the current paragraph. The "embed" tag is represented by block-level tags in the resulting XHTML code.

The "embed-inline" tag makes it possible to insert objects as inline elements. For example, this tag allows you to insert an image inside a text line. Usage:

```
<embed-inline href="" [class=""] [view=""] [align=""] [target=""] [size=""]
[id=""] [custom_parameter="" [...] ] /
>
```

This tag is represented by inline tags in the resulting XHTML code. The templates that are used for rendering "embed-inline" tags must not contain block-level XHTML tags.

The following table reveals the list of parameters supported by the "embed" and "embed-inline" tags.

Parameter	Description	Required
href	The "href" parameter must be a valid link to either a node or an object using the same notation as for hyperlinks (for example "eznode://134", "eznode://path/to/some/node" and "ezobject://1024"). If the provided link is a link to a node, eZ Publish will use the object that is encapsulated by that node. In other words, in both cases it is the object that will be inserted (the node notation is just a wrapper).	Yes.
class	The "class" parameter makes it possible to specify a custom stylesheet that should be used. In the template, the	No.

	specified stylesheet will be available in the $classification variable.	
view	The "view" parameter makes it possible to specify the view mode that should be used when the object is rendered (for example "full", "line" and so on). By default, the system uses the "embed" view mode when rendering objects that are inserted using the "embed" tag, while the "embed-inline" view mode is used together with "embed-inline" tags.	No.
align	The "align" parameter can be used to specify the positioning of the embedded object; possible values are "left", "center" and "right".	No.
target	The "target" parameter can be used to set the opening method (same browser tab/ window or new browser tab/ window) for the embedded item (for example "_self", "_blank", etc.).	No.
size	The "size" parameter can be used to set the image size that should be used when an image object is embedded (for example "small", "medium", "large", etc.). The available sizes are defined by image.ini.	No.
id	The "id" parameter makes it possible to assign a unique ID which will be the ID attribute in the resulting HTML.	No.
custom parameters	The available custom parameters must be specified using the "CustomAttributes[] (page 1744)" array in either the [embed] or [embed-inline] block within an over-	No.

5

	ride for the "content.ini" configuration file. When used, a custom parameter will be available as a template variable with the same name as it was specified in the tag itself.	

5

Custom tags

In addition to the default tags described above, the "XML block" datatype makes it possible to use custom tags. A custom tag can be used both as a block or an inline element. Custom tags must be specified using the "AvailableCustomTags[]" array in the [CustomTagSettings] block within an override for the "content.ini" configuration file. When the XML is rendered, the contents of a custom tag will be replaced by a custom template. The name of the template must be specified using the "name" parameter. Example of usage:

```
<custom name="template_name" [custom_parameter="value" [...] ]>
The quick brown fox jumps over the lazy dog.
</custom>
```

The custom tag in the example above will be replaced by a template called "template_name.tpl". This template must be located in the following directory within the current design: "/templates/ content/datatype/view/ezxmltags/" (or one of the fallback designs). It is also possible to create an override template. The contents of the tag will be available in the "$content" variable within the inserted template.

The custom parameters are optional. Their names must be specified using the "CustomAttributes[] (page 1744)" array within an override for the "content.ini" configuration file. When used, a custom parameter will be available as a template variable with the same name as it was specified in the tag itself.

5

Paragraphs

Paragraphs can be added by making use of either the 'p' or the 'paragraph' tag.

The optional 'class' parameter allows the use of a desired CSS class. If you do not specify the class parameter, the paragraph will be displayed in a natural way (without tags) in the administration interface. To create a non-classified paragraph, you can simply press 'Enter' key twice.

Usage:

```
<p [class=""] [custom_parameter="" [...] ]>Example</p>
```

or

```
<paragraph [class=""] [custom_parameter="" [...] ]>Example</paragraph>
```

By default, the system will use 'p' tag in the resulting XHTML code. This behavior can be changed by creating an override template for the '/content/datatype/view/ezxmltags/paragraph.tpl'.

The custom parameters are optional. Their names must be specified using the "CustomAttributes[] (page 1744)" array in the [paragraph] block within an override for the "content.ini" configuration file. When used, a custom parameter will be available as a template variable with the same name as it was specified in the tag itself.

5

5.3 Content classes

The classes are documented in the following sections:

\begin{itemize} \item{} Content\refer{labelceaae62cd6016eafcbfeb75e718a47e9} \item{} Media\refer{label306a386a3cb403e5f729d27c420ed5dc} \item{} Users\refer{label7441b093a466bfae04dd5d3ae4ba733d} \end{itemize}

5

5.3.1   Content

\begin{description} \item[Article]\refer{labele4f9ab7e282f65ee0a35f03c7773d6e5}\ \\ Defines a structure for storing articles.

\item[Comment]\refer{label935d2b39421c537bc1ca18eb41b8867e}\ \\ Defines a structure for storing comments/feedback.

\item[Company]\refer{labelfe040e4422c9b76e9081bb7711a5d7c4}\ \\ Defines a structure for storing information about companies.

\item[Feedback form]\refer{labelf09500fb1504c73fd0c3af674a8b2109}\ \\ Defines a structure for feedback forms.

\item[Folder]\refer{label7223eaa0fd7b25574944156ebe808574}\ \\ Defines a structure for folders / information pages.

\item[Forum]\refer{labelecc8c80df8a36ae7020e76c27a5fcf43}\ \\ Defines a structure for storing forums.

\item[Forum reply]\refer{label26114cb1884fb6587dde59fe3fb2329a}\ \\ Defines a structure for storing forum replies.

\item[Forum topic]\refer{labelb714284296adec4e0508b4e3170fc83e}\ \\ Defines a structure for storing forum topics.

\item[Gallery]\refer{labelaf5327ea5745b49a1d1abcc103c99f4a}\ \\ Defines a structure for storing image galleries.

\item[Link]\refer{label60f8573d2d0b7c96660807b2a4b2bacb}\ \\ Defines a structure for storing hyperlinks.

\item[Person]\refer{label8b8d33445c9de81e6b5e47a2597cc219}\ \\ Defines a structure for storing information about people.

\item[Poll]\refer{labeld74c39bd6551c16cc1470cbd10129ef6}\ \\ Defines a structure for storing polls.

\item[Product]\refer{label295701aa2b68b3490a47fd4dc8e98cd4}\ \\ Defines a structure for storing information about products.

\item[Review]\refer{labeld83b6ad0eac7289b3253249427041778}\ \\ Defines a structure for storing product reviews.

\item[Weblog]\refer{label062a76b8633dd1f3894b2adf1b19a0b4}\ \\ Defines a structure for storing personal logs.

\end{description}

5

Article

Summary

Defines a structure for storing articles.

Properties

Name	Identifier	Container	Object name pattern
Article	article	Yes.	<short_title|title>

Attributes

Name	Identifier	Datatype	R	S	C	T
Title	title	Text line	Yes.	Yes.	No.	Yes.
Short title	short_title	Text line	No.	Yes.	No.	Yes.
Author	author	Authors	No.	Yes.	No.	Yes.
Intro	intro	XML block	Yes.	Yes.	No.	Yes.
Body	body	XML block	Yes.	Yes.	No.	Yes.
Enable comments	enable_comments	Checkbox	No.	No.	No.	No.
Image	image	Object relation	No.	Yes.	No.	No.
Keywords	keywords	Keywords	No.	Yes.	No.	Yes.

R Required
S Searchable
C Collector
T Translatable

5

Comment

Summary

Defines a structure for storing comments/feedback.

Properties

Name	Identifier	Container	Object name pattern
Comment	comment	Yes.	<subject>

Attributes

Name	Identifier	Datatype	R	S	C	T
Subject	subject	Text line	Yes.	Yes.	No.	Yes.
Author	author	Text line	Yes.	Yes.	No.	Yes.
Message	message	Text block	Yes.	Yes.	No.	Yes.

R Required
S Searchable
C Collector
T Translatable

5

Company

Summary

Defines a structure for storing information about companies.

Properties

Name	Identifier	Container	Object name pattern
Company	company	Yes.	<company_name>

Attributes

Name	Identifier	Datatype	R	S	C	T
Company name	company_name	Text line	Yes.	Yes.	No.	Yes.
Company number	company_number	Text line	No.	Yes.	No.	Yes.
Company address	company_address	Matrix	No.	Yes.	No.	Yes.
Logo	logo	Image	No.	Yes.	No.	Yes.
Additional information	additional_information	Text block	No.	Yes.	No.	Yes.
Contact information	contact_information	Matrix	No.	Yes.	No.	Yes.
Contacts	contacts	Object relations	No.	Yes.	No.	Yes.

R Required
S Searchable
C Collector
T Translatable

5

Feedback form

Summary

Defines a structure for feedback forms.

Properties

Name	Identifier	Container	Object name pattern
Feedback form	feedback_form	Yes.	<name>

Attributes

Name	Identifier	Datatype	R	S	C	T
Name	name	Text line	Yes.	Yes.	No.	Yes.
Description	description	XML block	No.	Yes.	No.	Yes.
Subject	subject	Text line	Yes.	Yes.	Yes.	Yes.
Message	message	Text block	Yes.	Yes.	Yes.	Yes.
Email	email	E-mail	Yes.	No.	Yes.	Yes.
Recipient	recipient	E-mail	Yes.	No.	No.	Yes.

R   Required
S   Searchable
C   Collector
T   Translatable

Folder

Summary

Defines a structure for folders / information pages.

Properties

Name	Identifier	Container	Object name pattern
Folder	folder	Yes.	\<name|short_name\>

Attributes

Name	Identifier	Datatype	R	S	C	T
Name	name	Text line	Yes.	Yes.	No.	Yes.
Short name	short_name	Text line	Yes.	Yes.	No.	Yes.
Short description	short_description	XML block	No.	Yes.	No.	Yes.
Description	description	XML block	No.	Yes.	No.	Yes.
Show children	show_children	Checkbox	No.	No.	No.	No.

R   Required
S   Searchable
C   Collector
T   Translatable

5

Forum

Summary

Defines a structure for storing forums.

Properties

Name	Identifier	Container	Object name pattern
Forum	forum	Yes.	<name>

Attributes

Name	Identifier	Datatype	R	S	C	T
Name	name	Text line	Yes.	Yes.	No.	Yes.
Description	description	XML block	No.	Yes.	No.	Yes.

R Required
S Searchable
C Collector
T Translatable

5

Forum reply

Summary

Defines a structure for storing forum replies.

Properties

Name	Identifier	Container	Object name pattern
Forum reply	forum_reply	Yes.	<subject>

Attributes

Name	Identifier	Datatype	R	S	C	T
Subject	subject	Text line	Yes.	Yes.	No.	Yes.
Message	message	Text block	Yes.	Yes.	No.	Yes.

R Required
S Searchable
C Collector
T Translatable

5

Forum topic

Summary

Defines a structure for storing forum topics.

Properties

Name	Identifier	Container	Object name pattern
Forum topic	forum_topic	Yes.	<subject>

Attributes

Name	Identifier	Datatype	R	S	C	T
Subject	subject	Text line	Yes.	Yes.	No.	Yes.
Message	message	Text block	Yes.	Yes.	No.	Yes.
Sticky	sticky	Checkbox	No.	No.	No.	Yes.
Notifiy me about updates	nofity_me	Subtree subscription	No.	No.	No.	Yes.

R Required
S Searchable
C Collector
T Translatable

5

Gallery

Summary

Defines a structure for storing image galleries.

Properties

Name	Identifier	Container	Object name pattern
Gallery	gallery	Yes.	\<name\>

Attributes

Name	Identifier	Datatype	R	S	C	T
Name	name	Text line	Yes.	Yes.	No.	Yes.
Short description	short_description	XML block	No.	Yes.	No.	Yes.
Description	description	XML block	No.	Yes.	No.	Yes.
Image	image	Object relation	No.	Yes.	No.	Yes.

R Required
S Searchable
C Collector
T Translatable

5

Link

Summary

Defines a structure for storing hyperlinks.

Properties

Name	Identifier	Container	Object name pattern
Link	link	Yes.	<name>

Attributes

Name	Identifier	Datatype	R	S	C	T
Name	name	Text line	Yes.	Yes.	No.	Yes.
Description	description	XML block	No.	Yes.	No.	Yes.
Location	location	URL	No.	No.	No.	Yes.

R   Required
S   Searchable
C   Collector
T   Translatable

5

Person

Summary

Defines a structure for storing information about people.

Properties

Name	Identifier	Container	Object name pattern
Person	person	Yes.	<first_name> <last_name>

Attributes

Name	Identifier	Datatype	R	S	C	T
First name	first_name	Text line	Yes.	Yes.	No.	Yes.
Last name	last_name	Text line	Yes.	Yes.	No.	Yes.
Job title	job_title	Text line	No.	Yes.	No.	Yes.
Contact information	contact_information	Matrix	No.	Yes.	No.	Yes.
Picture	picture	Object relation	No.	Yes.	No.	Yes.
Comment	comment	XML block	No.	Yes.	No.	Yes.

R   Required
S   Searchable
C   Collector
T   Translatable

5

Poll

Summary

Defines a structure for storing polls.

Properties

Name	Identifier	Container	Object name pattern
Poll	poll	Yes.	<name>

Attributes

Name	Identifier	Datatype	R	S	C	T
Name	name	Text line	Yes.	Yes.	No.	Yes.
Description	description	XML block	No.	Yes.	No.	Yes.
Question	question	Option	Yes.	No.	Yes.	Yes.

R Required
S Searchable
C Collector
T Translatable

Product

Summary

Defines a structure for storing information about products.

Properties

Name	Identifier	Container	Object name pattern
Product	product	Yes.	<name>

Attributes

Name	Identifier	Datatype	R	S	C	T
Name	name	Text line	Yes.	Yes.	No.	Yes.
Short description	short_description	XML block	No.	Yes.	No.	Yes.
Description	description	XML block	No.	Yes.	No.	Yes.
Image	image	Object relation	No.	Yes.	No.	Yes.
Price	price	Price	No.	No.	No.	Yes.

R   Required
S   Searchable
C   Collector
T   Translatable

5

Review

Summary

Defines a structure for storing product reviews.

Properties

Name	Identifier	Container	Object name pattern
Review	review	Yes.	<summary>

Attributes

Name	Identifier	Datatype	R	S	C	T
Summary	summary	Text line	Yes.	Yes.	No.	Yes.
Author	author	Text line	No.	Yes.	No.	Yes.
Message	message	Text block	No.	Yes.	No.	Yes.
Rating	rating	Selection	No.	Yes.	No.	Yes.

R Required
S Searchable
C Collector
T Translatable

5

Weblog

Summary

Defines a structure for storing personal logs.

Properties

Name	Identifier	Container	Object name pattern
Weblog	weblog	Yes.	<title>

Attributes

Name	Identifier	Datatype	R	S	C	T
Title	title	Text line	Yes.	Yes.	No.	Yes.
Message	message	XML block	Yes.	Yes.	No.	Yes.
Enable comments	enable_commeents	Checkbox	No.	Yes.	No.	Yes.

R Required
S Searchable
C Collector
T Translatable

5

5.3.2   Media

\begin{description} \item[File]\refer{label6810fe0c506bf8387ea2c9a4ddce0977}\ \\ Defines a structure for storing binary files.

\item[Flash]\refer{label9d86751bb9fd4fdca00073faf2871438}\ \\ Defines a structure for storing Macromedia Flash files.

\item[Image]\refer{label2ae36be3ee10b2d2d1d33144bd734851}\ \\ Defines a structure for storing digital images.

\item[QuickTime]\refer{label02ff049de927042755d4f418d2ad533f}\ \\ Defines a structure for storing Apple QuickTime files.

\item[Real video]\refer{label36ffc88530c0cd8fe476e58155d015b1}\ \\ Defines a structure for storing Real video files.

\item[Windows media]\refer{labela11f65bc5471d7ddd1528d09e950cf03}\ \\ Defines a structure for storing ".avi" files.

\end{description}

5

File

Summary

Defines a structure for storing binary files.

Properties

Name	Identifier	Container	Object name pattern
File	file	Yes.	<name>

Attributes

Name	Identifier	Datatype	R	S	C	T
Name	name	Text line	Yes.	Yes.	No.	Yes.
Description	description	XML block	No.	Yes.	No.	Yes.
File	file	File	Yes.	No.	No.	Yes.

R   Required
S   Searchable
C   Collector
T   Translatable

5

Flash

Summary

Defines a structure for storing Macromedia Flash files.

Properties

Name	Identifier	Container	Object name pattern
Flash	flash	Yes.	<name>

Attributes

Name	Identifier	Datatype	R	S	C	T
Name	name	Text line	Yes.	Yes.	No.	Yes.
Description	description	XML block	No.	Yes.	No.	Yes.
File	file	Media	Yes.	No.	No.	Yes.

R   Required
S   Searchable
C   Collector
T   Translatable

5

Image

Summary

Defines a structure for storing digital images.

Properties

Name	Identifier	Container	Object name pattern
Image	image	Yes.	<name>

Attributes

Name	Identifier	Datatype	R	S	C	T
Name	name	Text line	Yes.	Yes.	No.	Yes.
Caption	caption	XML block	No.	Yes.	No.	Yes.
Image	image	Image	No.	No.	No.	Yes.

R   Required
S   Searchable
C   Collector
T   Translatable

5

QuickTime

Summary

Defines a structure for storing Apple QuickTime files.

Properties

Name	Identifier	Container	Object name pattern
QuickTime	quicktime	Yes.	<name>

Attributes

Name	Identifier	Datatype	R	S	C	T
Name	name	Text line	Yes.	Yes.	No.	Yes.
Description	description	XML block	No.	Yes.	No.	Yes.
File	file	Media	Yes.	No.	No.	Yes.

R   Required
S   Searchable
C   Collector
T   Translatable

5

Real video

Summary

Defines a structure for storing Real video files.

Properties

Name	Identifier	Container	Object name pattern
Real video	real_video	Yes.	<name>

Attributes

Name	Identifier	Datatype	R	S	C	T
Name	name	Text line	Yes.	Yes.	No.	Yes.
Description	description	XML block	No.	Yes.	No.	Yes.
File	file	Media	Yes.	No.	No.	Yes.

R Required
S Searchable
C Collector
T Translatable

5

Windows media

Summary

Defines a structure for storing ".avi" files.

Properties

Name	Identifier	Container	Object name pattern
Windows media	windows_media	Yes.	<name>

Attributes

Name	Identifier	Datatype	R	S	C	T
Name	name	Text line	Yes.	Yes.	No.	Yes.
Description	description	XML block	No.	Yes.	No.	Yes.
File	file	Media	Yes.	No.	No.	Yes.

R   Required
S   Searchable
C   Collector
T   Translatable

5

5.3.3 Users

\begin{description} \item[User]\refer{label2234cd64c3f1fdd1a8f3d83b0470eb0c}\ \\ Defines a structure for storing user accounts.

\item[User group]\refer{labele8d8d2c2aa021245a4f0064c3df34ecc}\ \\ Defines a structure for storing user groups.

\end{description}

5

User

Summary

Defines a structure for storing user accounts.

Properties

Name	Identifier	Container	Object name pattern
User	user	Yes.	<first_name> <last_name>

Attributes

Name	Identifier	Datatype	R	S	C	T
First name	first_name	Text line	Yes.	Yes.	No.	Yes.
Last name	last_name	Text line	Yes.	Yes.	No.	Yes.
User account	user_account	User account	Yes.	Yes.	No.	Yes.
Signature	signature	Text block	No.	Yes.	No.	Yes.
Image	image	Image	No.	No.	No.	Yes.

R Required
S Searchable
C Collector
T Translatable

5

User group

Summary

Defines a structure for storing user groups.

Properties

Name	Identifier	Container	Object name pattern
User group	user_group	Yes.	\<name\>

Attributes

Name	Identifier	Datatype	R	S	C	T
Name	name	Text line	Yes.	Yes.	No.	Yes.
Description	description	Text line	No.	Yes.	No.	Yes.

R   Required
S   Searchable
C   Collector
T   Translatable

5

5.4 Modules

\begin{description} \item[class]\refer{labelc146a473e33839db3a5fc701ae1e3cff}\ \\ Provides views for managing classes, class groups, etc.

\item[collaboration]\refer{labelcdbffe230491e005f12fe18ad35b290c}\ \\ Provides an interface to the collaboration engine.

\item[content]\refer{labelbcc7790f765a2f37194a891e41311b12}\ \\ Provides views for managing content (nodes, objects, searching, etc.)

\item[error]\refer{label972fbecf33e1466485241fffaa0c244f}\ \\ Provides an interface for error handling / reporting.

\item[ezinfo]\refer{label1ac35587886356901005765e184af596}\ \\ Provides views for displaying information about eZ publish.

\item[form]\refer{labela16f94e5a7b4c527f3052dda925eac1a}\ \\ Provides a view that generates an E-mail containing the data that was posted (DEPRECATED).

\item[infocollector]\refer{label8ac1efacf19076e81688deca52f1f5b9}\ \\ Provides views for managing collected information.

\item[layout]\refer{label591b2f550b6268bddcef86f4c2213768}\ \\ Provides a view that makes it possible to use alternative pagelayouts.

\item[notification]\refer{labelcbe8851bddcbcc298ea473b72ab483fd}\ \\ Provides an interface to the notification engine.

\item[package]\refer{label2a67ee5fef16f267188ff312b9d36538}\ \\ Provides views for importing/exporting packages.

\item[pdf]\refer{label6c2763cc00da3c12b89073fdbf39a56e}\ \\ Provides views for configuring PDF exports (DEPRECATED).

\item[reference]\refer{labelb47ab678ae94b1db697026d90e16c362}\ \\ Provides a view for displaying documentation generated by Doxygen.

\item[role]\refer{label94182e871cec747f7b9ec971d783412a}\ \\ Provides views for managing roles.

\item[rss]\refer{label29f5d5ac43621cb5e38ee57a0dc4f00f}\ \\ Provides views for managing RSS imports and exports.

\item[search]\refer{label49430a14de263750a46fb07150b1d7e3}\ \\ Provides a view that displays search statistics.

\item[section]\refer{label2a3a1534fea5512800f257e40994b329}\ \\ Provides views for managing sections.

\item[setup]\refer{labelb05971b446ee1cd4ba485c20d12902e1}\ \\ Provides the web based setup wizard.

\item[shop]\refer{labela15950e81574941c6276c8cdcd8c80f3}\ \\ Provides views for the webshop (basket, wish list, order list, etc.).

5

\item[trigger]\refer{labelf19407e4a0462b6c4dcd036e8e185731}\ \\ Provides a view for managing workflow triggers.

\item[url]\refer{label173fb25b4957feb67b5dc718c710878a}\ \\ Provides views for managing the URLs stored in the database.

\item[user]\refer{labele9f8960664d8112cc37534696dea1c84}\ \\ Provides views for logging users in/out, password changing, etc.

\item[workflow]\refer{labelc4dfa05ca43cea88f846537dbafa6f9c}\ \\ Provides views for managing workflows, workflow groups, workflow events, etc.

\end{description}

5

5.4.1 class

Summary

Provides views for managing classes, class groups, etc.

Description

This module provides several interfaces that can be used to view and manage the content classes and class groups that are present in the system. The views that the module provides are used by the "Class" section of the "Setup" part of the administration interface.

The module components are documented in the following sections:

- Fetch functions (page 648)

- Views (page 655)

5

Fetch functions

attribute_list   (page 649)

>   Fetches the attributes of a class.

latest_list   (page 650)

>   Fetches the most recently modified classes.

list   (page 651)

>   Fetches a collection of classes.

override_template_list   (page 653)

>   Fetches the override rules associated with a class.

5

attribute_list

Summary

Fetches the attributes of a class.

Usage

```
fetch( 'class', 'attribute_list', hash( 'class_id', class_id ) )
```

Parameters

Name	Type	Description	Required
class_id	interger	The ID number of the target class.	Yes.

Returns

Array of ezcontentclassattribute (page 1026) objects or FALSE.

Description

This function fetches the attributes of a class specified by the "class_id" parameter. The function returns an array of ezcontentclassattribute (page 1026) objects or FALSE.

Examples

Example 1

```
{def $attributes=fetch( 'class', 'attribute_list', hash( 'class_id', 13 ) )}

{foreach $attributes as $attribute}
    {$attribute.name|wash} <br />
{/foreach}
```

5

Outputs the names of the attributes that belong to class number 13.

latest_list

Summary

Fetches the most recently modified classes.

Usage

```
fetch( 'class', 'latest_list', hash( [ 'offset', offset ] ,
                                      [ 'limit',  limit  ] ) )
```

Parameters

Name	Type	Description	Required
offset	integer	The offset to start at.	No.
limit	integer	The number of classes that should be fetched.	No.

Returns

Array of ezcontentclass (page 1022) objects.

Description

This function fetches the most recently modified classes. The function returns an array of ezcontentclass (page 1022) objects. The "offset" and "limit" parameters are optional and can be used to narrow down the result. If the "offset" and "limit" parameters are omitted, the function will simply return all the available classes.

Examples

Example 1

```
{def $classes=fetch( 'class', 'latest_list', hash( 'limit', 10 ) )}

{foreach $classes as $class}
    {$class.name|wash} <br />
{/foreach}
```

Outputs the names of the ten most recently modified classes.

5

list

Summary

Fetches a collection of classes.

Usage

```
fetch( 'class', 'list', hash( [ 'class_filter', class_filter ],
                              [ 'sort_by',     sort_by    ] ) )
```

Parameters

Name	Type	Description	Required
class_filter	array	The classes that should be filtered.	No.
sort_by	array	The sorting mechanism that should be used.	No.

Returns

Array of ezcontentclass (page 1022) objects or FALSE.

Description

This function fetches a collection of classes. The optional "class_filter" parameter can be used to fetch only a given set of classes. This parameter must be either an array of class ID numbers or an array of class identifier strings (in other words: mixing is not possible). If the "class_filter" array is omitted, all classes will be returned.

The optional "sort_by" parameter makes it possible to sort the result in different ways. This parameter must be provided as an array. The first element of the array is the desired sorting method. The second element of the array is the sorting direction, it can be either true() or false() - ascending or descending. The following table shows the sorting methods that can be used.

Sorting method	Description
id	The classes are sorted by their ID numbers.
name	The classes are sorted by their names.

You can only use one sorting method per query. In other words, you can sort the classes by name or by ID, but not both in the same query.

5

Examples

Example 1

```
{def $classes=fetch( 'class', 'list' )}

{foreach $classes as $class}
    {$class.name|wash} <br />
{/foreach}
```

Outputs the names of all classes.

Example 2

```
{def $classes=fetch( 'class', 'list', hash( 'class_filter', array( 'folder',
'article' ) ) )}

{foreach $classes as $class}
    {$class.name|wash} <br />
{/foreach}
```

Outputs the names of the classes identified by the strings "folder" and "article".

Example 3

```
{def $classes=fetch( 'class', 'list',
                     hash( 'class_filter', array( 1, 2, 3 ),
                           'sort_by', array( 'name', true() ) ) )}
```

The function will fetch classes number 1, 2 and 3. The fetched classes will be sorted by name in ascending order.

5

override_template_list

Summary

Fetches the override rules associated with a class.

Usage

```
fetch( 'class', 'override_template_list', hash( 'class_id', class_id ) )
```

Parameters

Name	Type	Description	Required
class_id	integer	The ID number of the target class.	Yes.

Returns

An array of hashes containing information about the override rules.

Description

This function fetches the override rules that are associated with the class specified by the "class_id" parameter. The function returns an array of hashes. Each element of the returned array contains the following structure:

Attribute	Type	Description
siteaccess	string	The siteaccess that the override belongs to.
block	string	The name of the override block.
source	string	The path to the original template.
target	string	The path to the override template.

Examples

Example 1

```
{def $overrides=fetch( 'class', 'override_template_list', hash( 'class_id', 13
) )}

{foreach $overrides as $override}
```

```
    {$override.target} - ({$override.source}) <br />

{/foreach}
```

Outputs information about the overrides for class number 13.

Views

classlist   (page 656)
    Provides an interface for generating a class overview for a class group.

copy   (page 657)
    Provides an interface for copying a role.

down   (page 658)
    Provides an interface for moving an attribute to a lower position.

edit   (page 659)
    Provides an interface for editing a class.

groupedit   (page 660)
    Provides an interface for editing a class group.

grouplist   (page 661)
    Provides an interface for generating an overview of the class groups.

removeclass   (page 662)
    Provides an interface to the class removal mechanism.

removegroup   (page 663)
    Provides an interface to the class group removal mechanism.

up   (page 664)
    Provides an interface for moving an attribute to a higher position.

view   (page 665)
    Provides an interface for viewing a class.

5

classlist

Summary

Provides an interface for generating a class overview for a class group.

copy

Summary

Provides an interface for copying a role.

5

down

Summary

Provides an interface for moving an attribute to a lower position.

5

edit

Summary

Provides an interface for editing a class.

5

groupedit

Summary

Provides an interface for editing a class group.

5

grouplist

Summary

Provides an interface for generating an overview of the class groups.

removeclass

Summary

Provides an interface to the class removal mechanism.

5

removegroup

Summary

Provides an interface to the class group removal mechanism.

5

up

Summary

Provides an interface for moving an attribute to a higher position.

5

view

Summary

Provides an interface for viewing a class.

5.4.2 collaboration

Summary

Provides an interface to the collaboration engine.

Description

This module provides an interface to the collaboration engine inside the eZ publish kernel. The administration interface makes use of the views that the module provides in order to allow the management of collaboration items. Although possible, it isn't common to use these views when building a website (unless there is a need to replicate the collaboration management functionality of the administration interface).

The module components are documented in the following sections:

- Fetch functions (page 667)
- Views (page 674)

5

Fetch functions

\begin{description} \item[group_tree]\refer{label61f22364de74826cc9acf0fe03b409ab}\ \\ Not documented yet.

\item[message_list]\refer{label8f14e9dba012a7c156df28e9ec7f73e6}\ \\ Not documented yet.

\item[participant]\refer{label8d410e74b78f31026a4aedb2506b6f11}\ \\ Not documented yet.

\item[participant_list]\refer{label7480d4b8ea57ae3130e5e729dba1b231}\ \\ Not documented yet.

\item[participant_map]\refer{label539a784ccc20b0416cd49ceb0e1b4317}\ \\ Not documented yet.

\item[tree_count]\refer{label2d33bfc7c085aea7dc61975fd971cfb7}\ \\ Not documented yet.

\end{description}

5

group_tree

Summary

Not documented yet.

5

message_list

Summary

Not documented yet.

5

participant

Summary

Not documented yet.

participant_list

Summary

Not documented yet.

5

participant_map

Summary

Not documented yet.

5

tree_count

Summary

Not documented yet.

5

Views

action   (page 675)
>   Not documented yet.

group   (page 676)
>   Not documented yet.

item   (page 677)
>   Not documented yet.

view   (page 678)
>   Not documented yet.

5

action

Summary

Not documented yet.

5

group

Summary

Not documented yet.

5

item

Summary

Not documented yet.

5

view

Summary

Not documented yet.

5

5.4.3   content

Summary

Provides views for managing content (nodes, objects, searching, etc.)

Description

This module provides an interface to the content engine that is built into eZ Publish. It is the most important and most commonly utilized module. It provides views that make it possible to display and edit the contents of objects, manage nodes in the content tree, searching, translation, etc. A typical eZ Publish site uses many of the views and the fetch functions that this module provides.

The module components are documented in the following sections:

\begin{itemize} \item{} Fetch functions\refer{label8cab185c4edb59ab1d3aa81c002218de} \item{} Views\refer{labelf42eea74ccbe27eee00fa7bd0683ea4f} \end{itemize}

5

Fetch functions

\begin{description}   \item[access]\refer{label147b38d55c85c3aef14b7c9665f2475c}\   \\
Checks if the current user has access to a given function.

\item[bookmarks]\refer{label33d4b7d2f0031310b0ef7462607f5ca3}\ \\ Fetches the book-
marks of the current user.

\item[calendar]\refer{label0aa039a14b05dd8c03baa4209855cfd5}\ \\ This function fetches
the date time list when objects were published.

\item[can_instantiate_classes]\refer{label06bce55ef549912dfe4c931eafd2c628}\ \\ Checks if
the current user is allowed to create nodes.

\item[can_instantiate_class_list]\refer{labele742ce6f29f7cff6a0b8a3fa05d7e3b7}\ \\ Fetches
the classes that the current user can create objects of.

\item[class]\refer{label420564b2a01da5bc656b18af450929b4}\ \\ Fetches a content class.

\item[class_attribute]\refer{labelc734a4bebf31f60e22b0ed104102fba6}\ \\ Fetches an at-
tribute of a content class.

\item[class_attribute_list]\refer{label85dbea64b66060bdf1c9a5325e9eb2b8}\ \\ Fetches the
attributes of a class.

\item[collected_info_collection]\refer{label17e6710b23ff2d08cf4e464bbc3a9a08}\   \\
Fetches an information collection.

\item[collected_info_count]\refer{label4da3921187a3adc05f1472f1227e8810}\ \\ Fetches
the number of collections that match a certain criteria.

\item[collected_info_count_list]\refer{label808c89b9680d447b27f792daf99c8411}\   \\
Fetches the number of times different values were collected.

\item[contentobject_attributes]\refer{label7ea49c1a76a7353683d885c917658b5f}\   \\
Fetches the attributes of an object's version (and translation).

\item[country_list]\refer{labelc6ac6951487e660575b20ea0c42b8407}\ \\ Fetches the list of
countries specified in the "country.ini" configuration file.

\item[draft_count]\refer{label0508a893746be1b1a0f1fbf4a51c03c6}\ \\ Fetches the number
of drafts that belong to the current user.

\item[draft_version_list]\refer{label33e0478635469c1e821632d0c14065d6}\ \\ Fetches the
drafts that belong to the current user.

\item[keyword]\refer{labelea6cdac18692262780659d3ed6e98a64}\ \\ Fetches nodes that use
keywords starting with a given sequence.

\item[keyword_count]\refer{labelce374a7ffa66b6064ed47cb7cdb320f0}\ \\ Fetches the num-
ber of nodes that use certain keywords. Only nodes that are main locations for content objects
are counted.

\item[list]\refer{label3b56d2188a71c68cb234d8a2f61934dc}\ \\ Fetches the children of a
node or a collection of nodes.

5

\item[list_count]\refer{label5d0a5d050dbe5a81ea74e7486cb82164}\ \\ Fetches the number of children of a node.

\item[locale]\refer{label21e9d876fc377c5ecdf3dd78b01deade}\ \\ Fetches the current or a specified locale.

\item[locale_list]\refer{labelf4284e541ceb6ebfdd91a65df1e1fdf4}\ \\ Fetches the available locales.

\item[navigation_part]\refer{label8b37537bbb76761d3b7dde3ceb4477c2}\ \\ Fetches information about a navigation part.

\item[navigation_parts]\refer{label9933f01cac254ac221df21bef9658fd3}\ \\ Fetches all available navigation parts.

\item[node]\refer{label04459bca03e3361ff50521c6045e530d}\ \\ Fetches a node (identified by either an ID number or a path).

\item[non_translation_list]\refer{labelb86b51e8ad1fd5d17804f7aaa8f10fda}\ \\ Fetches locales that a version of an object may be translated into.

\item[object]\refer{labele562d4d9f8786e5a07365f3a18f2fab0}\ \\ Fetches a content object (specified by an ID number).

\item[object_by_attribute]\refer{label6ff27ea036d7b86c6cb28e37d1d3c790}\ \\ DEPRECATED

\item[object_count_by_user_id]\refer{label6179912310c62223cc5ecbb31f2b895a}\ \\ Fetches the number of objects (of a class) created by a user.

\item[pending_count]\refer{labelb83bca59cb84eaa1baef2efa0a3cd829}\ \\ Fetches the number of pending objects for the current user.

\item[pending_list]\refer{label22067a92da0b44a31ccd19a63f14c270}\ \\ Fetches the pending objects for the current user.

\item[recent]\refer{label0a3fb86e1d5e854c95f245c14b97f112}\ \\ Fetches nodes where the current user recently published something.

\item[related_objects]\refer{label8ee82f24bff9c5bf8257dbed89aa353e}\ \\ Fetches related objects.

\item[related_objects_count]\refer{labelb088ceaadfe32d5517bebe57e5dfeb83}\ \\ Fetches the number of related objects.

\item[reverse_related_objects]\refer{label63898582c6c3c7bde9df08a008161eff}\ \\ Fetches reverse related objects.

\item[reverse_related_objects_count]\refer{label6a1e1e8bc7f25b59868dc03ef7209675}\ \\ Fetches the number of reverse related objects.

\item[same_classattribute_node]\refer{label4a1a26a636df94fcbba1ac4c53e9696c}\ \\ Fetches nodes containing attributes that match a certain value.

\item[search]\refer{label7260f1f1a7251d4ace02d800d4940af2}\ \\ Fetches nodes containing data that match a certain criteria.

5

\item[section_list]\refer{labelb4eac7b1fb703f6ca44489aa6b298782}\ \\ Fetches the available sections.

\item[tipafriend_top_list]\refer{labelfbc017d5bd3ca915bdf715a4605658b0}\ \\ Fetches the most popular (most tipped) nodes.

\item[translation_list]\refer{label405f348c2471c51d23328b2020678555}\ \\ Fetches the locales that can be used to translate objects.

\item[trash_count]\refer{label51afc5823b3f9e089ae23c465f4c335a}\ \\ Fetches the number of objects that are in the trash.

\item[trash_object_list]\refer{labeld9a91324ed06e225ea6756e2ae29ab81}\ \\ Fetches the objects that are in the trash.

\item[tree]\refer{labelab12a39dac2db76cd9d35c8086b6365c}\ \\ Fetches the children of a node recursively.

\item[tree_count]\refer{label673b78c56d1fab7d8e5766a66c7836d2}\ \\ Fetches the number of children of a node recursively.

\item[version]\refer{label740cf91c0748037402100785754aab33}\ \\ Fetches a specific version of an object.

\item[version_count]\refer{label0ebfc4e5506bf16e10847566180b6b72}\ \\ Fetches the number of versions of a content object.

\item[version_list]\refer{label0fb09d24f10701c4e725e9ca3376f648}\ \\ Fetches all the versions of a content object.

\item[view_top_list]\refer{label04639dd10cf9cd7e96e2acb9c73bb478}\ \\ Fetches the most popular (most viewed) nodes.

\end{description}

5

access

Summary

Checks if the current user has access to a given function.

Usage

```
fetch( 'content', 'access',
       hash( 'access',                 access,
             'contentobject',          contentobject,
           [ 'contentclass_id',         contentclass_id,     ]
           [ 'parent_contentclass_id', parent_contentclass_id ] ) )
```

Parameters

Name	Type	Description	Required
access	string	The desired access method (see below).	Yes.
contentobject	object	The target/location (either an object or a node).	Yes.
contentclass_id	integer, string	The ID number or identifier of the class that should be included in the check.	No.
parent_contentclass_id	integer, string	The parent node's class ID number or identifier that should be included in the check.	No.

Returns

TRUE if access is allowed, FALSE otherwise.

Description

This function makes it possible to find out if the current user has access (read, edit, create, remove, etc.) to a given content object or a content node. The optional parameters "contentclass_id" and "parent_content_class_id" can be used to finetune the checking. These parameters are compatible with both class ID numbers and class identifier strings. The function supports checking for the following access methods:

- bookmark

- create

- edit

5

- move

- read

- remove

- pdf (deprecated)

- restore

- translate

- versionread

When checking "create" access and the "contentclass_id" is not specified, the function will return TRUE as long as there is a create access for the given object. However, the user could still not be allowed to create a specific class.

Examples

Example 1

```
{def $test=fetch( 'content', 'access',
                  hash( 'access', 'read',
                        'contentobject', $node ) )}

{if $test}
    The current user has read access to the given node.
{else}
    The current user does not have read access to the given node.
{/if}
```

Checks if the content node represented by $node is can be read by the current user.

Example 2

```
{def $test=fetch( 'content', 'access',
                  hash( 'access', 'create',
                        'contentobject', $node,
                        'contentclass_id', 'folder' ) )}

{if $test}
    The current user can create a folder below the given node.
{else}
    The current user can not create a folder below the given node.
{/if}
```

Checks if the current user can create an instance of a given class below the specified node.

bookmarks

Summary

Fetches the bookmarks of the current user.

Usage

```
fetch( 'content', 'bookmarks', hash( [ 'offset', offset, ]
                                      [ 'limit',  limit   ] ) )
```

Parameters

Name	Type	Description	Required
offset	integer	Number of bookmarks to skip.	No.
limit	integer	Maximum number of bookmarks to fetch.	No.

Returns

An array of ezcontentbrowsebookmark (page 1020) objects.

Description

Fetches the bookmarks of the current user and returns an array of ezcontentbrowsebookmark (page 1020) objects. The resulting array starts with the most recently added bookmark.

Examples

Example 1

```
{def $bookmarks=fetch( 'content', 'bookmarks' )}

{foreach $bookmarks as $bookmark}

    <a href={$bookmark.node.url_alias|ezurl}>{$bookmark.name|wash}</a> <br />

{/foreach}
```

Outputs all the bookmarks (as links) for the current user.

5

Example 2

```
{def $bookmarks=fetch( 'content', 'bookmarks',
                       hash( 'limit', 5 ) )}

{foreach $bookmarks as $bookmark}

    <a href={$bookmark.node.url_alias|ezurl}>{$bookmark.name|wash}</a> <br />

{/foreach}
```

Outputs the five most recently added bookmarks (as links) for the current user.

5

calendar

Summary

This function fetches the date time list when objects were published.

Usage

```
fetch( 'content', 'calendar',
hash( 'parent_node_id',    parent_node_id,
        [ 'offset',                    offset, ]
  [ 'limit',                   limit, ]
    [ 'depth',            depth, ]
   [ 'depth_operator',            depth_operator,      ]
    [ 'class_id',                  class_id,           ]
  [ 'attribute_filter',          attribute_filter, ]
  [ 'extended_attribute_filter',  extended_attribute_filter, ]
  [ 'class_filter_type',        class_filter_type, ]
  [ 'class_filter_array',        class_filter_array, ]
  [ 'group_by',                  group_by,          ]
  [ 'main_node_only',    main_node_only, ]
      [ 'ignore_visibility',                  ignore_visibility,      ]
  [ 'limitation',           limitation        ] ) )
```

5

Parameters

Name	Type	Description	Required
parent_node_id	integer	The ID number of the parent node.	Yes.
offset	integer	The offset to start at.	No.
limit	integer	The number of date time that should be fetched.	No.
depth	integer	The maximum level of depth that should be explored (1 by default).	No.
depth_operator	string	The logic to use when checking the depth.('le' by default).	No.
class_id	integer	The class id of the objects to be fetched.	No.
attribute_filter	mixed	The attribute level filter logic.	No.
extended_attribute_filter	mixed	The extended attribute level filter logic.	No.
class_filter_type	string	The type of class filtering (include/exclude).	No.
class_filter_array	array	The classes that should be filtered.	No.
group_by	array	Deprecated	No.
main_node_only	boolean	Type of nodes that should be fetched (all or main nodes only).	No.
ignore_visibility	boolean	Makes it possible to fetch hidden nodes. (false by default)	No.
limitation	array	The limitations that one user has on the objects to be fetched	No.

Returns

An array of publish date time of objects or FALSE.

Description

This function fetches the publish date time of objects. The parent node must be specified using "parent_node_id" parameter.

The depth parameter: The depth parameter can be used to specify the level of depth (within the branch) that the function should explore when it is running. If the depth is set to one, this function will simply act as the list function. If the depth is greater than one, the function will fetch nodes further down in the branch. The default value is unlimited.

The depth operator: The depth operator can be set to either "lt", "le", "eq", "gt" or "ge" - meaning "less than", "less equal", "equal to", "greater than" and "greater equal". For example, if it is set to 'eq', only nodes with the depth that was specified using the depth parameter will be fetched.

5

Examples

Example 1

```
{def $nodes=fetch( 'content', 'calendar',
 hash( 'parent_node_id', 42 ) )}
    {foreach $nodes as $node}
     {$node.published|wash} <br />
 {/foreach}
```

The example outputs publish date time of all objects under node 42 in depth 1.

Example 2

```
{def $nodes=fetch( 'content', 'calendar',
 hash( 'parent_node_id', 42,
        'depth', 1,
        'depth_opeator', 'ge' ) )}
   {foreach $nodes as $node}
     {$node.published|wash} <br />
   {/foreach}
```

The example outputs publish date time of all object under node id 42.

5

can_instantiate_classes

Summary

Checks if the current user is allowed to create nodes.

Usage

```
fetch( 'content', 'can_instantiate_classes' [, hash( 'parent_node',
parent_node ) ] )
```

Parameters

Name	Type	Description	Required
parent_node	object	Node to check if user can create new object in	No.

Returns

TRUE if the current user is allowed to create nodes, FALSE otherwise.

Description

This fetch function checks if the current user is allowed to create nodes either below the current position within the tree or below a given node (specified by the "parent_node" parameter). The function will return TRUE if the current user is allowed to create nodes, otherwise FALSE will be returned.

Examples

Example 1

```
{if fetch( 'content', 'can_instantiate_classes' )}
    The current user can create nodes below the current node.
{else}
    The current user can not create nodes below the current node.
{/if}
```

Example 2

```
{def $target=fetch( 'content', 'node', hash( 'node_id', 64 )}

{if fetch( 'content', 'can_instantiate_classes', hash( 'parent_node', $target
)}
    The current user can create nodes below node number 64.
{else}
    The current user can not create nodes below node number 64.
{/if}
```

5

can_instantiate_class_list

Summary

Fetches the classes that the current user can create objects of.

Usage

```
fetch( 'content', 'can_instantiate_class_list', hash( [ 'group_id',
group_id    ],
                                            [ 'parent_node',
parent_node ] )
```

Parameters

Name	Type	Description	Required
group_id	integer	The ID number of a class group to fetch classes from.	No.
parent_node	object	Alternative parent node.	No.

Returns

Array of ezcontentclass (page 1022) objects or FALSE.

Description

This function fetches a list of classes that the current user is allowed to create objects from. If no parameters are given, the class list will be generated based on the current node and classes from all class groups. The "group_id" parameter can be used to instruct the system to only fetch classes from a certain class group. The "parent_node" parameter can be used to instruct the system to check which classes the current user is allowed to instantiate below a certain node instead of the current node. The function returns an array of ezcontentclass (page 1022) objects or FALSE / empty array if the current user can not create instances of any class.

Examples

Example 1

```
{def $classes=fetch( 'content', 'can_instantiate_class_list' )}

{foreach $classes as $class}
    {$class.name} <br />
{/foreach}
```

5

Outputs the name of all classes that the current user is allowed to create below the current node.

Example 2

```
{def $classes=fetch( 'content', 'can_instantiate_class_list', hash(
'group_id', 3 )}

{foreach $classes as $class}
    {$class.name} <br />
{/foreach}
```

Outputs the name of classes belonging to class group number 3 that the current user is allowed to create below the current node.

5

class

Summary

Fetches a content class.

Usage

```
fetch( 'content', 'class', hash( 'class_id', class_id ) )
```

Parameters

Name	Type	Description	Required
class_id	integer	The identifier or ID number of the of class that should be fetched.	Yes.

Returns

The specified class as an ezcontentclass (page 1022) object or FALSE.

Description

This function fetches the class specified by the "class_id" parameter and returns it as an ezcontentclass (page 1022) object. The function will return FALSE if the specified class is unavailable.

Examples

Example 1

```
{def $class=fetch( 'content', 'class', hash( 'class_id', 13 ) )}
{$class.name|wash}
```

Outputs name of class number 13.

5

class_attribute

Summary

Fetches an attribute of a content class.

Usage

```
fetch( 'content', 'class_attribute', hash( 'attribute_id', attribute_id,
                                    [ 'version_id',   version_id ] ) )
```

Parameters

Name	Type	Description	Required
attribute_id	integer	The ID number of the class attribute that should be fetched.	Yes.
version_id	integer	The status/version of the class.	No.

Returns

The specified class attribute as an ezcontentclassattribute object or FALSE.

Description

This function fetches the class attribute specified by the "attribute_id" parameter and returns it as an ezcontentclassattribute (page 1026) object. The function will return FALSE if the specified class attribute is unavailable. The optional "version_id" parameter can be used to fetch the specified class attribute from a certain version of a class. The following versions / status codes can be used:

- 0: defined

- 1: temporary

- 2: modified

Examples

Example 1

```
{def$class_attribute=fetch('content','class_attribute',hash('attribute_id',231))}
{$class_attribute.name|wash}
```

Outputs the name of class attribute number 231.

class_attribute_list

Summary

Fetches the attributes of a class.

Usage

```
fetch( 'content', 'class_attribute_list',
       hash( 'class_id', class_id,
           [ 'version_id', version_id ] ) )
```

Parameters

Name	Type	Description	Required
class_id	integer	The ID number of the target class.	Yes.
version_id	integer	The version/status of the class.	No.

Returns

Array of ezcontentclassattribute (page 1026) objects or FALSE.

Description

This function fetches the attributes of a class. The ID number of the target class must be specified using the "class_id" parameter. The function returns an array of ezcontentclassattribute (page 1026) objects or FALSE if the specified class attribute is unavailable. The optional "version_id" parameter can be used to fetch class attributes from a certain version of a class. The following versions / status codes can be used:

- 0: defined

- 1: temporary

- 2: modified

Examples

Example 1

```
{def $attributes=fetch( 'content', 'class_attribute_list', hash( 'class_id',
13 ) )}

{foreach $attributes as $attribute}
```

```
    {$attribute.name|wash} <br />
{/foreach}
```

Outputs the names of the attributes that belong to class number 13.

5

collected_info_collection

Summary

Fetches an information collection.

Usage

```
fetch( 'content', 'collected_info_collection',
       hash( 'collection_id',    collection_id,
             'contentobject_id', contentobject_id ) )
```

Parameters

Name	Type	Description	Required
collection_id	integer	The ID number of the collection that should be fetched.	Yes.
contentobject_id	integer	The ID number of the object that should be fetched.	Yes.

Returns

An ezinformationcollection (page 1071) object or FALSE.

Description

This function fetches an information collection. Both the ID number of the collection and the contentobject must be provided. The function returns an ezinformationcollection (page 1071) object or FALSE.

Examples

Example 1

```
{def $collection=fetch( 'content', 'collected_info_collection',
                        hash( 'collection_id',    123,
                              'contentobject_id', 456 ) )}

{foreach $collection.attributes as $attribute}
    {$attribute.contentclass_attribute_name} <br />
{/foreach}
```

Outputs the attributes for the 123rd information collection for object number 456.

collected_info_count

Summary

Fetches the number of collections that match a certain criteria.

Usage

```
fetch( 'content', 'collected_info_count',
     hash( [ 'object_attribute_id', object_attribute_id, ]
           [ 'value',              value,              ]
           [ 'object_id',          object_id,          ]
           [ 'creator_id',         creator_id, ]
           [ 'user_identifier',    user_identifier ] ) )
```

Parameters

Name	Type	Description	Required
object_attribute_id	integer	The ID number of the target object attribute.	No.
object_id	integer	The ID number of the target content object.	No.
value	integer	Value filtering on the attribute level.	No.
creator_id	integer	The creator of the information. Only used when object_id is used	No.
user_identifier	string	The user identification string generated by eZ Publish when the information was collected. Only used when object_id is used	No.

Returns

The number of collections (as an integer).

Description

This function counts the number of collections based on the provided parameters. An object's ID number (using the "object_id" parameter) or an object attribute's ID number (using the "object_attribute_id" parameter) must be specified. In addition, it is possible to filter out collections that match a certain value. This is typically useful when it comes to counting the number of times a specific value was submitted to a poll. If the "value" parameter is used then the "object_attribute_id" parameter must also be provided. The function returns a positive integer if the system is able to find collections matching the given parameters; if not, zero will be returned.

5

Note: 'value' is only valid with 'object_attribute_id'; 'creator_id', 'user_identifier' are only valid with 'object_id'. If 'object_attribute_id' and 'object_id' are provided, only 'object_attribute_id' will be used.

Examples

Example 1

```
{def $collections=fetch( 'content', 'collected_info_count',
                         hash( 'object_attribute_id', 42,
                               'value',               1 ) )}
{$collections}
```

Outputs the number of collected information for object with attribute id #42 and value 1.

Example 2

```
{def $collections=fetch( 'content', 'collected_info_count',
                         hash( 'object_id', 42,
                               'creator_id',20 ) )}
{$collections}
```

Outputs the number of collected information for object with id #42 and information creator id #20.

Example 3

```
def $collections=fetch( 'content', 'collected_info_count',
                         hash( 'object_id', 42,
                               'user_identifier',
'66v8qb7jp7ih9c8d3pmjl25fe3' ) )}
{$collections}
```

Outputs the number of collected information for object with id #42 and information user's identifier '66v8qb7jp7ih9c8d3pmjl25fe3'.

5

collected_info_count_list

Summary

Fetches the number of times different values were collected.

Usage

```
fetch( 'content', 'collected_info_count_list', hash( 'object_attribute_id',
object_attribute_id ) )
```

Parameters

Name	Type	Description	Required
object_attribute_id	integer	The ID number of the target object attribute.	Yes.

Returns

An array of integers representing a count for every value.

Description

This function calculates and returns the sum of collected values (integers). It was developed for the poll feature.

Examples

Example 1

```
{def $counts=fetch( 'content', 'collected_info_count_list',
                    hash( 'object_attribute_id', 1024 ) )}

{foreach $counts as $count}
    {$count} <br />
{/foreach}
```

Outputs the number of times different values were collected.

5

contentobject_attributes

Summary

Fetches the attributes of an object's version (and translation).

Usage

```
fetch( 'content', 'contentobject_attributes',
       hash( 'version',        version,
           [ 'language_code', language_code ] ) )
```

Parameters

Name	Type	Description	Required
version	object	The target version (must be an ezcontentobjectversion object).	Yes.
language_code	string	The language code.	No.

Returns

An array of ezcontentobjectattribute (page 1042) objects or FALSE.

Description

This function fetches the attributes that belong to a certain version. The version must be provided (as an ezcontentobjectversion (page 1056)) using the "version" parameter. The "language_code" parameter is optional and can be used to get the attributes that belong to a specific translation. The function returns an array of ezcontentobjectattribute (page 1042) objects or FALSE if something went wrong.

Examples

Example 1

```
{def $object=fetch( 'content', 'object', hash( 'object_id', 14 ) )
     $attributes=fetch( 'content, 'contentobject_attributes',
                        hash( 'version', $object.current ) )}

{foreach $attributes as $attribute}
    {$attribute.data_type_string} <br />
{/foreach}
```

5

Outputs the names of the datatypes that are used by the different attributes within the current version of object number 14.

5

country_list

Summary

Fetches the list of countries specified in the "country.ini" configuration file.

Usage

```
fetch( 'content', 'country_list',
       hash( [ 'filter', filter, ]
             [ 'value', value    ]  ) )
```

Parameters

Name	Type	Description	Required
filter	string	The type of filtering.	No.
value	string	The value of the filter.	No.

Returns

An array of hashes (see below) or FALSE.

Description

This function returns a structure that contains information about countries which are mentioned in the "country.ini" configuration file. The optional "filter" parameter sets the type of filtering that will be used when the countries are fetched. It can be set to "Name", "Alpha2", "Alpha3" or "IDC". The "value" parameter can be used to set the filter's value. For example, it could be set to "Norway".

Note that the names of the countries are fetched according to the current locale settings.

The function returns an array of hashes. The hashes consist of the following elements:

Key	Type	Description
Name	string	The name of the country (for example €Norway€).
Alpha2	string	The ISO 3166-1 alpha-2 code of the country (for example €NO€).
Alpha3	string	The ISO 3166-1 alpha-3 code of the country (for example €NOR€).
IDC	string	The international dialing code of the country (for example €47€).

5

Examples

Example 1

```
{def $all_countries=fetch( 'content', 'country_list' )}

{foreach $all_countries as $country}
    {$country.Name} <br />
{/foreach}
```

Outputs the names of all countries specified in the "country.ini" configuration file or its override.

Example 2

```
{def $country=fetch( 'content', 'country_list', hash( 'filter', 'Name',
'value', 'Norway' ) )}

{$country.Name} : {$country.IDC}
```

Outputs the country name and international dialing code for Norway.

5

draft_count

Summary

Fetches the number of drafts that belong to the current user.

Usage

```
fetch( 'content', 'draft_count' )
```

Returns

The number of drafts (as an integer) that belong to the current user.

Description

This function fetches the number of drafts that belong to the current user and returns it as an integer.

Examples

Example 1

```
{def $drafts=fetch( 'content', 'draft_count' )}
The current user has {$drafts} drafts.
```

Outputs the number of drafts that belong to the current user.

5

draft_version_list

Summary

Fetches the drafts that belong to the current user.

Usage

```
fetch( 'content', 'draft_version_list',
       hash( [ 'offset', offset, ]
             [ 'limit',  limit  ] ) )
```

Parameters

Name	Type	Description	Required
offset	integer	The offset to start at.	No.
limit	integer	The number of drafts/versions that should be fetched.	No.

Returns

An array of ezcontentobjectversion (page 1056) objects or FALSE.

Description

This function fetches the drafts that belong to the current user. The optional parameters "offset" and "limit" can be used to limit the result. The function returns an array of ezcontentobjectversion (page 1056) objects. If no drafts can be found or if something goes wrong, the function returns FALSE.

Examples

Example 1

```
{def $drafts=fetch( 'content', 'draft_version_list' )}

{foreach $drafts as $draft}
    {$draft.id}: {$draft.name} <br />
{/foreach}
```

Outputs the ID numbers and the names of all drafts that belong to the current user.

5

keyword

Summary

Fetches nodes that use keywords starting with a given sequence.

Usage

```
fetch( 'content',  'keyword',
 hash( 'alphabet',           alphabet,
     [ 'strict_matching',    strict_matching,    ]
     [ 'classid',            classid,            ]
     [ 'offset',             offset,             ]
     [ 'limit',              limit,              ]
     [ 'owner',              owner,              ]
     [ 'parent_node_id',     parent_node_id,     ]
     [ 'include_duplicates', include_duplicates, ]
     [ 'sort_by',            sort_by             ] ) )
```

Parameters

Name	Type	Description	Required
alphabet	string	The sequence that should be matched.	Yes.
strict_matching	boolean	Enables or disables exact matching. If FALSE (or omitted), the function will look for keywords that start with the specified sequence.	No.
classid	array	Filtering: the ID number of the class or an array of the ID numbers.	No.
offset	integer	The offset to start at.	No.
limit	integer	The number of elements that should be returned.	No.
owner	integer	Filtering by owner: the ID number of the object representing the user.	No.
parent_node_id	integer	The ID number of the parent node.	No.
include_duplicates	boolean	Makes it possible to avoid duplicates (different keywords but same nodes) in the result. If TRUE (or omitted), duplicates are allowed.	No.
sort_by	array	The sorting mechanism that should be used.	No.

Returns

An array of hashes (see below) or FALSE.

Description

This function fetches nodes that encapsulate objects which make use of certain keywords. It returns an array of hashes where each hash represents a "keyword-node" pair.

The keyword must be provided using the "alphabet" parameter. This parameter can be a letter, a part of a word or an entire word - the function will look for keywords that start with the specified sequence. This means that if you specify "car" in the "alphabet" parameter, the system will look for keywords like "car", "card", "carnival", "car painting" and so on (but not "scarf" for example). Every time such a keyword is found, the main node of the object that uses this keyword will be included in the result along with the matching keyword. Note that if an object contains multiple keywords that start from the specified sequence, the main node of this object will be fetched several times. The following table shows an example of the structure that may be returned if there are three articles that contain matching keywords:

Keyword	Node
car enamel	The main node of the "Auto" article (as a ez-contentobjecttreenode (page 1051) object).
car painting	The main node of the "Auto" article (as a ez-contentobjecttreenode (page 1051) object).
car sickness	The main node of the "Driving" article (as a ezcontentobjecttreenode (page 1051) object).
cars	The main node of the "Traffic" article (as a ez-contentobjecttreenode (page 1051) object).

As you can see from the table, the article called "Auto" uses two keywords that start from "car": "car enamel" and "car painting". The remaining two articles ("Driving" and "Traffic") make use of one keyword each: "car sickness" and "cars".

The "strict_matching" parameter is optional and can be set to either true() or false(). When set to true(), the fetch function will look for the exactly same keyword as the specified sequence. For example, if the "alphabet" parameter is set to "car", only items that use "car" as a keyword will be fetched; keywords like "card", "carnival", "car painting" and so on will be ignored.

By default, the function will fetch nodes that encapsulate objects of all types created by any user. However, it is possible to fetch objects of a certain type or that were created by a specific user, by using the optional "classid" and "owner" parameters. The "offset" and "limit" parameters can be used to limit the result.

The optional "parent_node_id" parameter makes it possible to fetch nodes that are directly under the specified parent node. If an object makes use of a keyword that starts with the specified sequence, the function will check if the main node of this object is a child on the specified parent node. If yes, it will be included in the result.

The "include_duplicates" parameter makes it possible to avoid duplicates (different keywords but same nodes) in the result. This parameter is optional and can be set to either true() or false(). When set to false(), the result will only contain unique nodes. For example, if an article contains both "car enamel" and "car painting" keywords, and you specify "car" in the "alphabet" parameter, the main node of this article will be included in the result once. The corresponding keyword-node pair will contain the original sequence ("car") instead of the actual keyword that was matched

("car enamel" or "car painting"). The following table shows the structure that would be returned if the "include_duplicates" parameter is set to false() in the three-articles example above:

Keyword	Node
car	The main node of the "Auto" article (as a ez-contentobjecttreenode (page 1051) object).
car sickness	The main node of the "Driving" article (as a ezcontentobjecttreenode (page 1051) object).
cars	The main node of the "Traffic" article (as a ez-contentobjecttreenode (page 1051) object).

The "sort_by" parameter makes it possible to sort the result in different ways. This parameter must be provided as an array. The first element of the array must be the desired sorting method. The second element of the array must be the sorting direction, it can be either true() or false() - ascending or descending. Note that this parameter works in the very same way as the "sort_by" parameter of the list (page 714) fetch function. However, it currently only supports the following sorting methods:

- attribute

- class_identifier

- class_name

- modified

- name

- published

- section

Note that using other sort methods will lead to an error.

The function returns an array of hashes. Each hash consists of the following elements:

Key	Type	Description
keyword	string	The keyword that was matched.
link_object	object	The node (as a ezcontentobjecttreenode (page 1051) object) that encapsulates an object which uses the matched keyword.

If no match is found, the function will return FALSE.

5

Examples

Example 1

```
{def $list=fetch( 'content', 'keyword',
                  hash( alphabet, 'computer',
                        classid,   3 ) )}

{foreach $list as $element}
    {$element.link_object.name|wash} ({$element.keyword|wash}) <br />
{/foreach}
```

Outputs the names of nodes that encapsulate objects which make use of keywords starting with the string "computer". In addition, the matched keywords are also printed.

Example 2

```
{def $list=fetch( 'content', 'keyword',
                  hash( alphabet, 'computer',
                        classid, array( 1, 3 ) ) )}
```

Only nodes that encapsulate objects of the specified two classes will be fetched.

Example 3

```
{def $list=fetch( 'content', 'keyword',
                  hash( alphabet, 'computer',
                        classid, array( 1, 3 ),
                        owner, 14,
                        sort_by, array( 'modified', false() ) ) )}
```

Only the nodes/objects that are created by the user that has ID number 14 will be included in the result. The nodes will be sorted by the modification time of the objects; the node that encapsulates the most recently modified object will be the first element in the collection.

5

keyword_count

Summary

Fetches the number of nodes that use certain keywords. Only nodes that are main locations for content objects are counted.

Usage

```
fetch( 'content', 'keyword_count',
 hash( 'alphabet',           alphabet,
     [ 'strict_matching',    strict_matching,    ]
     [ 'classid',            classid,            ]
     [ 'owner',              owner,              ]
     [ 'parent_node_id',     parent_node_id,     ]
     [ 'include_duplicates', include_duplicates  ] ) )
```

Parameters

Name	Type	Description	Required
alphabet	string	The sequence that should be matched.	Yes.
strict_matching	boolean	Enables or disables exact matching. If FALSE (or omitted), the function will look for keywords that start with the specified sequence.	No.
classid	integer	Filtering: the ID number of the class.	No.
owner	integer	Filtering by owner: the ID number of the object representing the user.	No.
parent_node_id	integer	The ID number of the parent node.	No.
include_duplicates	boolean	Makes it possible to avoid counting duplicates (different keywords but same nodes). If TRUE (or omitted), duplicates are allowed.	No.

Returns

The number of matching nodes (as an integer).

Description

This fetch function operates in almost the same way as the "keyword" fetch function. However, instead of returning the actual nodes along with the matching keywords, it returns the count of the nodes. The "keyword_count" function takes the same parameters as the "keyword" function

with some exceptions (for example sorting is not supported). Refer to the documentation of the "keyword (page 708)" function for a detailed description of the parameters.

Examples

Example 1

```
{def $count=fetch( 'content', 'keyword_count',
                   hash( 'alphabet', 'computer' ) )}

There are {$count} number of nodes using keywords starting with "computer".
```

Outputs the number of nodes that encapsulate objects which use keywords starting with "computer".

5

list

Summary

Fetches the children of a node or a collection of nodes.

Usage

```
fetch( 'content', 'list',
 hash( 'parent_node_id',            parent_node_id,
     [ 'sort_by',                   sort_by,                     ]
     [ 'offset',                    offset,                      ]
     [ 'limit',                     limit,                       ]
     [ 'attribute_filter',          attribute_filter,            ]
     [ 'extended_attribute_filter', extended_attribute_filter, ]
     [ 'class_filter_type',         class_filter_type,           ]
     [ 'class_filter_array',        class_filter_array,          ]
     [ 'only_translated',           only_translated,             ]
     [ 'language',                  language,                    ]
     [ 'main_node_only',            main_node_only,              ]
     [ 'as_object',                 as_object,                   ]
     [ 'depth',                     depth,                       ]
     [ 'limitation',                limitation,                ]
     [ 'ignore_visibility',         ignore_visibility,         ]
     [ 'load_data_map',         load_data_map,          ]
     [ 'objectname_filter',         objectname_filter,         ] ) )
```

5

Parameters

Name	Type	Description	Required
parent_node_id	mixed	The ID number(s) of the parent node(s).	Yes.
sort_by	array	The sorting mechanism that should be used.	No.
offset	integer	The offset to start at.	No.
limit	integer	The number of nodes that should be fetched.	No.
attribute_filter	mixed	The attribute level filter logic.	No.
extended_attribute_filter	mixed	The extended attribute level filter logic.	No.
class_filter_type	string	The type of class filtering (include/exclude).	No.
class_filter_array	array	The classes that should be filtered.	No.
group_by	array	DEPRECATED	No.
only_translated	boolean	Translation filtering (on/off). DEPRECATED	No.
language	string	The translations that should be fetched.	No.
main_node_only	boolean	Type of nodes that should be fetched (all or main nodes only).	No.
as_object	boolean	If TRUE (or omitted), an array of "ezcontentobjecttreenode" objects will be fetched. Otherwise, an array of arrays will be returned.	No.
depth	integer	The maximum level of depth that should be explored (1 by default).	No.
limitation	array	Limitation array (emtpy array = access override).	No.
ignore_visibility	boolean	Makes it possible to fetch hidden nodes.	No.
load_data_map	boolean	Forces preloading of data map when fetch is performed. Default is on if fetch returns less than 15 items.	No.
objectname_filter	string	searches object names, which start with the given string: 'car' finds objects which names start with car and cars. 'others' finds objects which don't start with a letter.	No.

Returns

An array of ezcontentobjecttreenode (page 1051) objects or FALSE.

Description

This function fetches the children of a single node or a collection of nodes. The parent node(s) must be specified using the "parent_node_id" parameter. This parameter can either be the ID number of a single node or an array containing several node ID numbers. The function will fetch the nodes that are directly under the specified parent node(s). The collection is returned as an array of ezcontentobjecttreenode (page 1051) objects. If no nodes are found or if an error occurs, the function will return FALSE. The optional "group_by" parameter (grouping on a date/time basis) is deprecated since this functionality is not supported by Oracle and PostgreSQL.

Sorting

The "sort_by" parameter makes it possible to sort the result in different ways. This parameter must be provided as an array. The first element of the array must be the desired sorting method. The second element of the array must be the sorting direction, it can be either true() or false() - ascending or descending. The following table shows the sorting methods that can be used.

Sorting method	Description
attribute	The nodes are sorted according to the value of a specific attribute.
class_identifier	The nodes are sorted by the class identifiers of the objects.
class_name	The nodes are sorted by the class names of the objects.
depth	The nodes are sorted by their depth in the content tree.
modified	The nodes are sorted by the modification time of the objects.
name	The nodes are sorted by the names of the objects.
path	The nodes are sorted by their node ID path strings (/1/2/43/56).
path_string	The nodes are sorted by their virtual path strings (/company/about).
priority	The nodes are sorted by their priority.
published	The nodes are sorted by the creation time of the objects.
section	The nodes are sorted by the section IDs of the objects.

It is possible to combine different sorting methods. For example, it is possible to sort the nodes alphabetically and by their publish date/time at the same time.

Sorting on the attribute level is supported for the following datatypes:

- Checkbox (page 518)

5

- Country (page 520)

- Date (page 522)

- Date and time (page 524)

- E-mail (page 526)

- Identifier (page 534)

- Integer (page 541)

- Multi-price (page 570) (only supported in sort_by in content/list and content/tree, not for content/search)

- Object relation (page 572) (on related object's ID)

- Price (page 582)

- Product category (page 584) (category ID)

- Selection (page 588) (will not work when used as multiple selector)

- Text line (page 593)

- Time (page 595)

The syntax for doing attribute level sorting is almost the same as for normal sorting. The only difference is that the ID number or the identifier of the target attribute must be specified. If the identifier is used then both the identifier of the class and the attribute must be specified (separated by a slash, like this: "my_class/my_attribute"). Attribute sorting can only be used if the returned collection contains the same type of nodes.

Fetching subsets

By making use of the "offset" and "limit" parameters, it is possible to fetch only a subset of the collection that would have been returned if these parameters were omitted. While the "offset" parameter defines the start of the subset, the "limit" parameter defines the length (number of elements/nodes) of the subset. These parameters are processed at the final stage. In other words, it is possible to do advanced sorting/filtering and grab only a specific chunk of from the sorted/filtered/etc. collection.

Class filtering

The class filter mechanism is controlled by the "class_filter_type" and "class_filter_array" parameters. The "class_filter_type" parameter tells eZ Publish to include or exclude specific node types. This parameter must be either "include" or "exclude". It is not possible to include one set and to exclude another at the same time. The value of the "class_filter_array" parameter specifies the type of nodes that should be included or excluded. This parameter must be an array of class ID numbers or class identifier strings.

Attribute filtering

The attribute filter mechanism is controlled by the "attribute_filter" parameter. Attribute filtering makes it possible to fetch a set of nodes where an attribute (or several attributes) contains some specific data. Filtering on the attribute level is supported for the following datatypes:

- Checkbox

- Date

- Date and time

- E-mail

- Integer

- Object relation

- Selection (will not work when used as multiple selector)

- Text line

- Time

The "attribute_filter" parameter must be an array. The first element may be set to either "and" or "or" - this controls how the matching specified in the upcoming elements should be carried out. If this parameter is omitted, the system will default to "and". The rest of the elements are arrays, each array specifies a match. The elements in this array are:

1. Attribute ID

2. Match type

3. Match value

The attribute ID can be specified either as the ID number of the attribute or as a string that contains the identifier of the class and the identifier of the attribute separated by a slash. (Note that it is also possible to specify one of the pre-defined strings in order to do filtering on a non-attribute.) The ID number of the attribute or the identifier of the class and the attribute can be found when viewing/editing content classes.

The match type tells how eZ Publish should try to match the values. The match value is the data that should be matched. The following match types can be used:

Type	Description
=	Equal
!=	Not equal
<	Less than
>	Greater than
<=	Less than or equal to
>=	Greater than or equal to
in	Matches one of the array values
not in	Does not match any of the array values
between	Is a part of the range determined by the start and end values
not between	Is outside the range determined by the start and end values
like	Matches the wildcard string (the wildcard character is '*')

Note: the state attribute filter only support =, !=, in, and not_in filter type. It is not possible to filter on attributes of different classes, for example it is not possible to filter on both "article/ show_comments" and "folder/show_comments" in the same filter.

Filtering on a non-attribute

Instead of the class/attribute syntax, you can use "published" and "priority" as "AttributeID". It is also possible to use "owner" as "AttributeID" which refers to the owner's identifier of the content object. This can for instance be used to find all objects of a given user.

There are also some other pre-defined strings that can be used as "AttributeID" (refer to the following table). When one of these strings is used as "AttributeID", filtering is based on template object attributes (not on content object (page 111) attributes). This means that if you use for example "priority" as "AttributeID", the system will check the value of the "priority" attribute of the "ezcontentobjecttreenode (page 1051)" objects that represent information about content nodes.

The following table describes the pre-defined strings and corresponding attributes.

Name	Description
class identifier	The identifier of the class which the object encapsulated by the node is an instance of.
class name	The name of the class which the object encapsulated by the node is an instance of.
depth	The depth of the node within the content node tree.
modified	A UNIX timestamp pinpointing the exact date/ time when the object encapsulated by the node was last modified (the "modified" attribute of the corresponding "ezcontentobject"

	object).
modified_subnode	A UNIX timestamp pinpointing the exact time a sub node was changed.
name	The name of the object the node encapsulates.
owner	The ID number of the object representing the user who initially created the object (the "owner_id" attribute of the corresponding "ezcontentobject" object).
path	The node's path string (the "path_string" attribute).
priority	The node's priority.
published	A UNIX timestamp pinpointing the exact date/ time when the object was published for the first time (the "published" attribute of the corresponding "ezcontentobject" object).
section	The ID number of the section that the object belongs to (the "section_id" attribute of the corresponding "ezcontentobject" object).
state	This parameter allows to filter on a content object state id or an array of content object states ids. This parameter only supports =, !=, in and not_in operator in the attribute filter. With = and != operators, it's possible to filter on a numeric id while in and not_in are dedicated to filter an array of id.

Extended attribute filtering

The "extended_attribute_filter" parameter can be used to do advanced attribute filtering. It is for expert/experienced users. This mechanism makes it possible to introduce additional SQL conditions within the fetch. It allows filtering based on values in custom tables and/or special filtering within the "*ezcontentobject_attribute*" table.

When an extended attribute filter adds a new column in the result set from the database, it is possible to use it for sorting, by specifying new_table.column in the sort_by parameter of the fetch.

The mechanism must be configured using a configuration override for "*extendedattributefilter.ini*". This file allows the site administrator to set up different custom filters. Each filter must be a named collection of configuration settings. The name of a filter is the name of the configuration block under which the filter's settings are defined. The following text shows an example that demonstrates the setup of a filter called "MyFilter".

```
#The name of the filter.
[MyFilter]

#The name of the extension where the filtering code is defined.
ExtensionName=myextension
```

5

```
#The name of the filter class.
ClassName=eZMyExtendedFilter

#The name of the method which is called to generate the SQL parts.
MethodName=createSqlParts

#The file which should be included (extension/
myextension will automatically be prepended).
FileName=classes/ezmyextendedfilter.php
```

The function defined by the "MethodName" setting must return an associative array containing two strings:

```
array( 'tables' => '...', 'joins' => '...' , 'columns' => '...' ));
```

The contents of "tables" must start with a comma. The rest of the string should contain a comma separated list of other tables that should be included in the query. The contents of "joins" will be added inside the "WHERE" section of the query. This string must start with a space and contain an "AND" and an additional space at the end.

The example below shows a solution that filters the content based on values within an additional/custom table. This table is joined with the node table (by object ID and object version). The additional/custom table would most likely be maintained by a special/custom datatype.

```
class eZMyExtendedFilter
{
    /*!
     Constructor
    */
    function eZMyExtendedFilter()
    {
        // Empty...
    }

    function createSqlParts( $params )
    {
        $sqlTables= ', ezmyfiltertable ';

        $sqlJoins = ' ezcontentobject_tree.contentobject_id =
ezmyfiltertable.contentobject_id AND
ezcontentobject_tree.contentobject_version = ezmyfiltertable.version AND ';


        if ( isset( $params['value1'] ) )
        {
```

```
            $value1 = $params['value1'];
        }
        else
        {
            $value1 = "10";
        }
        if ( isset( $params['value2'] ) )
        {
            $value2 = $params['value2'];
        }
        else
        {
            $value2 = "10";
        }

        $sqlCondArray = array();

        $sqlCondArray[] = 'ezmyfiltertable.my_cond1 = ' . $value1;

        $sqlCondArray[] = 'ezmyfiltertable.my_cond1 = ' . $value2;

        $sqlCond = implode( ' or ', $sqlCondArray );

        $sqlCond = ' ( ' . $sqlCond . ' ) AND ' . $sqlJoins . ' ';

        return array( 'tables' => $sqlTables, 'joins' => $sqlCond, 'columns'
=>'' );


    }
}
```

The following template code shows how the extended attribute filter (see the PHP code above)
can be used within a fetch.

```
{fetch( 'content', 'list',
        hash( 'parent_node_id', 2,
              'sort_by', array( 'priority', false() ),
              'limit', 15,
              'extended_attribute_filter',
                          hash( 'id', 'MyFilter',
                                       'params', hash( 'value1', 15,
                                                       'value2', 30 ) ),
              'depth', 10,
              'main_node_only', true() ) )}
```

There is an additional example at the bottom of this page: Example 20.

Fetching main nodes only

The "main_node_only" parameter can be used to tell eZ Publish that it should only fetch main nodes (nodes that are main locations for content objects). This parameter is optional and can be set to either true() or false(). When set to true(), the fetch function will only return main nodes. This functionality can be used to avoid duplicates (different nodes but same objects) in the result.

Fetching translated objects only

Note: This description applies to eZ Publish 3.8 or earlier. The "only_translated" parameter is deprecated in all later versions. In some multi-language scenarios, not all objects are translated into other/all languages. When the "only_translated" parameter is set to "true()" the system will only fetch objects that have been translated into the language used by the siteaccess (the current language). The "language" parameter can be used to control which translations that should be fetched. The value of this parameter must be a valid locale string, for example "ger-DE".

Access override

The "limitation" parameter makes it possible to instruct the system to use an alternate set of access limitations instead of the ones that belong to the current user. This is typically useful when there is a need to fetch nodes that the current user does not have access to. All permission checking can be skipped by providing an empty array. It is also possible to provide arrays that dictate the access limitations of other users and/or virtual sets of limitations. The array follows an internal structure that will be documented in the future.

Visibility override

The "ignore_visibility" parameter makes it possible to fetch hidden nodes. It can be set to either "true()" or "false()". If set to "true()", the fetch will ignore the visibility flags of nodes and thus it will return all nodes regardless of their visibility status. In other words, this parameter overrides the "ShowHiddenNodes" (page 2515) configuration directive for a specific fetch operation in a template.

Examples

Example 1

```
{def $nodes=fetch( 'content', 'list',
        hash( 'parent_node_id', 42,
              'depth', 3 ) )}
```

```
{foreach $nodes as $node}
    {$node.name|wash} <br />
{/foreach}
```

This example demonstrates how to fetch nodes which are until the third level under the parent node number 42. Outputs the names of all these nodes.

Example 2

```
{fetch( 'content', 'list',
        hash( 'parent_node_id', 42,
              'sort_by',        array( 'published', false() ) ) )}
```

This example demonstrates how to fetch all nodes that are the children of node number 42. The nodes are sorted by the time they were created; the most recently created node will be the first element in the collection.

Example 3

```
{fetch( 'content', 'list',
        hash( 'parent_node_id', array( '42', '43', '45' ),
              'sort_by',        array( 'published', false() ) ) )}
```

This example demonstrates how to fetch all nodes that are the children of multiple nodes; number 42, 43 and 45. The nodes are sorted by the time they were created; the most recently created node will be the first element in the collection.

Example 4

```
{fetch( 'content', 'list',
        hash( 'parent_node_id', 42,
              'sort_by',        array( array( 'name',      false() ),
                                       array( 'published', false() ) ) ) )}
```

This example demonstrates how to combine different sorting methods. The fetch function will fetch the child nodes of node number 42. The fetched nodes will be sorted by their name and the time they were created.

Example 5

```
{fetch( 'content', 'list',
        hash( 'parent_node_id', 42,
              'sort_by',        array( 'attribute',
                                       false(),
                                       107 ) ) )}
```

This example demonstrates attribute sorting by using the ID number of the attribute.

Example 6

```
{fetch( 'content', 'list',
        hash( 'parent_node_id', 42,
              'sort_by',        array( 'attribute',
                                       false(),
                                       'person/last_name' ) ) )}
```

This example demonstrates attribute sorting by using the string notation to pinpoint the identifier
of the class and the attribute.

Example 7

```
{fetch( 'content', 'list',
        hash( 'parent_node_id',     42,
              'class_filter_type',  'include',
              'class_filter_array', array( '13' ) ) )}
```

This example demonstrates how to use class filtering to fetch only nodes that reference objects
that are instances of class number 13.

Example 8

```
{fetch( 'content', 'list',
        hash( 'parent_node_id',     42,
              'class_filter_type',  'include',
              'class_filter_array', array( 'folder' ) ) )}
```

This example demonstrates how to use class filtering to fetch only nodes that reference folder
objects.

Example 9

5

```
{fetch( 'content', 'list',
        hash( 'parent_node_id',     42,
              'class_filter_type',  'exclude',
              'class_filter_array', array( 'article',
                                           'comment' ) ) )}
```

This example demonstrates how to use class filtering to exclude articles and comments from a fetch.

Example 10

```
{fetch( 'content', 'list',
        hash( 'parent_node_id',   42,
              'attribute_filter', array( 'or',
                                         array( 152, '=', 'abc' ),
                                         array( 153, '=', '42' ) ) ) )}
```

This example demonstrates how to do attribute filtering. Only nodes that have attributes number 152 and 153 set to "abc" and 42 respectively will be included in the result.

Example 11

```
{fetch( 'content',
        'list',
        hash( 'parent_node_id',   42,
              'attribute_filter', array( array( 'article/title',
                                                'like',
                                                '*story*' ) ) ) )}
```

This example demonstrates how to do attribute filtering. Instead of specifying the ID number of the attribute (as in the previous example), the identifier of the class and the attribute is used. Only article nodes that contain the text "story" in their title attribute will be included in the result.

Example 12

```
{fetch( 'content',
        'list',
        hash( 'parent_node_id',   42,
              'attribute_filter', array( array( 'article/image',
                                                '=',
                                                87 ) ) ) )}
```

This example demonstrates how to do attribute filtering using an attribute of the "Object relation" datatype. Only article nodes that store relation to the specified content object (object ID = 87) in their "image" attribute will be included in the result.

Example 13

```
{fetch( 'content',
        'list',
        hash( 'parent_node_id',   42,
              'attribute_filter', array( array( 'review/rating',
                                                'between',
                                                array( 0, 2 ) ) ) ) )}
```

This example demonstrates how to do attribute filtering using an attribute of the "Selection" datatype. Only review nodes that contain "Very good", "Good" or "Ok" (options 0, 1 or 2) in their "rating" attribute will be included in the result.

Example 14

```
{fetch( 'content',
        'list',
        hash( 'parent_node_id',   42,
              'attribute_filter', array( array( 'bug/priority',
                                                'in',
                                                array( 1, 4 ) ) ) ) )}
```

Fetches bugs with priority 1 and 4 (i.e. bug nodes that have their priority attribute set to 1 or 4).

Example 15

```
{fetch( 'content',
        'list',
        hash( 'parent_node_id',   42,
              'attribute_filter', array( array( 'bug/priority',
                                                'between',
                                                array( 2, 4 ) ) ) ) )}
```

Fetches bugs with priority 2, 3 and 4 (i.e. bug nodes that have their priority attribute value between 2 and 4).

Example 16

```
{fetch( 'content', 'list',
        hash( 'parent_node_id',   42,
              'attribute_filter', array( 'and',
                                         array( 'priority', '>', '20' ),
                                         array( 'article/
title', '=', 'abc' ) ) ) )}
```

This example demonstrates how to do filtering on a non-attribute. Instead of specifying the identifier of the class and the attribute, "priority" is used. Only article nodes that have their title attribute set to "abc" and have a priority greater than 20 will be included in the result.

Example 17

```
{def $nodes=fetch( 'content', 'list',
        hash( 'parent_node_id', 42,
            'attribute_filter', array( 'and',
                                    array( 'owner',
                                     '=',
                                    $current_user.contentobject_id ) ,
                                    array( 'class_identifier', '=', 'folder' ) ) ,
            'depth', 3 ) )}

{foreach $nodes as $node}
    {$node.name|wash}<br />
{/foreach}

{undef $nodes}
```

Fetches folder nodes which are until the third level under the parent node number 42 and are created by the current user.

Example 18

```
{fetch( 'content', 'list',
        hash( 'parent_node_id',  2,
            'only_translated', true() ) )}
```

This example demonstrates how to fetch only nodes that reference objects that have been translated into the current language (the language that the siteaccess is using).

Example 19

```
{fetch( 'content', 'list',
        hash( 'parent_node_id',  2,
            'limit',          3,
            'only_translated', true(),
            'language',        'ger-DE' ) )}
```

This example demonstrates another way to fetch translated objects. The fetch will return nodes that reference objects which have been translated into the German language.

Example 20

```
{fetch( 'content', 'list',
        hash( 'parent_node_id', 2,
              'sort_by', array( 'priority', false() ),
              'limit', 15,
              'extended_attribute_filter',
                        hash( 'id', 'MyFilter',
                        'params', hash( 'value1', 'text1',
                                        'value2', 'text2' ) ) ) )}
```

This example demonstrates how the extended attribute filter can be used. The template code above will fetch objects that match the following condition: any object using the ezstring datatype containing either "text1" or "text2". The necessary PHP implementation is shown in the example below.

```
class eZMyExtendedFilter
{
    /*!
     Constructor
    */
    function eZMyExtendedFilter()
    {
        // Empty...
    }

    function createSqlParts( $params )
    {
        $sqlTables= ', ezcontentobject_attribute as myfilter_alias ';

        $sqlJoins = ' ezcontentobject_tree.contentobject_id =
myfilter_alias.contentobject_id AND ezcontentobject_tree.contentobject_version
= myfilter_alias.version AND myfilter_alias.data_type_string = "ezstring"
AND';


        if ( isset( $params['value1'] ) )
        {
            $value1 = $params['value1'];
        }
        else
        {
            $value1 = 'fooo';
        }
        if ( isset( $params['value2'] ) )
        {
```

5

```
                $value2 = $params['value2'];
        }
        else
        {
                $value2 = 'boooo';
        }

        $sqlCondArray = array();

        $sqlCondArray[] = 'myfilter_alias.data_text = "' . $value1 . '"';
        $sqlCondArray[] = 'myfilter_alias.data_text = "' . $value2 . '"';

        $sqlCond = implode( ' or ', $sqlCondArray );

        $sqlCond = ' ( ' . $sqlCond . ' ) AND ' . $sqlJoins . ' ';

        return array( 'tables' => $sqlTables, 'joins' => $sqlCond );

    }
}
```

Example 21

```
{fetch( 'content', 'list', hash( 'parent_node_id', 2,
                                  'ignore_visibility', true() ) )}
```

This example demonstrates how to fetch nodes regardless of their visibility status. The operation above will return both visible and hidden nodes that are directly below node #2.

Example 22

```
{def $nodes=fetch( 'content', 'list', hash( 'parent_node_id', 2,
                                  'as_object', false() ) )}

The result is an array of arrays. The first inner array reveals information
about the node "{$nodes[0].name|wash}":
<br /> {$nodes[0]|attribute( show, 1 )}
```

This example demonstrates the behavior when the "as_object" parameter is set to FALSE. The following output will be produced:

The result is an array of arrays. The first inner array reveals information about the node "Multi-price products":

Attribute	Type	Value
contentclass_id	string	1
current_version	string	1
id	string	61
initial_language_id	string	2
is_published	string	
language_mask	string	3
modified	string	1161682117
name	string	'Multiprice products'
owner_id	string	14
published	string	1161682117
remote_id	string	'f3f9d3443e61498e01523e3016ca136b'
section_id	string	1
status	string	1
contentobject_id	string	61
contentobject_is_published	string	1
contentobject_version	string	1
depth	string	2
is_hidden	string	
is_invisible	string	
main_node_id	string	63
modified_subnode	string	1161682290
node_id	string	63
parent_node_id	string	2
path_identification_string	string	'multiprice_products'
path_string	string	'/1/2/63/'
priority	string	
sort_field	string	2
sort_order	string	
class_name	string	'Folder'
class_identifier	string	'folder'
real_translation	string	'eng-GB'

Example 23

```
{def $items=fetch('content', 'tree', hash( 'parent_node_id', 2,
                     'attribute_filter', array( 'and',
                                        p;array('state', "=", 3))
                 ))}
```

The following fetch function would bring back all nodes under node 2 that has a state with the id of 3.

list_count

Summary

Fetches the number of children of a node.

Usage

```
fetch( 'content', 'list_count',
       hash( 'parent_node_id',          parent_node_id,
           [ 'class_filter_type',        class_filter_type,        ]
           [ 'class_filter_array',       class_filter_array,       ]
           [ 'attribute_filter',         attribute_filter,         ]
           [ 'extended_attribute_filter', extended_attribute_filter, ]
           [ 'main_node_only',           main_node_only,           ] ) )
```

Parameters

Name	Type	Description	Required
parent_node_id	integer	The ID number of the parent node.	Yes.
class_filter_type	string	The type of class filtering (include/ exclude).	No.
class_filter_array	array	The classes that should be filtered.	No.
attribute_filter	mixed	Filter logic for attribute level filtering.	No.
extended_attribute_ filter	mixed	The extended attribute level filter logic.	No.
main_node_only	integer	Type of nodes that should be fetched (all or main nodes only).	No.

Returns

An integer (the number of nodes).

Description

This function operates in almost the same way as the "list" fetch function. The difference is that instead of returning the actual nodes, it returns the count (the number of nodes that were found). The "list_count" function takes the same parameters as the "list" function with some exceptions (sorting, grouping, limit/offset, etc. is not supported). Please refer to the documentation of the "list" (page 714) function for a detailed description of the parameters.

Examples

Example 1

```
{def $count=fetch( 'content', 'list_count', hash( 'parent_node_id', 42 ) )}
Node number 42 has {$count} number of children.
```

Outputs the number of nodes that are below node number 42.

5

locale

Summary

Fetches the current or a specified locale.

Usage

```
fetch( 'content', 'locale' [, hash( 'locale_code', locale_code ) ] )
```

Parameters

Name	Type	Description	Required
locale_code	string	The code of the locale to be fetched.	No.

Returns

An ezlocale (page 1076) object.

Description

This function fetches either the default/current locale (specified using the "Locale" directive within the "[RegionalSettings]" block of "site.ini") or the locale specified by the optional "locale_code" parameter. The "locale_code" must be a valid locale code, for example "eng-GB", "nor-NO" and so on. Locales are listed and specified in the "share/locale" directory.

Examples

Example 1

```
{fetch( 'content', 'locale' )}
```

Fetches and returns the current locale.

Example 2

```
{fetch( 'content', 'locale', hash( 'locale_code', 'rus-RU' ) )}
```

Fetches and returns the Russian locale.

locale_list

Summary

Fetches the available locales.

Usage

```
fetch( 'content', 'locale_list' )
```

Returns

An array of ezlocale (page 1076) objects.

Description

This function fetches all the available locales and returns an array of ezlocale (page 1076) objects.

Examples

Example 1

```
{def $locales=fetch( 'content', 'locale_list' )}

{foreach $locales as $locale}
    {$locale.locale_code} : {$locale.country_name} <br />
{/foreach}
```

Outputs the locale codes and the names of the countries that the different locales belong to.

5

navigation_part

Summary

Fetches information about a navigation part.

Usage

```
fetch( 'content', 'navigation_part', hash( 'identifier', identifier ) )
```

Parameters

Name	Type	Description	Required
identifier	string	The identifier of the desired navigation part.	Yes.

Returns

An associative array (see below) or FALSE.

Description

This function fetches information about a navigation part. The identifier of the navigation part must be specified using the "identifier" parameter. The function returns a hash that consists of the following elements:

Key	Type	Description
name	string	The actual name of the navigation part (for example "Content structure").
identifier	string	The identifier of the navigation part (for example "ezcontentnavigationpart").

Examples

Example 1

```
{def $navigation_part=fetch( 'content', 'navigation_part',
                             hash( 'identifier', 'ezcontentnavigationpart' ) )}
{$navigation_part.name}
```

Outputs the name of the navigation part identified by the string "ezcontentnavigationpart".

navigation_parts

Summary

Fetches all available navigation parts.

Usage

```
fetch( 'content', 'navigation_parts' )
```

Returns

An array of hashes (see below).

Description

This function fetches all the available navigation parts. The function returns an array with hashes containing the following elements:

Key	Type	Description
name	string	The actual name of the navigation part (for example "Content structure").
identifier	string	The identifier of the navigation part (for example "ezcontentnavigationpart").

Examples

Example 1

```
{def $navigation_parts=fetch( 'content', 'navigation_parts' )}

{foreach $navigation_parts as $navigation_part}
    {$navigation_part.identifier} : {$navigation_part.name} <br />
{/foreach}
```

Outputs the identifiers and names of all the available navigation parts.

node

Summary

Fetches a node (identified by either an ID number or a path).

Usage

```
fetch( 'content', 'node', hash( [ 'node_id',      node_id,  ]
                                 [ 'node_path',   node_path, ]
                                 [ 'language_code', language_code, ]
                                 [ 'remote_id',    remote_id ] ) )
```

Parameters

Name	Type	Description	Required
node_id	integer	The ID number of the node that should be fetched.	No.
node_path	string	The path of the node that should be fetched.	No.
language_code	string	A new parameter since eZ Publish 4.1. This enables you to fetch a node in a given language.	No.
remote_id	string	A new parameter since eZ Publish 4.1. This enables you to fetch a node by its remote ID number.	No.

Returns

An ezcontentobjecttreenode (page 1051) object of FALSE.

Description

This function fetches a single node and returns it as a ezcontentobjecttreenode (page 1051). The target node must be specified using either the "node_id" or the "node_path" parameter. If no node can be found, or if an error occurs, the function will return FALSE.

Examples

Example 1

```
{def $my_node=fetch( 'content', 'node', hash( 'node_id', 96 ) )}
{$my_node.name|wash}
```

Fetches node number 96 and outputs the name of the object that is encapsulated by that node.

Example 2

```
{def $my_node=fetch( 'content', 'node', hash( 'node_path', 'news/
article_test' ) )}
{$my_node.name|wash}
```

Fetches the node by the specified path and outputs the name of the object that is encapsulated by that node.

Example 3

```
{def $my_node=fetch( 'content', 'node', hash( 'node_path', 'news/
article_test' ) )}
Views: {$my_node.view_count}
```

Fetches the node by the specified path and outputs the number of times this node has been viewed. (The "updateviewcount.php" cronjob must be run periodically.)

Example 4

```
fetch('content', 'node', hash('node_id', 66, 'language_code', 'pol-PL'))
```

This parameter is especially useful when you need to grab content or a URL in a different language than the one users are viewing a siteaccess in.

Example 5

```
fetch('content', 'node', hash('remote_id', '1b26c0454b09bb49dfb1b9190ffd67cb'
))
```

5

non_translation_list

Summary

Fetches locales that a version of an object may be translated into.

Usage

```
fetch( 'content', 'non_translation_list', hash( 'object_id', id, 'version',
version )
```

Parameters

Name	Type	Description	Required
object_id	integer	The ID number of the target object.	Yes.
version	integer	The target version number.	Yes.

Returns

An array of ezlocale (page 735) objects or FALSE.

Description

This function will fetch all the locales that a specific version of a content object may be translated to. The locales which the version is already translated to will not be included. The function returns an array of ezlocale (page 735) objects. It will return FALSE if there are no more alternate locales.

Examples

Example 1

```
{def $locales=fetch( 'content', 'non_translation_list',
                            hash( 'object_id', 42,
                                 'version',   3 ) )}

{foreach $locales as $locale}
    {$locale.language_name} <br />
{/foreach}
```

Outputs the language names of the locales that version number 3 of object number 42 can be translated to.

5

object

Summary

Fetches a content object (specified by an ID number).

Usage

```
fetch( 'content', 'object', hash( 'object_id', object_id )
                                  ( 'remote_id', remote_id ) )
```

Parameters

Name	Type	Description	Required
object_id	integer	The ID number of the target object.	Yes.
remote_id	string	New parameter since eZ Publish 4.1. This enables you to fetch an object based on its remote ID.	No.

Returns

An ezcontentobject (page 1032) object or FALSE.

Description

This function fetches a content object. The ID number of the object must be specified using the "object_id" parameter. The function returns an ezcontentobject (page 1032) object. It will return FALSE if a non-existing ID number is provided or if an error occurs.

Examples

Example 1

```
{def $object=fetch( 'content', 'object', hash( 'object_id', 13 ) )}
{$object.name|wash}
```

Outputs the name of object number 13.

Example 2

```
fetch('content', 'object', hash('remote_id',
'f5c88a2209584891056f987fd965b0ba'))
```

object_by_attribute

Summary

DEPRECATED

object_count_by_user_id

Summary

Fetches the number of objects (of a class) created by a user.

Usage

```
fetch( 'content', 'object_count_by_user_id',
       hash( 'class_id', class_id,
             'user_id',  user_id ) )
```

Parameters

Name	Type	Description	Required
class_id	integer	The ID number of the target class.	Yes.
user_id	integer	The ID number of the user (object ID).	Yes.

Returns

The number of objects (as an integer).

Description

This function counts the number of objects (of a certain type) that were created by a user. Both
the type of the object (the class) and the user must be specified. The function returns an integer
revealing the number of objects that were found.

Examples

```
{def count=fetch( 'content', 'object_count_by_user_id', hash( 'class_id', 13,
'user_id', 14 ) )}
{$count}
{undef $count}
```

Outputs the number of objects (of class number 13) that have been created by user number 14.

pending_count

Summary

Fetches the number of pending objects for the current user.

Usage

```
fetch( 'content', 'pending_count' )
```

Returns

An integer revealing the number of pending objects.

Description

This function reveals the number of pending objects that belong to the current user.

Examples

Example 1

```
{def $count=fetch( 'content', 'pending_count' )}
There are {$count} pending objects.
```

Outputs the number of pending objects that belong to the current user.

5

pending_list

Summary

Fetches the pending objects for the current user.

Usage

```
fetch( 'content', 'pending_list' )
```

Returns

An array of (page 1032) objects or FALSE.

Description

This function fetches the pending objects that belong to the current user. The function returns an array of (page 1032) or FALSE if there are no pending objects.

Examples

Example 1

```
{def $pending=fetch( 'content', 'pending_list' )}

{foreach $pending as $object}
    {$object.name}
{/foreach}
```

Outputs the names of the pending objects that belong to the current user.

5

recent

Summary

Fetches nodes where the current user recently published something.

Usage

```
fetch( 'content', 'recent' )
```

Returns

An array of ezcontentbrowserecent (page 1021) objects or FALSE.

Description

This function fetches nodes under which the current user recently has published something. The function returns an array of ezcontentbrowserecent (page 1021) objects. If there are no nodes under which the current user has published something, the function will return FALSE.

Examples

Example 1

```
{def $recent=fetch( 'content', 'recent' )}
{foreach $recent as $element}
    {$element.name}
{/foreach}
```

Outputs the names of the nodes under which the current user recently published something.

5

related_objects

Summary

Fetches related objects.

Usage

```
fetch( 'content', 'related_objects',
       hash( 'object_id',            object_id,
           [ 'attribute_identifier', attribute_identifier, ]
           [ 'all_relations',        boolean,              ]
           [ 'group_by_attribute',   boolean,              ]
           [ 'sort_by',              sort_by,              ]
           [ 'limit',                limit,                ]
           [ 'offset',               offset,               ]
           [ 'as_object',            boolean,              ]
           [ 'load_data_map',        boolean               ] ) )
```

Parameters

Name	Type	Description	Required
object_id	integer	The ID number of the target object.	Yes.
attribute_identifier	mixed	The ID number or class/attribute identifier of the target attribute.	No.
all_relations	mixed	Controls what types of relations that should be fetched, default is FALSE.	No.
group_by_attribute	boolean	Groups the result based on the attributes, default value is TRUE.	No.
sort_by	array	The sorting mechanism that should be used.	No.
limit	integer	A new parameter since eZ Publish 4.1 which enables you to fetch an exact number of items.	No.
offset	integer	A new parameter since eZ Publish 4.1 which enables you to define an offset.	No.
as_object	boolean	A new parameter since eZ Publish 4.1 which enables you to fetch the items as objects.	No.
load_data_map	boolean	A new parameter since eZ Publish 4.1 which enables you to load the data_map array.	No.

5

Returns

An array of ezcontentobject (page 1032) objects or a two-dimensional array if 'group_by_attribute' is TRUE. If no objects are found, the function will return FALSE.

Description

This function fetches the objects that have been related to an object specified by the "object_id" parameter. It is capable of returning all related objects regardless of their relation type (attributes using the "Object relation" (page 572) or "Object relations" (page 574) datatype or standard object-level relations classified as "Common", "XML linked" or "XML embedded").

The "attribute_identifier" parameter makes it possible to specify either an ID number or an identifier string (class/attribute - for example "my_class/my_attribute") of an attribute. This parameter is not required. The default value is zero, which makes the function return only objects that are related on an object level, not at the attribute level. This behavior is similar to 'related_contentobject_array' functional attribute of a content object. When the parameter is used, the system will return objects that have been related using an attribute that is based on either the "Object relation" (page 572) or "Object relations" (page 574) datatype.

The "all_relations" parameter makes it possible to fetch all types of relations (when set to TRUE). This parameter is not required and the default value is FALSE. In addition, the value of this parameter can also be an array. The array may consist of the following strings: "common", "xml_link", "xml_embed" and "attribute". Each of them correspond to a relation type. The types can be mixed and provided in any order, which means that it is possible to use this parameter to do basic filtering. Refer to the "Object relations" page of the "Concepts and basics" chapter for more information about the relation types.

The "group_by_attribute" parameter can only be used when the value of the "all_relations" parameter is non-FALSE (either TRUE or an array). When the "group_by_attribute" parameter has been set to TRUE, the function will return a two-dimensional array instead of just an array of objects. The following example shows how this structure is built up:

```
$related_objects_grouped = array(

0 => array( $object1, $object2 ... ),

    // Objects related on content object level
    attr_id_1 => array( $object1, $object2 ... ),
    attr_id_2 => array( $object1, $object2 ... ),

    ...

    // Objects related by attributes
);
```

The "sort_by" parameter makes it possible to sort the result in different ways. This parameter

5

must be provided as an array of arrays that define the sorting methods. The first element of each array must be the desired sorting method. The second element of the array must be the sorting direction, it can be either TRUE or FALSE (ascending or descending). Please note that this parameter works in the very same way as the "sort_by" parameter of the list (page 714) fetch function. However, it currently only supports the following sorting methods:

- class_identifier

- class_name

- modified

- name

- published

- section

Please note that using other sort methods will lead to an error.

Examples

Example 1

```
{def $related=fetch( 'content', 'related_objects',
hash( 'object_id', $node.object.id,
      'all_relations', true(),
      'group_by_attribute', true(),
      'sort_by', array( array( 'class_name', true() ),
                        array( 'name', true() ) ) ) )}
```

Returns all relations grouped in arrays by attribute ID, then sorted by class name and by object's name in ascending order.

Example 2

```
fetch('content', 'related_objects', hash(..., 'limit', '10', offset, '0',
'as_object', true(), 'load_data_map', false()))
```

5

related_objects_count

Summary

Fetches the number of related objects.

Usage

```
fetch( 'content', 'related_objects_count',
   hash( 'object_id',              object_id,
       [ 'attribute_identifier',   attribute_identifier, ]
       [ 'all_relations',          boolean,              ] ) )
```

Parameters

Name	Type	Description	Required
object_id	integer	The ID number of the target object.	Yes.
attribute_identifier	mixed	The ID number or class/attribute identifier of the target attribute.	No.
all_relations	mixed	Controls what type of relations that should be feetched, default is FALSE.	No.

Returns

An integer (the number of related objects).

Description

This fetch function operates in almost the same way as the "related_objects" fetch function. The difference is that instead of returning the related objects themselves, it returns the count (the number of related objects that were found). The "related_objects_count" function takes the same parameters as the "related_objects" function with some exceptions (for example sorting is not supported). Please refer to the documentation of the "related_objects" function for a detailed description of the parameters.

5

reverse_related_objects

Summary

Fetches reverse related objects.

Usage

```
fetch( 'content', 'reverse_related_objects',
       hash( 'object_id',            object_id,
           [ 'attribute_identifier', attribute_identifier, ]
           [ 'all_relations',        boolean,              ]
           [ 'group_by_attribute',   boolean,              ]
           [ 'sort_by',              sort_by               ]
           [ 'ignore_visibility',    ignore_visibility     ]
           [ 'limit',                limit                 ]
           [ 'offset',               offset                ]
           [ 'as_object',            as_object             ]
           [ 'load_data_map',        load_data_map         ] ) )
```

Parameters

Name	Type	Description	Required
object_id	integer	The ID number of the target object.	Yes.
attribute_identifier	mixed	The ID number or class/attribute identifier of the target attribute.	No.
all_relations	boolean	Controls whether to fetch all types of relastions or not, default is FALSE.	No.
group_by_attribute	boolean	Groups the result based on the attributes, default value is TRUE.	No.
sort_by	array	The sorting mechanism that should be used.	No.
ignore_visibility	boolean	Makes it possible to fetch hidden nodes (TRUE by default).	No.
limit	integer	A new parameter since eZ Publish 4.1 which enables you to fetch an exact number of items	No.
offset	integer	A new parameter since eZ Publish 4.1 which enables you to define an offset.	No.
as_object	boolean	A new parameter since eZ Publish 4.1 which enables you to fetch the items as objects.	No.
load_data_map	boolean	A new parameter since eZ Publish 4.1 which enables you to load the data_map array.	No.

5

Returns

An array of ezcontentobject (page 1032) objects or FALSE.

Description

This function makes it possible to fetch reverse related objects. The target object must be specified using the "object_id" parameter. The function will return an array of ezcontentobject (page 1032) objects which are using the target object through the conventional object relation mechanism. If no objects are found, the function will return FALSE.

Class attribute filtering

By making use of the "attribute_identifier" parameter, it is possible to fetch reverse related objects that make use of the target object by the way of an attribute. The attribute must use either the "Object relation" (page 572) or the "Object relations" (page 574) datatype. The "attribute_identifier" parameter can either be the ID number of the class attribute or a string that consists of the class identifier, a slash and the class attribute identifier (for example "my_class/my_attribute").

The "all_relations" parameter makes it possible to fetch all types of relations. This parameter is not required and the default value is FALSE.

The "group_by_attribute" parameter can only be used when the "all_relations" parameter has been set to TRUE. When the "group_by_attribute" parameter has been set to TRUE, the function will return a two-dimensional array instead of just an array of objects. The following example shows how this structure is built up:

```
$related_objects_grouped = array(

0 => array( $object1, $object2 ... ),

    // Objects related on content object level
    attr_id_1 => array( $object1, $object2 ... ),
    attr_id_2 => array( $object1, $object2 ... ),

    ...

    // Objects related by attributes
);
```

The "sort_by" parameter makes it possible to sort the result in different ways. This parameter must be provided as an array of arrays that define the sorting methods. The first element of each array must be the desired sorting method. The second element of the array must be the sorting direction, it can be either TRUE or FALSE (ascending or descending). Please note that this parameter works in the very same way as the "sort_by" parameter of the list (page 714) fetch function. However, it currently only supports the following sorting methods:

- class_identifier

- class_name

- modified

- name

- published

- section

Please note that using other sort methods will lead to an error.

Fetching hidden nodes

The "ignore_visibility" parameter makes it possible to fetch hidden nodes. It can be set to either "true()" or "false()". If set to "true()", the fetch will ignore the visibility flags of nodes and thus it will return all nodes regardless of their visibility status.

Examples

Example 1

```
{def $objects=fetch( 'content', 'reverse_related_objects',
                     hash( 'object_id', 256 ) )}

{foreach $objects as $object}
    {$object.name|wash} <br />
{/foreach}
```

Outputs the names of the objects that make use of object number 256 through the conventional related objects mechanism.

Example 3

```
{def $objects=fetch( 'content', 'reverse_related_objects',
                     hash( 'object_id',            256,
                           'attribute_identifier', '4096' ) )}

{foreach $objects as $object}
    {$object.name|wash} <br />
{/foreach}
```

Outputs the names of the objects that make use of object number 256 through class attribute number 4096.

5

Example 3

```
{def $objects=fetch( 'content', 'reverse_related_objects',
                     hash( 'object_id',          256,
                           'attribute_identifier', 'my_class/my_attribute' ) )}

{foreach $objects as $object}
    {$object.name|wash} <br />
{/foreach}
```

Outputs the names of the objects that make use of object number 256 through an attribute called
"my_attribute" that is a part of class "my_class".

Example 4

```
fetch('content', 'reverse_related_objects', hash(..., 'limit', '10', offset,
'0', 'as_object', true(), 'load_data_map', false()))</
pre>
```

5

reverse_related_objects_count

Summary

Fetches the number of reverse related objects.

Usage

```
fetch( 'content', 'reverse_related_objects_count',
       hash( 'object_id',            object_id,
           [ 'attribute_identifier', attribute_identifier, ]
           [ 'all_relations',        all_relations,        ]
           [ 'ignore_visibility',    ignore_visibility     ] ) )
```

Parameters

Name	Type	Description	Required
object_id	integer	The ID number of the target object.	Yes.
attribute_identifier	mixed	The ID number or class/attribute identifier of the target attribute.	No.
all_relations	boolean	Controls whether to fetch all types of relastions or not, default is FALSE.	No.
ignore_visibility	boolean	Makes it possible to fetch hidden nodes.	No.

Returns

An integer (number of reverse related objects).

Description

This function is the same as the "reverse_related_objects" function. However, instead of returning an array of objects, it returns the number of objects that were found. Please refer to the documentation of the "reverse_related_objects" (page 751) function for information about the parameters.

5

same_classattribute_node

Summary

Fetches nodes containing attributes that match a certain value.

Usage

```
fetch( 'content', 'same_classattribute_node',
       hash( 'classattribute_id', classattribute_id,
             'value',             value,
     'datatype',        datatype ) )
```

Parameters

Name	Type	Description	Required
id	integer	The ID number of the class attribute that should be examined.	Yes.
value	mixed	The value that should be matched.	Yes.
datatype	string	Must be either "int", "float" or "text".	Yes.

Returns

An array of ezcontentobjecttreenode (page 1051) objects or FALSE.

Description

This function will go through all object attributes that are instances of the class attribute specified by the "classattribute_id" parameter. The value that should be matched must be specified using the "value" parameter. In addition, the type of data (either "int", "float" or "text") that the datatype representing the attribute must be provided. The function returns an array of ezcontentobjecttreenode (page 1051) objects or FALSE (if there is no match).

Examples

```
{def $matched_nodes=fetch( 'content', 'same_classattribute_node',
                       hash( 'classattribute_id', 245,
                             'value',             'example',
                             'datatype',         'text' ) )}

{foreach $matched_nodes as $matched_node}
    {$matched_node.name|wash} <br />
{/foreach}
```

Outputs the names of the nodes that make use of class attribute number 245 and where the text contents of the object attribute equals the string "example".

5

search

Summary

Fetches nodes containing data that match a certain criteria.

Usage

```
fetch( content, search,
       hash( text, text,
           [ offset,            offset,            ]
           [ limit,             limit,             ]
           [ section_id,        id,                ]
           [ subtree_array,     array,             ]
           [ publish_timestamp, time,              ]
           [ publish_date,      date,              ]
           [ class_id,          id,                ]
           [ class_attribute_id, id,               ]
           [ sort_by,           sort_by,           ]
           [ limitation,        limitation,        ]
           [ ignore_visibility, ignore_visibility, ] ) )
```

5

Parameters

Name	Type	Description	Required
text	string	The text that should be matched.	Yes.
subtree_array	mixed	Array node ID number under which the system should search.	No.
offset	integer	The offset to start at.	No.
limit	integer	The number of nodes that should be returned.	No.
publish_timestamp	integer	Only search objects with the specified publishing date/time (as a UNIX timestamp).	No.
publish_date	integer	Only search objects published during the last day / week / month / three months / one year.	No.
section_id	integer	Only match objects that are in this section.	No.
class_id	integer	Only match objects that are instances of this class. This parameter can also be an array of class ID numbers.	No.
class_attribute_id	integer	Only search within this attribute. This parameter can also be an array of attribute ID numbers.	No.
sort_by	mixed	Sort the result. See description below.	No.
limitation	array	Limitation array (emtpy array = access override).	No.
ignore_visibility	boolean	Makes it possible to get hidden nodes.	No.

Returns

An array of hashes (see below) or FALSE.

Description

This function will perform a search and it will return the hits that the current user has read access to. The function returns an array of hashes. The hashes consist of the following elements:

SearchResult	array	An array of the nodes (as ez-contentobjecttreenode (page 1051) objects) that matched the search conditions.
SearchCount	integer	The total number of nodes that matched the search conditions.
StopWordArray	array	An array of strings contain-

		ing words that were excluded from the search.

Sorting

The "sort_by" parameter makes it possible to sort the result in different ways. This parameter behaves exactly in the same way as it does for the "list" function. Please refer to the documentation page of the "list (page 714)" function for a complete explanation of this parameter.

Searching by the date of publishing

The "publish_date" parameter makes it possible to search objects published during the specified period. The following table reveals the possible values of this parameter.

1	one day
2	one week
3	one month
4	three months
5	one year

The "publish_timestamp" parameter makes it possible to search objects with the specified publishing date/time. This value must be a UNIX timestamp. You can also use an array of two elements in order to search withing the given range.

Please note that you can not use both "publish_timestamp" and "publish_date" at the same time. If you use "publish_timestamp" then "publish_date" will be ignored.

Access override

The "limitation" parameter makes it possible to instruct the system to use an alternate set of access limitations instead of the ones that belong to the current user. This is typically useful when there is a need to match and return nodes that the current user does not have access to. All permission checking can be skipped by providing an empty array. It is also possible to provide arrays that define access limitations of other users and/or custom sets of limitations. Note that this is for advanced/experienced users. You'll have to look in the source code of the kernel for details related to the format of this array.

Visibility override

The "ignore_visibility" parameter makes it possible to allow the search function to get hidden nodes. It can be set to either "true()" or "false()" (default). If set to "true()", the search engine will ignore the visibility flags of nodes and thus the result will contain all kinds of nodes regardless if they are hidden or not. In other words, this parameter overrides the "ShowHiddenNodes" (page 2515) configuration directive for a specific search operation in a template.

Examples

Example 1

```
{def $search=fetch( 'content', 'search',
                    hash( 'text',     'example',
                          'class_id', array( '2', '5' ) ) )}

The search returned {$search.SearchCount} matches. <br />

{foreach $search.SearchResult as $matched_node}
    {$matched_node.name|wash} <br />
{/foreach}
```

Outputs the names of all nodes which encapsulte objects of classes 2 and 5 containing the word "example".

Example 2

```
{def $search=fetch( 'content', 'search',
                    hash( 'text',     'example',
                          'publish_timestamp', array( '1033920746',
'1033920789' ) ) )}

The search returned {$search.SearchCount} matches. <br />

{foreach $search.SearchResult as $matched_node}
    {$matched_node.name|wash} <br />
{/foreach}
```

Outputs the names of all nodes which encapsulte objects published between the specified date/time values.

5

section_list

Summary

Fetches the available sections.

Usage

```
fetch( 'content', 'section_list' )
```

Returns

An array of ezsection (page 1111) objects.

Description

This function fetches all the available sections. An array of ezsection (page 1111) objects will be
returned.

Examples

Example 1

```
{def $sections=fetch( 'content', 'section_list' )}
{foreach $sections as $section}
    {$section.name} <br />
{/foreach}
```

Outputs the names of all the available sections.

5

tipafriend_top_list

Summary

Fetches the most popular (most tipped) nodes.

Usage

```
fetch( 'content', 'tipafriend_top_list',
    hash( [ 'offset',     offset,    ]
          [ 'limit',      limit,     ]
          [ 'start_time', start_time, ]
          [ 'end_time',   end_time,  ]
          [ 'duration',   duration,  ]
          [ 'ascending',  ascending, ]
          [ 'extended',   extended   ] ) )
```

Parameters

Name	Type	Description	Required
offset	integer	The offset to start at.	No.
limit	integer	The number of nodes that should be returned.	No.
start_time	integer	The time to start at.	No.
end_time	integer	The time to end at.	No.
duration	integer	The period of time that should be used.	No.
ascending	boolean	The sorting direction that should be used. If FALSE (default), the results will be sorted descending by the number of times each node was tipped.	No.
extended	boolean	If TRUE, an array of hashes will be returned.	No.

Returns

An array of ezcontentobjecttreenode (page 1051) objects, an array of hashes (see below) or FALSE.

Description

This function fetches the nodes that were most tipped using the "Tip a friend" feature. The "Tip a friend" feature can be accessed by requesting the "tipafriend" view of the "content" module, it takes a node ID as a parameter, for example: "http://www.example.com/content/tipafriend/44".

The returned result is sorted by the number of times the nodes were tipped. The sorting direction can be controlled using the "ascending" parameter, it can be set to either TRUE or FALSE (default).

The "limit" and "offset" parameters can be used to control the number of items returned and the offset to start at.

The "start_time", "end_time" and "duration" parameters can be used to specify a time period that the function should operate within. These parameters make it possible to fetch the most popular/ tipped nodes for a certain period of time (for example last 24 hours, last week, last month, etc.). If the "start_time" and "duration" parameters are specified, results for the "start_time+duration" period of time will be fetched. If the "end_time" and "duration" parameters are specified, results for the "end_time-duration" period of time will be fetched. If all of these parameters are omitted, the function will cover all nodes, regardless when they were tipped. If both the "start_time" and "duration" parameters are omitted, then only results for the period before "end_time" will be fetched. If only the "start_time" or "duration" parameter is set, the function will automatically use the current time as the "end_time" parameter.

By default (when the "extended" parameter is FALSE), the function returns an array of ezcontentobjecttreenode (page 1051) objects. If the "extended" parameter is set to TRUE, the function will return an array of the following hashes:

Key	Type	Description
node_id	string	The ID number of the node.
count	string	The number of times the node was tipped.
node	object	The node itself (as an ez-contentobjecttreenode (page 1051) object).

If the function is unable to find any tipped nodes or if an error occurs, it will return FALSE.

Examples

Example 1

```
{def $popular_nodes=fetch( 'content', 'tipafriend_top_list',
                    hash( 'limit',  10,
                          'offset', 0,
                          'start_time', maketime( 0, 0, 0 ),
                          'duration', mul( 60, 60, 24 )
                           ) )}

{foreach $popular_nodes as $popular_node}
    {$popular_node.name|wash} <br />
{/foreach}
{undef}
```

5

Outputs the names of the ten most popular nodes during the last 24 hours. The "start_time" parameter sets 00:00:00 as the time and the current date. The "duration" parameter is set to a period that equals one day (60 seconds x 60 minutes x 24 hours).

Example 2

```
{def $popular_nodes=fetch( 'content', 'tipafriend_top_list',
                    hash( 'ascending', false(),
                          'extended', true()
                        ) )}
{foreach $popular_nodes as $popular_node}
    <tr>
        <td>{$popular_node.node.name|wash}</td>
        <td>{$popular_node.count}</td>
    </tr>
{/foreach}
{undef}
```

Generates a list of popular nodes by outputting their names and the number of times they were tipped. The results are sorted by the number of times the nodes were tipped.

5

translation_list

Summary

Fetches the locales that can be used to translate objects.

Usage

```
fetch( 'content', 'translation_list' )
```

Returns

An array of ezlocale (page 1076) objects.

Description

This function fetches all the available locales that can be used to translate the contents of objects. This translations can be managed from within the "Translations" section of the "Setup" part within the administration interface. The function returns an array of ezlocale (page 1076) objects.

Examples

Example 1

```
{def $locales=fetch( 'content', 'translation_list' )}

{foreach $locales as $locale}
    {$locale.language_name} <br />
{/foreach}
```

Outputs the names of the languages that can be used to translate the contents of objects.

5

trash_count

Summary

Fetches the number of objects that are in the trash.

Usage

```
fetch('content', 'trash_count', hash ('attribute_filter', array))
```

Parameters

Name	Type	Description	Required
attribute_filter	mixed	A new parameter since eZ Publish 4.1. This enables you to run attribute filtering for trash items	No.

Returns

An integer revealing the number of objects that are in the trash.

Description

This function returns the number of objects (as an integer) that are in the trash. An object is considered to be in the trash if its status field is set to archived.

For more information regarding the use of the "attribute_filter" parameter, please visit the following page.

Examples

Example 1

```
{def $trash_count=fetch('content', 'trash_count' )}
There are {$trash_count} objects in the trash.
```

Outputs the number of items that are in the trash.

Example 2

```
fetch('content', 'trash_count', hash('attribute_filter', array(...)))</pre>
```

5

trash_object_list

Summary

Fetches the objects that are in the trash.

Usage

```
fetch( 'content', 'trash_object_list',
       hash( [ 'offset', offset, ]
               [ 'limit',  limit   ]
               [ 'attribute_filter',   attribute_filter   ]
               [ 'sort_by',   sort_by   ]
               [ 'as_object',   as_object   ] ) )
```

Parameters

Name	Type	Description	Required
offset	integer	The offset to start at.	No.
limit	integer	The number of objects that should be returned.	No.
attribute_filter	mixed	A new parameter since eZ Publish 4.1. This enables attribute filtering.	No.
sort_by	mixed	A new parameter since eZ Publish 4.1. This enables sorting.	No.
as_object	boolean	A new parameter since eZ Publish 4.1. This enables fetching elements as objects.	No.

Returns

An array of ezcontentobject (page 1032) objects or FALSE.

Description

This function fetches all the objects that are in the trash. An object is considered to be in the trash if its status field is set to "archived". The optional "offset" and "limit" parameters can be used to limit the result. The function returns an array of ezcontentobject (page 1032) objects or FALSE if no objects are found.

For more information regarding the use of the "attribute_filter" parameter, please visit the following page.

5

Examples

Example 1

```
{def $trashed_objects=fetch( 'content', 'trash_object_list' ) }

{foreach $trashed_objects as $object}
    {$object.name|wash} <br />
{/foreach}
```

Outputs the names of the objects that are in the trash.

Example 2

```
fetch('content', 'trash_object_list', hash('attribute_filter', array(...),
'sort_by', array(), 'as_object', true()))</
pre>
```

5

tree

Summary

Fetches the children of a node recursively.

Usage

```
fetch( 'content', 'tree',
       hash( 'parent_node_id',      parent_node_id,
            [ 'sort_by',                   sort_by,                  ]
            [ 'offset',                    offset,                   ]
            [ 'limit',                     limit,                    ]
            [ 'attribute_filter',          attribute_filter,         ]
            [ 'extended_attribute_filter', extended_attribute_filter, ]
            [ 'class_filter_type',         class_filter_type,        ]
            [ 'class_filter_array',        class_filter_array,       ]
            [ 'only_translated',           only_translated,          ]
            [ 'language',                  language,                 ]
            [ 'main_node_only',            main_node_only,           ]
            [ 'as_object',                 as_object,                ]
            [ 'depth',                     depth,                    ]
            [ 'depth_operator',            depth_operator            ]
            [ 'limitation',                limitation                ]
            [ 'ignore_visibility',         ignore_visibility         ] ) )
```

5

Parameters

Name	Type	Description	Required
parent_node_id	integer	The ID number of the parent node.	Yes.
sort_by	array	The sorting mechanism that should be used.	No.
offset	integer	The offset to start at.	No.
limit	integer	The maximum number of nodes that should be fetched.	No.
attribute_filter	mixed	Filter logic for attribute level filtering.	No.
extended_attribute_filter	mixed	The extended attribute level filter logic.	No.
class_filter_type	string	The type of class filtering (include/exclude).	No.
class_filter_array	array	The type of nodes that should be filtered.	No.
only_translated	boolean	Translation filtering (on/off).	No.
language	string	The language that should be filtered.	No.
main_node_only	boolean	Type of nodes that should be fetched (all or main nodes only).	No.
as_object	boolean	If TRUE (or omitted), an array of "ezcontentobjecttreenode" objects will be fetched. Otherwise, an array of arrays will be returned.	No.
depth	integer	The maximum level of depth that should be explored.	No.
depth_operator	string	The logic to use when checking the depth.	No.
limitation	array	Limitation array (emtpy array = access override).	No.
ignore_visibility	boolean	Makes it possible to fetch hidden nodes.	No.

Description

The "tree" function is very similar to the "list" fetch function. The only difference is that the tree function fetches child nodes recursively. The recursion depth can be controlled by the "depth" and "depth_operator" parameters. The rest of the parameters behave exactly in the same way as they do for the "list" function. Please refer to the documentation page of the "list (page 714)" function for a complete explanation of the parameters.

Depth

The depth parameter can be used to specify the level of depth (within the branch) that the function should explore when it is running. If the depth is set to one, this function will simply act

as the list function. If the depth is greater than one, the function will fetch nodes further down in the branch. The default value is unlimited.

Depth operator

The depth operator can be set to either "lt", "eq" or "gt" - meaning "less than", "equal to" and "greather than". For example, if it is set to 'eq', only nodes with the depth that was specified using the depth parameter will be fetched.

Examples

Example 1

```
{def $nodes=fetch( 'content', 'tree',
                   hash( 'parent_node_id', 42 ) )}

{foreach $nodes as $node}
    {$node.name|wash} <br />
{/foreach}
```

This example demonstrates how to fetch all the nodes that are under node number 42 recursively (all children, grand-children, etc. will be fetched). The names of the nodes are displayed.

5

tree_count

Summary

Fetches the number of children of a node recursively.

Usage

```
fetch( 'content', 'tree_count',
       hash( 'parent_node_id',           parent_node_id,
           [ 'class_filter_type',         class_filter_type,        ]
           [ 'class_filter_array',        class_filter_array,       ]
           [ 'attribute_filter',          attribute_filter,         ]
           [ 'extended_attribute_filter', extended_attribute_filter, ]
           [ 'main_node_only',            boolean,                  ]
           [ 'depth',                     depth,                    ]
           [ 'depth_operator',            depth_operator            ] ) )
```

Parameters

Name	Type	Description	Required
parent_node_id	integer	The ID number of the parent node.	Yes.
class_filter_type	string	Filter type for class filtering (include/exclude).	No.
class_filter_array	array	The type of nodes that should be filtered.	No.
attribute_filter	mixed	Filter logic for attribute level filtering.	No.
extended_attribute_filter	mixed	The extended attribute level filter logic.	No.
main_node_only	boolean	Type of nodes that should be fetched (all or main nodes only).	No.
depth	integer	The maximum level of depth that should be explored.	No.
depth_operator	string	The logic to use when checking the depth.	No.

Returns

An integer (number of nodes).

Description

This function works in the very same way as the "tree" function. The difference is that it returns only the number of nodes (instead of the actual nodes). Please refer to the documentation of the

"list" (page 714), "list_count" (page 732) and the "tree" (page 771) fetch functions.

Examples

Example 1

```
{def $count=fetch( 'content', 'tree_count',
                   hash( 'parent_node_id',     42,
                         'class_filter_type',  'exclude',
                         'class_filter_array', array( 'folder', 'comment' ) )
)}

Number of nodes: {$count}
```

This example counts the number of nodes that are children of node number 42, recursively.
Nodes that reference "folder" or "comment" objects will be excluded from the count.

5

version

Summary

Fetches a specific version of an object.

Usage

```
fetch( 'content', 'version', hash( 'object_id',  object_id,
                                   'version_id', version_id ) )
```

Parameters

Name	Type	Description	Required
object_id	integer	The ID number of the target object.	Yes.
version_id	integer	The version number that should be fetched.	Yes.

Returns

An ezcontentobjectversion (page 1056) object or FALSE.

Description

This function fetches a specific version of a content object. Both the ID number of the target object and the number of the desired version must be specified using the "object_id" and the "version_id" parameters. The function returns an ezcontentobjectversion (page 1056) object. It will return FALSE if invalid parameters have been provided or if the current user has insufficient permissions.

Examples

Example 1

```
{def $version=fetch( 'content', 'version', hash( 'object_id',  13,
                                                 'version_id',  3 ) )}

Name of version: {$version.name}
```

Outputs the name of version number 3 for object number 13.

5

version_count

Summary

Fetches the number of versions of a content object.

Usage

```
fetch( 'content', 'version_count', hash( 'contentobject', object ) )
```

Parameters

Name	Type	Description	Required
contentobject	object	The target object.	Yes.

Returns

The number of versions of an object (as an integer).

Description

This function retrieves the number of versions of a content object. The target object must be specified using the "contentobject" parameter.

Examples

Example 1

```
{* Fetch object number 13. *}
{def $object=fetch( 'content', 'object', hash( 'object_id', 13 ) )}

{* Fetch the number of versions for object number 13. *}
{def $versions=fetch( 'content', 'version_count', hash( 'contentobject',
$object ) )}

Object number 13 consists of {$versions} versions.
```

Outputs the numbre of versions that make up object number 13.

version_list

Summary

Fetches all the versions of a content object.

Usage

```
fetch( 'content', 'version_list', hash( 'contentobject', object,
                                  [ 'offset',       offset, ]
                                  [ 'limit',        limit   ]
                                  [ 'sorts',        sorts   ] ) )
```

Parameters

Name	Type	Description	Required
contentobject	object	The target object.	Yes.
offset	integer	Offset to start at.	No.
limit	integer	The number of versions that should be fetched.	No.
sorts	array	A new parameter since eZ Publish 4.1 which enables you to sort the returned versions	No.

Returns

An array of ezcontentobjectversion (page 1056) objects.

Description

This function fetches all the versions of a certain object. The object itself must be specified using the "contentobject" parameter. The "offset" and "limit" parameters are optional and can be used to limit the result. The function returns an array of ezcontentobjectversion (page 1056) objects.

Examples

Example 1

```
{* Fetch object number 13. *}
{def $object=fetch( 'content', 'object', hash( 'object_id', 13 ) )}

{* Fetch all the versions of object number 13. *}
```

5

```
{def $versions=fetch( 'content', 'version_list', hash( 'contentobject',
$object ) )}

{* Loop through all versions and display their names. *}
{foreach $versions as $version}
    {$version.name} <br />
{/foreach}
```

Outputs the names of all versions that belong to object number 13.

Example 2

```
fetch('content', 'version_list', hash('sorts', array(...)))</pre>
```

5

view_top_list

Summary

Fetches the most popular (most viewed) nodes.

Usage

```
fetch( 'content', 'view_top_list',
       hash( [ 'section_id', section_id, ]
             [ 'class_id',   class_id,   ]
             [ 'offset',     offset,     ]
             [ 'limit',      limit       ] ) )
```

Parameters

Name	Type	Description	Required
section_id	integer	The ID number of the section.	No.
class_id	integer	The ID number of the class.	No.
offset	integer	The offset to start at.	No.
limit	integer	The number of nodes that should be returned.	No.

Returns

An array of ezcontentobjecttreenode (page 1051) objects.

Description

This function fetches the most popular (most viewed) nodes. The function returns an array of ezcontentobjecttreenode (page 1051) objects. The "section_id" and "class_id" parameters can be used to filter out objects of certain type and/or objects that belong to a certain section. The "offset" and "limit" parameters can be used to limit the result.

In order to work, this function requires the use of the cronjob script. A part of this script will update the view counters of the nodes by analyzing the Apache log files. The "Scripts[]" array of a configuration override for "cronjob.ini" should include the "updateviewcount.php" script:

```
...
Scripts[]=updateviewcount.php
...
```

In addition, the logfile settings in "logfile.ini" should match the syntax of the Apache log files.

Examples

Example 1

```
{def $popular_nodes=fetch( 'content', 'view_top_list',
                           hash( 'class_id',  2,
                                 'limit',    10,
                                 'offset',    0 ) )}


{foreach $popular_nodes as $popular_node}
    {$popular_node.name|wash} <br />
{/foreach}
```

Outputs the names of the ten most popular nodes that encapsulate objects of class number 2.

5

Views

\begin{description} \item[action]\refer{label6efd6338b40193ef2ae28c185b4fd011}\ \\ Provides an interface to different actions (AddToBasket, SwapNode, etc.).

\item[advancedsearch]\refer{label77b75b795094516b6d122682840b4e97}\ \\ Provides the advanced search interface.

\item[bookmark]\refer{label4628d8e271d48bbaa747b84a43de6c63}\ \\ Provides an interface for managing the current user's bookmarks.

\item[browse]\refer{label5f4cc0521ef3ea2de595346cdab0e513}\ \\ Provides an interface for selecting node(s) by browsing the node tree.

\item[collectedinfo]\refer{label3339232b899ad7b22f3a57b128aea10e}\ \\ Provides an interface for displaying the information that was collected.

\item[collectinformation]\refer{labelbf636433e3484516f62777a796e8c7e9}\ \\ Provides an interface for collecting information.

\item[copy]\refer{label02f902272d29e3a6b4045bc381a36d83}\ \\ Provides an interface for copying a single node.

\item[copysubtree]\refer{labeld008da672bc91448c1826c89fed40b95}\ \\ Provides an interface for copying an entire subtree of nodes.

\item[diff]\refer{label9ac58c698587e4c4b88d1fe0c4cdb092}\ \\ Provides an interface for comparing two different versions of an object (DEPRECATED).

\item[download]\refer{label14e2a502024b77cd990f7f7b2a1e6253}\ \\ Provides an interface for downloading files stored by the "File" datatype.

\item[draft]\refer{labele3e266920824dc0d1d09520e6554b82d}\ \\ Provides an interface for managing the current user's drafts.

\item[edit]\refer{label9e41586622d070808abf434809b89c09}\ \\ Provides an interface for editing and translating the contents of objects.

\item[hide]\refer{labelfab7773fa6c53fc61dfdd68b90672d45}\ \\ Provides an interface for hiding and revealing nodes.

\item[history]\refer{labeleba13753af7533c15bf5c11b10464ee7}\ \\ Provides an interface for managing the versions of an object and comparing two different versions of an object.

\item[keyword]\refer{labelaaff20db2a2a2c43f29310468df1fabb}\ \\ Loads a template that can fetch objects which use keyword.

\item[move]\refer{labeld565626163d5332f8076763ad526e506}\ \\ Provides an interface for changing the location of a node.

\item[new]\refer{labelb77973e57ed9cd67a42978386e0bff82}\ \\ Loads a template that can be used to display new content since last visit.

\item[pdf]\refer{label4da5e15b79c5d18acb01d71550e2bd84}\ \\ Provides on-the-fly PDF generation of a node (DEPRECATED).

5

\item[pendinglist]\refer{label2ae1f71f10e446d8ed97f79befec706f}\ \\ Provides an overview of the current user's pending items.

\item[removeassignment]\refer{labelaa1d80c9b0dbde65963254ff50608aa4}\ \\ Provides an interface for removing node assignments.

\item[removeeditversion]\refer{labelffa852779edf896d19fb21444d352b0b}\ \\ Provides an interface for draft removal.

\item[removenode]\refer{label7102ad95e293af4fbc174137dfb6d006}\ \\ Provides an interface for removing nodes.

\item[removeobject]\refer{label2c9933e72278320e1bc773a5fdf9e1ea}\ \\ Provides an interface for removing objects.

\item[restore]\refer{label5303515b950a1dc71386a915d1febadf}\ \\ Provides an interface for restoring objects from the trash.

\item[reverserelatedlist]\refer{labela0a4cc35881d5b888d5a6b7b9674ef42}\ \\ Provides an interface for viewing the reverse related objects of a node and it's children (recursively).

\item[search]\refer{label000755b34afae3a5297ee9ffed0135a0}\ \\ Provides the standard search interface.

\item[tipafriend]\refer{label14e699718c339f88184702a91e40f36e}\ \\ Provides an interface to the "tip a friend" feature.

\item[translate]\refer{label631ad7df96c7d8b102025994aae62c1a}\ \\ Provides an interface for the translation of a node (DEPRECATED).

\item[translations]\refer{label1f549dd9bb48196c3e4a5e202cd1f976}\ \\ Provides an interface for managing content translations.

\item[trash]\refer{label5ac1420adc9f615eca107c8c8978b628}\ \\ Provides an interface for managing the trash.

\item[treemenu]\refer{labelf4f7301a2e26f4c3b50837269e3f8ca8}\ \\ Fetches a node's list of children for the dynamic content structure tree menu.

\item[upload]\refer{label69294a4b108e4b650e6b5e527f17d9a2}\ \\ Provides an interface for uploading a file which will become a node.

\item[urlaliases]\refer{labelab53df18fe1714dd06c1e0bdaa1d889c}\ \\ Provides an interface for managing virtual URLs of the content nodes.

\item[urltranslator]\refer{label31fcdc686e3d5abcf1e85b2195b7c035}\ \\ Provides an interface for managing virtual URLs.

\item[versions]\refer{labele80cc2d71c96d8a0ee5635078e4646da}\ \\ Provides an interface for managing the versions of an object (DEPRECATED).

\item[versionview]\refer{label10ec41fffbab66d930d4d689de4c50e0}\ \\ Provides an interface for viewing a version of an object.

\item[view]\refer{labelaf54aa865627eb5f948fc8ca5c382c99}\ \\ Provides an interface for viewing a node.

5

\end{description}

action

Summary

Provides an interface to different actions (AddToBasket, SwapNode, etc.).

5

advancedsearch

Summary

Provides the advanced search interface.

5

bookmark

Summary

Provides an interface for managing the current user's bookmarks.

browse

Summary

Provides an interface for selecting node(s) by browsing the node tree.

5

collectedinfo

Summary

Provides an interface for displaying the information that was collected.

5

collectinformation

Summary

Provides an interface for collecting information.

copy

Summary

Provides an interface for copying a single node.

copysubtree

Summary

Provides an interface for copying an entire subtree of nodes.

5

diff

Summary

Provides an interface for comparing two different versions of an object (DEPRECATED).

5

download

Summary

Provides an interface for downloading files stored by the "File" datatype.

5

draft

Summary

Provides an interface for managing the current user's drafts.

5

edit

Summary

Provides an interface for editing and translating the contents of objects.

5

hide

Summary

Provides an interface for hiding and revealing nodes.

5

history

Summary

Provides an interface for managing the versions of an object and comparing two different versions of an object.

5

keyword

Summary

Loads a template that can fetch objects which use keyword.

5

move

Summary

Provides an interface for changing the location of a node.

new

Summary

Loads a template that can be used to display new content since last visit.

5

pdf

Summary

Provides on-the-fly PDF generation of a node (DEPRECATED).

pendinglist

Summary

Provides an overview of the current user's pending items.

5

removeassignment

Summary

Provides an interface for removing node assignments.

5

removeeditversion

Summary

Provides an interface for draft removal.

removenode

Summary

Provides an interface for removing nodes.

5

removeobject

Summary

Provides an interface for removing objects.

5

restore

Summary

Provides an interface for restoring objects from the trash.

5

reverserelatedlist

Summary

Provides an interface for viewing the reverse related objects of a node and it's children (recursively).

5

search

Summary

Provides the standard search interface.

tipafriend

Summary

Provides an interface to the "tip a friend" feature.

5

translate

Summary

Provides an interface for the translation of a node (DEPRECATED).

5

translations

Summary

Provides an interface for managing content translations.

5

trash

Summary

Provides an interface for managing the trash.

treemenu

Summary

Fetches a node's list of children for the dynamic content structure tree menu.

5

upload

Summary

Provides an interface for uploading a file which will become a node.

5

urlaliases

Summary

Provides an interface for managing virtual URLs of the content nodes.

5

urltranslator

Summary

Provides an interface for managing virtual URLs.

5

versions

Summary

Provides an interface for managing the versions of an object (DEPRECATED).

versionview

Summary

Provides an interface for viewing a version of an object.

5

view

Summary

Provides an interface for viewing a node.

5

5.4.4 error

Summary

Provides an interface for error handling / reporting.

Description

This module is used internally by other modules that wish to report/display error messages. The view(s) that hte module provides can be used when developing/creating a custom module. It doesn't provide any direct/usable functionality when it comes to building a site (templatework only) with eZ publish.

5

5.4.5 ezinfo

Summary

Provides views for displaying information about eZ publish.

Description

This is a small module that makes it possible to extract some generic information about the system. It doesn't interface with any engines inside the kernel. What it does is that it provides three views: "about", "copyright" and "is_alive". The "about" view returns information about eZ publish, the "copyright" view returns copyright information (related to eZ publish) and the "is_alive" view checks the database connection and if everything seems to be okay, it returns the text "eZ publish is alive!".

The module components are documented in the following sections:

- Views (page)

5

Views

about   (page 825)
    Provides information about the system (version number, etc.).

copyright   (page 826)
    Provides copyright information related to eZ publish.

is_alive   (page 827)
    Provides information about the database connection.

5

about

Summary

Provides information about the system (version number, etc.).

5

copyright

Summary

Provides copyright information related to eZ publish.

5

is_alive

Summary

Provides information about the database connection.

5

5.4.6   form

Summary

Provides a view that generates an E-mail containing the data that was posted (DEPRECATED).

Description

This module has been deprecated and should not be used. It is strongly recommended to use the information collection (page 135) feature instead.

This module is one of the simplest modules in eZ Publish. It only contains one view, which is the "process" view. This view can only be called using HTTP POST (form action). The process view simply sends out an e-mail containing the data that was posted using the input fields. This module does not act as an interface to an engine inside the eZ Publish kernel. It only provides a simple form processing mechanism.

Note that the view provided by this module is insecure by design. It is possible to specify the sender's and the receiver's E-mail address using certain hidden input fields. In other words, it can be easily exploited by spammers. That's why this module is disabled by default. It can be enabled by setting "Module=enabled" in the [FormProcessSettings] block within an configuration override file for "site.ini". Hint: by commenting out the sender/receiver lines within the switch statement in "/kernel/form/process.php" it is possible to make this module much more secure than it is by default.

The module components are documented in the following sections:

- Views (page 829)

5

Views

process   (page 830)
>   Provides an interface for creating an E-mail based on form data (DEPRECATED).

5

process

Summary

Provides an interface for creating an E-mail based on form data (DEPRECATED).

5

5.4.7   infocollector

Summary

Provides views for managing collected information.

Description

This module provides various interfaces that can be used to inspect/view and delete information that was collected by content objects.

The module components are documented in the following sections:

- Views (page 832)

Views

collectionlist   (page 833)
  Provides an interface for viewing and removing the collections of an object.

overview   (page 834)
  Provides an interface for viewing objects that have collected information.

view   (page 835)
  Provides an interface for viewing and deleting a specific collection.

5

collectionlist

Summary

Provides an interface for viewing and removing the collections of an object.

5

overview

Summary

Provides an interface for viewing objects that have collected information.

5

view

Summary

Provides an interface for viewing and deleting a specific collection.

5

5.4.8   layout

Summary

Provides a view that makes it possible to use alternative pagelayouts.

Description

This module has only one view, "set". The view can be used to force the system to make use of a different pagelayout template than the default one (pagelayout.tpl). It simply takes care of setting the variable that eZ publish uses when picking out the pagelayout template file.

When finished, the module that is specified (after the layout part in the URL) will be executed. In other words, the layout module offers a prefix/prerun-mechanism that can be used to specify which pagelayout eZ publish should use when rendering a specific page. For example, it can be used to set the printer-friendly layout:

```
http://www.example.com/layout/set/print/content/view/full/45
```

Note that the URL actually contains two module-view pairs. The first module/view combination will make sure that the print pagelayout is used, the second module/view combination instructs eZ publish to display a full view of node number 45. The result will be the following: eZ publish will render the full view of node number 45, but instead of using "pagelayout.tpl" as the main template, it will use the pagelayout that is associated with "print" in the configuration override for "layout.ini".

The module components are documented in the following sections:

- Fetch functions (page 837)

- Views (page 839)

5

Fetch functions

sitedesign_list   (page 838)
        DEPRECATED (Fetches the names of the currently used designs.)

5

sitedesign_list

Summary

DEPRECATED (Fetches the names of the currently used designs.)

Usage

```
fetch( 'layout', 'sitedesign_list' )
```

Returns

An array of strings containing the design names.

Description

This function fetches the different designs that are used by the current siteaccess. The function returns an array of strings containing the names of the designs.

Examples

Example 1

```
{def $designs=fetch( 'layout', 'sitedesign_list' )}

{foreach $designs as $design}
    {$design} <br />
{/foreach}
```

Outputs the names of the designs that are used by the current siteaccess.

5

Views

set   (page 840)
    Forces the system to use an alternate pagelayout.

5

set

Summary

Forces the system to use an alternate pagelayout.

5

5.4.9   notification

Summary

Provides an interface to the notification engine.

Description

This module provides an interface to the notification engine inside the eZ publish kernel. The administration interface makes use the views that this module provides in order to allow the management of notifications (add new, remove, edit, etc.). Although possible, it isn't common to use these views when building a website (unless there is a need to replicate the notification management functionality of the administration interface).

The module components are documented in the following sections:

\begin{itemize} \item{} Fetch functions\refer{label0f729e52a587528ef7572589c6151850} \item{} Views\refer{labelc8dab8837ef6bc801d604973c1cbac80} \end{itemize}

5

Fetch functions

digest_handlers   (page 843)
> Fetches the notification handlers for the notification items that should be sent as digest to the current user.

digest_items   (page 844)
> Fetches notification items that should be sent as digest to the current user.

event_content   (page 845)
> Fetches the contents of the notification event.

handler_list   (page 846)
> Fetches the available notification handlers.

subscribed_nodes   (page 847)
> Fetches nodes that the current user has subscribed to.

subscribed_nodes_count   (page 848)
> Fetches the number of nodes that the current user has subscribed to.

5

digest_handlers

Summary

Fetches the notification handlers for the notification items that should be sent as digest to the current user.

5

digest_items

Summary

Fetches notification items that should be sent as digest to the current user.

event_content

Summary

Fetches the contents of the notification event.

5

handler_list

Summary

Fetches the available notification handlers.

Usage

```
fetch( 'notification', 'handler_list' )
```

Returns

An array of notification handler objects.

Description

This function fetches all the available notification handlers and returns an array containing handler objects.

Examples

Example 1

```
{def $handlers=fetch( 'notification', 'handler_list' )}
{foreach $handlers as $handler}
    {$handler.id_string} <br />
{/foreach}
```

Outputs the identification string of all the available notification handlers.

5

subscribed_nodes

Summary

Fetches nodes that the current user has subscribed to.

Usage

```
fetch( 'notification', 'subscribed_nodes',
       hash( [ 'offset', offset, ]
             [ 'limit',  limit  ] ) )
```

Parameters

Name	Type	Description	Required
offset	integer	The offset to start at.	No.
limit	integer	The number of nodes that should be fetched.	No.

Returns

Array of ezsubtreenotificationrule (page 1114) objects or FALSE.

Description

This function fetches the nodes that the current user has subsribed to. The "offset" and "limit" parameters are optional. The function returns an array of ezsubtreenotificationrule (page 1114) objects or FALSE if no nodes could be found.

Examples

Example 1

```
{def $subscriptions=fetch( 'notification', 'subscribed_nodes' )}

{foreach $subscriptions as $subscription}
    {$subscription.node.name|wash} <br />
{/foreach}
```

Outputs the names of the nodes that the current user has subscribed to.

subscribed_nodes_count

Summary

Fetches the number of nodes that the current user has subscribed to.

Usage

```
fetch( 'notification', 'subscribed_nodes_count' )
```

Returns

The number of nodes that the current user has subscribed to (as an integer).

Description

This function fetches and returns the number of nodes that the current user has subsribed to.

Examples

Example 1

```
{def $node_count=fetch( 'notification', 'subscribed_node_count' )}
The current user has subscribed to {$node_count} number of nodes.
```

Outputs the number of nodes that the current user has subscribed to.

5

Views

addtonotification   (page 850)

   Provides a mechanism that adds a new subtree notification.

runfilter   (page 851)

   Provides an interface for launching the main notification processing script and for generating a new time event.

settings   (page 852)

   Provides an interface for tweaking user notification settings.

5

addtonotification

Summary

Provides a mechanism that adds a new subtree notification.

5

runfilter

Summary

Provides an interface for launching the main notification processing script and for generating a new time event.

5

settings

Summary

Provides an interface for tweaking user notification settings.

5.4.10   package

Summary

Provides views for importing/exporting packages.

Description

This module provides an interface to the packaging system. It makes it possible to view, import, export, install, uninstall, etc. eZ publish packages. The views that this module provides are used within the setup part of the administration interface.

The module components are documented in the following sections:

- Fetch functions (page 854)

- Views (page 868)

5

Fetch functions

can_create   (page [855](#))
        Not documented yet.

can_edit   (page [856](#))
        Not documented yet.

can_export   (page [857](#))
        Not documented yet.

can_import   (page [858](#))
        Not documented yet.

can_install   (page [859](#))
        Not documented yet.

can_list   (page [860](#))
        Not documented yet.

can_read   (page [861](#))
        Not documented yet.

can_remove   (page [862](#))
        Not documented yet.

dependent_list   (page [863](#))
        Not documented yet.

item   (page [864](#))
        Not documented yet.

list   (page [865](#))
        Not documented yet.

maintainer_role_list   (page [866](#))
        Not documented yet.

repository_list   (page [867](#))
        Not documented yet.

5

can_create

Summary

Not documented yet.

5

can_edit

Summary

Not documented yet.

5

can_export

Summary

Not documented yet.

5

can_import

Summary

Not documented yet.

5

can_install

Summary

Not documented yet.

can_list

Summary

Not documented yet.

5

can_read

Summary

Not documented yet.

can_remove

Summary

Not documented yet.

5

dependent_list

Summary

Not documented yet.

5

item

Summary

Not documented yet.

list

Summary

Not documented yet.

maintainer_role_list

Summary

Not documented yet.

repository_list

Summary

Not documented yet.

5

Views

create   (page 869)
> Not documented yet.

export   (page 870)
> Not documented yet.

install   (page 871)
> Not documented yet.

list   (page 872)
> Not documented yet.

uninstall   (page 873)
> Not documented yet.

upload   (page 874)
> Not documented yet.

view   (page 875)
> Not documented yet.

5

create

Summary

Not documented yet.

5

export

Summary

Not documented yet.

5

install

Summary

Not documented yet.

5

list

Summary

Not documented yet.

5

uninstall

Summary

Not documented yet.

5

upload

Summary

Not documented yet.

5

view

Summary

Not documented yet.

5

5.4.11   pdf

Summary

Provides views for configuring PDF exports (DEPRECATED).

Description

This module provides an interface to the PDF (Portable Document Format) engine inside the kernel. This module has a couple of views that make it possible to configure PDF exports. The views provided by this module are used within the setup part of the administration interface.

The module components are documented in the following sections:

- Views (page 877)

5

Views

edit   (page 878)
      Provides an interface for editing PDF exports (DEPRECATED).

list   (page 879)
      Provides an interface for generating an overview of the PDF exports (DEPRECATED).

5

edit

Summary

Provides an interface for editing PDF exports (DEPRECATED).

5

list

Summary

Provides an interface for generating an overview of the PDF exports (DEPRECATED).

5

5.4.12   reference

Summary

Provides a view for displaying documentation generated by Doxygen.

Description

This module provides a view for displaying documentation generated by Doxygen.

5

5.4.13   role

Summary

Provides views for managing roles.

Description

This module provides an interface to the permission system inside the eZ Publish kernel. It contains views that can be used to manage roles, role assignments, policies and so on. These views are used by the administration interface.

The module components are documented in the following sections:

\begin{itemize}   \item{}   Views\refer{labeledd1e1c02627b589671faddee40a03ec}
\end{itemize}

5

Views

assign   (page [888](#888))
    Provides an interface for assigning roles to users and user groups.

copy   (page [887](#887))
    Provides an interface for copying roles.

edit   (page [886](#886))
    Provides an interface for editing roles.

list   (page [885](#885))
    Provides an interface for generating an overview of all available roles.

policyedit   (page [884](#884))
    Provides an interface for editing policies.

view   (page [883](#883))
    Provides an interface for viewing a role.

5

view

Summary

Provides an interface for viewing a role.

5

policyedit

Summary

Provides an interface for editing policies.

list

Summary

Provides an interface for generating an overview of all available roles.

edit

Summary

Provides an interface for editing roles.

5

copy

Summary

Provides an interface for copying roles.

assign

Summary

Provides an interface for assigning roles to users and user groups.

5

5.4.14   rss

Summary

Provides views for managing RSS imports and exports.

Description

This module provides an interface to the RSS (Really Simple Syndication) engine inside the eZ Publish kernel. The views that this module provide are used by the administration interface. These views make it possible to view and mange incoming and outgoing RSS feeds.

The module components are documented in the following sections:

- Views (page 890)

5

Views

edit_export   (page 891)
   Provides an interface for editing an RSS export.

edit_import   (page 892)
   Provides an interface for editing an RSS import.

feed   (page 893)
   Not documented yet.

list   (page 894)
   Provides an interface for generating an overview of RSS imports and exports.

5

edit_export

Summary

Provides an interface for editing an RSS export.

edit_import

Summary

Provides an interface for editing an RSS import.

feed

Summary

Not documented yet.

list

Summary

Provides an interface for generating an overview of RSS imports and exports.

5.4.15   search

Summary

Provides a view that displays search statistics.

Description

This module provides only a single view which is the "stats" view. The view can be used to display and reset the search statistics. The search module itself doesn't take care of indexing words, etc. This is done from within the content engine when objects are published.

The module components are documented in the following sections:

- Fetch functions (page 896)

- Views (page 900)

5

Fetch functions

list   (page 897)
    Fetches the phrases that have been searched ordered by their usage frequency.

list_count   (page 899)
    Fetches the number of unique phrases that have been searched.

5

list

Summary

Fetches the phrases that have been searched ordered by their usage frequency.

Usage

```
fetch( 'search', 'list',
       hash( [ 'offset', offset, ]
             [ 'limit',  limit  ] ) )
```

Parameters

Name	Type	Description	Required
offset	integer	The offset to start at.	No.
limit	integer	The number of items that should be returned.	No.

Returns

An array of hashes (see below) or FALSE.

Description

This function fetches the phrases that have been searched on the site. The phrases are ordered by their usage frequency. If two or more phrases have the same usage frequency, they are ordered by the number of nodes that match the search conditions.

The "offset" and "limit" parameters are optional and can be used to limit the result. If the "offset" and "limit" parameters are omitted, the function will return all the search phrases.

The function returns an array of hashes. The hashes consist of the following elements:

Key	Type	Description
id	string	The ID number of the search phrase.
phrase	string	The text of the search phrase.
phrase_count	string	The number of times the phrase has been searched for.
result_count	string	The number of nodes that match the search conditions (average search results that have been returned).

If no search phrases are found, the function returns FALSE.

5

Examples

Example 1

```
{def $search_list=fetch( 'search', 'list' ) )}
{foreach $search_list as $s}
        {$s.id } : {$s.phrase}<br />
{/foreach}
```

Outputs all phrases that have been searched for along with their ID numbers.

Example 2

```
{def $search_list=fetch( 'search', 'list', hash( 'limit','10'  ) )}
{foreach $search_list as $s}
        {$s.phrase } : {$s.phrase_count}<br />
{/foreach}
```

Outputs the ten most frequently searched phrases and their usage frequency.

5

list_count

Summary

Fetches the number of unique phrases that have been searched.

Usage

```
fetch( 'search', 'list_count' )
```

Returns

The number of phrases that have been searched (as an integer).

Description

This function fetches the number of unique phrases that have been searched and returns it as an integer.

Examples

Example 1

```
{def $count=fetch( 'search', 'list_count' )}
{$count} phrases were searched on this site.
```

Outputs the number of unique phrases that have been searched on the site.

5

Views

stats   (page 901)

> Provides an interface for viewing and resetting the search statistics.

stats

Summary

Provides an interface for viewing and resetting the search statistics.

5.4.16   section

Summary

Provides views for managing sections.

Description

This module provides an interface to the content engine inside the eZ Publish kernel. The section mechanism can be used to create groups of objects. Please refer to the "sections" (page 132) part of the "Concepts and basics" chapter for more information about sections. This module provides views that are used by the administration interface.

The module components are documented in the following sections:

- Fetch functions (page 903)
- Views (page 913)

5

Fetch functions

object   (page [904](#))
     Fetches a section.

object_list   (page [906](#))
     Fetches objects that belong to certain section.

object_list_count   (page [908](#))
     Fetches the number of objects that belong to certain section.

roles   (page [910](#))
     Fetches roles that have at least one policy limited to a certain section.

user_roles   (page [912](#))
     Fetches users and/or user groups with role limitations related to a certain section.

5

object

Summary

Fetches a section.

Usage

```
fetch( 'section', 'object', hash( 'section_id', section_id, 'identifier',
identifier ) )
```

Parameters

Name	Type	Description	Required
section_id	integer	The ID number of the section that should be fetched.	No.
identifier	string	The identifier of the section that should be fetched. This parameter was added in eZ Publish 4.4.	No.

Returns

An ezsection (page 1111) object or FALSE.

Description

This function fetches an object that represents a section. A parameter is required but there should be one and only one parameter in this function. If no parameter is provided or both "section_id" and "identifier" are specified, you will get a FALSE return. The "section_id" parameter must be a valid section ID number. The "identifier" parameter must be a valid section identifier string. The function returns an ezsection (page 1111) object. If an invalid ID number or invalid identifier string is specified, the function will return FALSE.

Examples

Example 1

```
{def $section=fetch( 'section', 'object', hash( 'section_id', 13 ) )}
{$section.name|wash}
```

Outputs the name of section number 13.

5

Example 2

```
{def  $section=fetch( 'section', 'object', hash( 'identifier',  'standard' )
)}{$section.name|wash}
```

Outputs the name of section identifier 'standard'.

object_list

Summary

Fetches objects that belong to certain section.

Usage

```
fetch( 'section', 'object_list',
       hash( 'section_id', section_id,
           [ 'offset',     offset,    ]
           [ 'limit',      limit,     ]
           [ 'sort_order', sort_order ]
           [ 'status',     status     ] ) )
```

Parameters

Name	Type	Description	Required
section_id	integer	The ID number of the target section.	Yes.
offset	integer	The offset to start at.	No.
limit	integer	The number of objects that should be fetched.	No.
sort_order	array	The desired sorting order.	No.
status	string	The status of the target objects ('published' by default).	No.

Returns

An array of ezcontentobject (page 1032) objects or FALSE.

Description

This function fetches a set of objects that belong to a certain section. The section must be specified by its ID number using the "section_id" parameter. The "offset", "limit" and "sort_order" parameters work in the same way as in the list (page 714) fetch function of the "content" module.

The optional "status" parameter can be used to specify the status of objects that should be fetched. If this parameter is omitted or set to "published", the function will fetch all published objects that belong to the specified section. If set to "archived", the archived (moved to the trash) objects that belong to the specified section will be fetched. (Note that fetching all draft objects that belong to a certain section is not supported.)

The function returns an array of ezcontentobject (page 1032) objects. If no objects can be found, or if the provided section ID number is invalid, FALSE will be returned.

5

Examples

Example 1

```
{def $objects=fetch( 'section', 'object_list',
                     hash( 'section_id', 13 ) )}

{foreach $objects as $object}
{$object.name} <br />
{/foreach}
```

Outputs the names of the published objects that belong to section number 13.

Example 2

```
{def $objects=fetch( 'section', 'object_list',
                     hash( 'section_id', 5,
                           'status', 'archived' ) )}

{foreach $objects as $object}
{$object.name} <br />
{/foreach}
```

Outputs the names of the objects that belong to section number 5 and are moved to the trash.

5

object_list_count

Summary

Fetches the number of objects that belong to certain section.

Usage

```
fetch( 'section', 'object_list_count',
        hash( 'section_id', section_id,
            [ 'status',    status    ] ) )
```

Parameters

Name	Type	Description	Required
section_id	integer	The ID number of the target section.	Yes.
status	string	The status of the target objects ('published' by default).	No.

Returns

The number of objects (as an integer) that belong to the section.

Description

This function counts the number of objects that belong to a section specified by the "section_id" parameter. The optional "status" parameter works in the same way as in the "object_list (page 906)" fetch function.

The function returns the count as an integer.

Examples

Example 1

```
{def $count=fetch( 'section', 'object_list_count',
                hash( 'section_id', 13 ) )}

There are {$count} number of  objects in section number 13.
```

Outputs the number of published objects that belong to section number 13.

5

Example 2

```
{def $count=fetch( 'section', 'object_list_count',
                   hash( 'section_id', 5,
                         'status', 'archived' ) )}

There are {$count} number of archived objects in section number 5.
```

Outputs the number of objects that belong to section number 5 and are moved to the trash.

5

roles

Summary

Fetches roles that have at least one policy limited to a certain section.

Usage

```
fetch( 'section', 'roles', hash( 'section_id', section_id ) )
```

Parameters

Name	Type	Description	Required
section_id	integer	The ID number of the target section.	Yes.

Returns

An array (see below) or FALSE.

Description

This function returns a structure that contains information about roles which have at least one policy limited to a certain section. The function returns an array with two keys:

Name	Description
roles	Contains a list of roles with at least one policy limited to the given section.
limited_policies	Contains a 2D array (the role ID as the first key) of the limited policies.

Examples

Example 1

```
{def $roles_array=fetch( 'section', 'roles', hash( 'section_id', 13 ) )
    $roles=$roles_array.roles
    $policies=$roles_array.limited_policies}

{foreach $roles as $role}
    {$role.name}:
    {foreach $limited_policies[$role.id] as $policy}
        {$policy.module_name}/{$policy.function_name}
        {delimiter}, {/delimiter}
```

5

```
    {/foreach}
    <br />
{/foreach}
```

Outputs information about roles that have limitations associated with section number 13.

5

user_roles

Summary

Fetches users and/or user groups with role limitations related to a certain section.

Usage

```
fetch( 'section', 'user_roles', hash( 'section_id', section_id ) )
```

Parameters

Name	Type	Description	Required
section_id	integer	The ID number of the target section.	Yes.

Returns

An array (see below) or FALSE.

Description

This function fetches user and/or user groups that have role limitations associated with a certain section. The section must be defined using a valid section ID number through the "section_id" parameter. Please refer to the example below to see how the returned information can be used.

Examples

Example 1

```
{def $user_roles=fetch( 'section', 'user_roles', hash( 'section_id', 13 ) )}

{foreach $user_roles as $user_role}
    User (or group) {$user_role.user.name} has limitation on the section 13.
    The "touched" role is {$user_role.role.name}. <br />
{/foreach}
```

5

Views

assign   (page 917)
    Provides an interface for assigning objects to a section.

edit   (page 916)
    Provides an interface for editing a section.

list   (page 915)
    Provides an interface for generating an overview of all the available sections.

view   (page 914)
    Provides an interface for viewing a section.

5

view

Summary

Provides an interface for viewing a section.

list

Summary

Provides an interface for generating an overview of all the available sections.

5

edit

Summary

Provides an interface for editing a section.

assign

Summary

Provides an interface for assigning objects to a section.

5

5.4.17   setup

Summary

Provides the web based setup wizard.

Description

This module provides the views for the web based setup wizard.

5

5.4.18   shop

Summary

Provides views for the webshop (basket, wish list, order list, etc.).

Description

This module provides an interface to the content and the e-commerce engine inside the eZ Publish kernel. The views that this module offers are typically used when building a webshop oriented site. This module also provides some views that are used by the administration interface.

Note: From 4.3 onwards the eZ Webshop tab is disabled by default in the eZ Publish installation. Follow this link (page 2165) to see how you enable the eZ Webshop functionality in eZ Publish.

The module components are documented in the following sections:

- Fetch functions (page 920)
- Views (page 936)

Fetch functions

basket   (page 921)
> Fetches the current user's shopping basket.

best_sell_list   (page 922)
> Fetches the most popular / most sold products.

currency   (page 925)
> Fetches a currency object.

currency_list   (page 926)
> Fetches the available currencies.

preferred_currency_code   (page 928)
> Fetches the current user's preferred currency.

product_category   (page 929)
> Fetches a product category.

product_category_list   (page 930)
> Fetches the available product categories.

related_purchase   (page 931)
> Fetches products that were purchased together with a given product.

wish_list   (page 933)
> Fetches the products of a given wishlist.

wish_list_count   (page 935)
> Fetches a wishlist and returns the number of items in it.

5

basket

Summary

Fetches the current user's shopping basket.

Usage

```
fetch( 'shop', 'basket')
```

Returns

An ezbasket (page 1016) object.

Description

This function fetches the shopping basket that belongs to the current user. The function returns an ezbasket (page 1016) object.

Examples

Example 1

```
{def $basket=fetch( 'shop', 'basket' )}

{if $basket.is_empty}

    There are no products in the basket.

{else}

    There are {count( $basket.items )} items in the basket. <br />
    Total price (ex. VAT) :  {$basket.total_ex_vat|l10n( currency )} <br />
    Total price (inc. VAT): {$basket.total_inc_vat|l10n( currency )}

{/if}
```

Outputs basic information about the current user's shopping basket.

5

best_sell_list

Summary

Fetches the most popular / most sold products.

Usage

```
fetch( 'shop', 'best_sell_list',
       hash( [ 'top_parent_node_id', parent_node_id, ]
             [ 'offset',             offset,         ]
             [ 'limit',              limit,          ]
             [ 'start_time',         start_time,     ]
             [ 'end_time',           end_time,       ]
             [ 'duration',           duration,       ]
             [ 'ascending',          ascending,      ]
             [ 'extended',           extended        ] ) )
```

Parameters

Name	Type	Description	Required
top_parent_node_id	integer	The ID number of the top node.	Yes.
limit	integer	The number of objects that should be returned.	No.
offset	integer	The offset to start at.	No.
start_time	integer	The time to start at.	No.
end_time	integer	The time to end at.	No.
duration	integer	The period of time that should be used.	No.
ascending	boolean	The sorting direction that should be used. If FALSE (default), the results will be sorted descending by the number of times each product was bought.	No.
extended	boolean	If TRUE, an array of hashes will be returned.	No.

Returns

An array of ezcontentobject (page 1032) objects, an array of hashes (see below) or FALSE.

Description

This function fetches the most popular / most sold products that are located within a specified part of the content node tree. The "top_parent_node_id" parameter must be used to tell the function under which node it should look for popular products.

The returned result is sorted by the number of times the products were bought. The sorting direction can be controlled using the "ascending" parameter, it can be set to either TRUE or FALSE (default).

The "limit" and "offset" parameters can be used to control the number of items returned and the offset to start at.

The "start_time", "end_time" and "duration" parameters can be used to specify a time period that the function should operate within. These parameters make it possible to fetch the most popular/ sold products for a certain period of time (for example last 24 hours, last week, last month, etc.). If the "start_time" and "duration" parameters are specified, results for the "start_time+duration" period of time will be fetched. If the "end_time" and "duration" parameters are specified, results for the "end_time-duration" period of time will be fetched. If all of these parameters are omitted, the function will cover all products, regardless when they were purchased. If both the "start_time" and "duration" parameters are omitted, then only results for the period before "end_time" will be fetched. If only the "start_time" or "duration" parameter is set, the function will automatically use the current time as the "end_time" parameter.

By default (when the "extended" parameter is FALSE), the function returns an array of ezcontentobject (page 1032) objects. If the "extended" parameter is set to TRUE, the function will return an array of the following hashes:

Key	Type	Description
count	integer	The number of times the product was bought.
contentobject_id	string	The ID number of the product.
object	object	The product itself (as an ezcontentobject (page 1032) object).

If the function is unable to find any products, it will return FALSE.

Examples

Example 1

```
{def $best_sellers=fetch( 'shop', 'best_sell_list',
                          hash( 'top_parent_node_id', 2,
                                'limit', 5,
                                'offset', 0,
                                'start_time', maketime( 0, 0, 0 ),
                                'duration', mul( 60, 60, 24 ) ) ) }

{foreach $best_sellers as $product}
    {$product.name}
{/foreach}
{undef}
```

5

Outputs the names of the five most popular products below node 2 during the last 24 hours. The
"start_time" parameter sets 00:00:00 as the time and the current date. The "duration" parameter
is set to a period that equals one day (60 seconds x 60 minutes x 24 hours).

Example 2

```
{def $best_sellers=fetch( 'shop', 'best_sell_list',
                          hash( 'top_parent_node_id', 2,
      'ascending', false(),
                                'extended', true() ) ) }


<table>
{foreach $best_sellers as $product}
    <tr>
    <td>{$product.object.name}</td>
    <td>{$product.object.main_node.path_identification_string}</td>
    <td>{$product.count}</td>
    </tr>
{/foreach}
</table>
{undef}
```

Generates a list of popular products by outputting the names, node paths and the number of times
they were bought. The results are sorted by the number of times the products were bought.

5

currency

Summary

Fetches a currency object.

Usage

```
fetch( 'shop', 'currency',
       hash( 'code', code ) )
```

Parameters

Name	Type	Description	Required
code	string	The three-character code of the target currency.	Yes.

Returns

An ezcurrencydata (page 1060) object or FALSE.

Description

This function fetches the currency specified by the "code" parameter and returns it as an ezcurrencydata (page 1060) object. The function will return FALSE if the specified currency is unavailable.

Examples

Example 1

```
{def $code = 'EUR'}

{def $currency = fetch( 'shop', 'currency',
                        hash( 'code', $code ) )}

{if $currency}
     The currency symbol for EUR is {$currency.symbol}. <br />
{/if}
```

Outputs the currency symbol that is used for the "EUR" currency if available.

5

currency_list

Summary

Fetches the available currencies.

Usage

```
fetch( 'shop', 'currency_list' [, hash( 'status', status )] )
```

Parameters

Name	Type	Description	Required
status	string	The status of the target currencies.	No.

Returns

An array of ezcurrencydata (page 1060) objects or FALSE.

Description

This function fetches the available currencies and returns an array of ezcurrencydata (page 1060) objects. The resulting array starts with the most recently added currency. When the optional "status" parameter is used then only currencies with the specified status will be fetched.

Examples

Example 1

```
{def $currency_list = fetch( 'shop', 'currency_list' )}
{if count( $currency_list )}
    {foreach $currency_list as $Currency}
        {$Currency.code} <br />
    {/foreach}
{else}
    There are no currencies.
{/if}
```

Outputs the currency codes for all the available currencies.

Example 2

```
{def $currency_list = fetch( 'shop', 'currency_list', hash( 'status', 'active'
) )}
{if count( $currency_list )}
    The following currencies are active: <br />
    {foreach $currency_list as $Currency}
        {$Currency.code} <br />
    {/foreach}
{else}
    There are no active currencies.
{/if}
```

Outputs the currency codes for active currencies only.

5

preferred_currency_code

Summary

Fetches the current user's preferred currency.

Usage

```
fetch( 'shop', 'preferred_currency_code' )
```

Returns

A three-character currency code or FALSE.

Description

This function fetches the preferred currency of the current user and returns a three-character currency code. A value for the preferred currency is taken from (sorted by priority ascending)

- shop.ini

- user's preferences

- session variable

The function will return FALSE if the preferred currency is not specified.

Examples

Example 1

```
{def $currency = fetch( 'shop', 'preferred_currency_code' )}
{if $currency}
    Your preferred currency is {$currency}.
{else}
    You should choose the preferred currency.
{/if}
```

Outputs the currency code of the current user's preferred currency.

product_category

Summary

Fetches a product category.

Usage

```
fetch( 'shop', 'product_category',
        hash( 'category_id', category_id ) )
```

Parameters

Name	Type	Description	Required
category_id	integer	The identifier of the target product category.	Yes.

Returns

An ezproductcategory (page 1105) object or FALSE.

Description

This function fetches the product category specified by the "category_id" parameter and returns it as an ezproductcategory (page 1105) object. The function will return FALSE if the specified product category is unavailable.

Examples

Example 1

```
{def $category = fetch( 'shop', 'product_category',
                      hash( 'category_id', 3 ) )}

{if $category}
    The product category called "{$category.name}" has id=3. <br />
{/if}
```

Outputs the name of the product category with the specified identifier.

product_category_list

Summary

Fetches the available product categories.

Usage

```
fetch( 'shop', 'product_category_list' )
```

Returns

An array of ezproductcategory (page 1105) objects or FALSE.

Description

This function fetches the available product categories and returns an array of ezproductcategory (page 1105) objects.

Examples

Example 1

```
{def $product_categories = fetch( 'shop', 'product_category_list' )}
{if count( $product_categories )}
    {foreach $product_categories as $Category}
         {$Category.name} <br />
    {/foreach}
{else}
    There are no product categories.
{/if}
```

Outputs the category names for all the available product categories.

5

related_purchase

Summary

Fetches products that were purchased together with a given product.

Usage

```
fetch( 'shop', 'related_purchase',
       hash( 'contentobject_id', id,
             'limit',           limit ) )
```

Parameters

Name	Type	Description	Required
contentobject_id	integer	The ID number of the object representing the source product.	Yes.
limit	integer	The number of objects that should be fetched.	Yes.

Returns

An array of ezcontentobject (page 1032) objects or FALSE.

Description

This function fetches a collection of products (content objects) that were purchased together with a given product. It can be used to create a "People who bought this product has also bought..." list. The source product must be specified using the "contentobject_id" parameter. This parameter must be an integer that reveals the ID number of the content object that represents the source product. The "limit" parameter must be used to limit the result. The function will return an array of ezcontentobject (page 1032) objects or FALSE if no objects were found.

Examples

Example 1

```
{def $other_products=fetch( 'shop', 'related_purchase',
                            hash( 'contentobject_id', 32,
                                  'limit',           5 ) )}

{foreach $other_products as $product}
    {$product.name} <br />
```

```
{/foreach}
```

Outputs the names of 5 products that were bought together with a product represented by object number 32.

5

wish_list

Summary

Fetches the products of a given wishlist.

Usage

```
fetch( 'shop', 'wish_list',
       hash( 'production_id', production_id,
           [ 'offset', offset, ]
           [ 'limit',  limit  ] ) )
```

Parameters

Name	Type	Description	Required
production_id	integer	The ID of the target wishlist.	Yes.
offset	integer	The offset to start at.	No.
limit	integer	The number of products that should be fetched.	No.

Returns

An array of arrays containing information about the items in the wishlist (see below) or FALSE.

Description

This function fetches the products that belong to a certain wishlist. The system stores wishlists using the same solution that is used to store the contents of shopping baskets, the "product collection" schema. The desired wishlist must be specified using the ID number of the product collection that contains the wishlist. The optional "offset" and "limit" parameters can be used to control the resulting set. The function returns an array of arrays containing information about each item in the wishlist. The following table shows the structure that is used for each element of the array.

Attribute	Type	Description
id	string	The ID number of the item in the basket.
vat_value	string	32
item_count	string	The quantity.
node_id	string	The ID number of the node that represents the item.
object_name	string	The name of the object that

		represents the item.
price_ex_vat	double	The price of the item excluding the VAT.
price_inc_vat	string	The price of the item including the VAT.
discount_percent	string	The discount percentage (if any).
total_price_ex_vat	double	The total price excluding the VAT.
total_price_inc_vat	double	The total price including the VAT.
item_object	object	The item itself (as an ezproductcollectionitem (page 1106) object).

If the system is unable to find any products, an empty wishlist / FALSE will be returned.

Examples

Example 1

```
{def $wishlist=fetch( 'shop', 'wish_list',  hash( 'production_id', 13 ) )}

{foreach $wishlist as $wish}
    {$wish.object_name|wash} <br />
{/foreach}
```

Outputs the names of the objects that make up wishlist number 13.

5

wish_list_count

Summary

Fetches a wishlist and returns the number of items in it.

Usage

```
fetch( 'shop', 'wish_list_count', hash( 'production_id', production_id ) )
```

Parameters

Name	Type	Description	Required
production_id	integer	The ID number of the target wishlist.	Yes.

Returns

The number of items that make up the wishlist (as an integer).

Description

This function fetches the products that belong to a certain wishlist and returns an integer. The integer reveals the number of items that make up the wishlist. The system stores wishlists using the same solution that is used to store the contents of shopping baskets, the "product collection" schema. The desired wishlist must be specified using the ID number of the product collection that contains the target wishlist.

Examples

Example 1

```
{def $sum=fetch( 'shop', 'wish_list_count',  hash( 'productcollection_id', 13
) )}
There are {$sum} items in wishlist number 13.
```

Outputs the number of items that make up wishlist number 13.

5

Views

basket (page 938)

Provides an interface to the shopping basket of the current user.

checkout (page 939)

Provides the checkout interface.

confirmorder (page 940)

Provides the interface that asks the user to confirm an order.

currencylist (page 941)

Provides an interface for viewing currencies and makes it possible to edit some of the currency's attributes for several currencies at the same time. Allows to update auto rates for all currencies.

customerlist (page 942)

Provides an interface for generating an overview of the customers.

customerorderview (page 943)

Provides an interface for viewing information the orders of a customer.

discountgroup (page 944)

Provides an interface for generating an overview of the discount groups.

discountgroupedit (page 945)

Provides an interface for editing a discount group.

discountgroupview (page 946)

Provides an interface for viewing a discount group.

discountruleedit (page 947)

Provides an interface for editing a discount rule.

editcurrency (page 948)

Provides an interface for editing a currency.

editvatrule (page 949)

Provides an interface for editing a VAT charging rule.

orderlist (page 950)

Provides an interface for generating an overview of orders.

orderview (page 951)

Provides an interface for viewing an order.

preferredcurrency (page 952)

Provides an interface for setting the user's preferred currency.

productcategories (page 953)

Provides an interface for viewing and managing product categories.

productsoverview   (page 954)
>   Provides an interface for generating an overview of products and allows sorting by product's name or price.

register   (page 955)
>   Provides an interface for registering a customer.

removeorder   (page 956)
>   Provides an interface for removing an order.

statistics   (page 957)
>   Provides an interface for generating sales statistics.

userregister   (page 958)
>   Provides an interface for registering a user.

vatrules   (page 959)
>   Provides an interface for generating an overview of the VAT charging rules.

vattype   (page 960)
>   Provides an interface for managing VATs.

wishlist   (page 961)
>   Provides an interface for viewing and managing the current user's wishlist.

5

basket

Summary

Provides an interface to the shopping basket of the current user.

checkout

Summary

Provides the checkout interface.

5

confirmorder

Summary

Provides the interface that asks the user to confirm an order.

5

currencylist

Summary

Provides an interface for viewing currencies and makes it possible to edit some of the currency's attributes for several currencies at the same time. Allows to update auto rates for all currencies.

5

customerlist

Summary

Provides an interface for generating an overview of the customers.

customerorderview

Summary

Provides an interface for viewing information the orders of a customer.

5

discountgroup

Summary

Provides an interface for generating an overview of the discount groups.

discountgroupedit

Summary

Provides an interface for editing a discount group.

5

discountgroupview

Summary

Provides an interface for viewing a discount group.

5

discountruleedit

Summary

Provides an interface for editing a discount rule.

5

editcurrency

Summary

Provides an interface for editing a currency.

editvatrule

Summary

Provides an interface for editing a VAT charging rule.

5

orderlist

Summary

Provides an interface for generating an overview of orders.

5

orderview

Summary

Provides an interface for viewing an order.

preferredcurrency

Summary

Provides an interface for setting the user's preferred currency.

5

productcategories

Summary

Provides an interface for viewing and managing product categories.

productsoverview

Summary

Provides an interface for generating an overview of products and allows sorting by product's name or price.

5

register

Summary

Provides an interface for registering a customer.

removeorder

Summary

Provides an interface for removing an order.

statistics

Summary

Provides an interface for generating sales statistics.

5

userregister

Summary

Provides an interface for registering a user.

vatrules

Summary

Provides an interface for generating an overview of the VAT charging rules.

vattype

Summary

Provides an interface for managing VATs.

wishlist

Summary

Provides an interface for viewing and managing the current user's wishlist.

5.4.19   trigger

Summary

Provides a view for managing workflow triggers.

Description

This module provides an interface to the workflow engine inside the eZ Publish kernel. It consists of a view that can be used to list and manage the workflows that should be triggered before or after a specific function within a specific module is executed. The administration interface makes use of this view to allow the administrator to view and manage triggers.

The module components are documented in the following sections:

- Views (page 963)

Views

list   (page 964)

Provides an interface for viewing and managing the workflow triggers.

5

list

Summary

Provides an interface for viewing and managing the workflow triggers.

5

5.4.20 url

Summary

Provides views for managing the URLs stored in the database.

Description

This module provides an interface to the content engine inside the eZ Publish kernel. Every address that is input as a link into an attribute using the XML block (page 601) or the URL (page 597) 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. Please refer to the "URL storage" (page 134) part of the "Concepts and basics" chapter for more information about this feature. The "url" module provides views that make it possible to manage the URLs. The administration interface makes use of this module to allow the users to manage the URLs.

The module components are documented in the following sections:

- Fetch functions (page 966)

- Views (page 970)

5

Fetch functions

list   (page 967)

Fetches the URLs that are stored in the URL table.

list_count   (page 969)

Fetches the number of URLs that are stored in the URL table.

5

list

Summary

Fetches the URLs that are stored in the URL table.

Usage

```
fetch( 'url', 'list', hash( [ 'is_valid', is_valid, ]
                            [ 'offset',   offset,  ]
                            [ 'limit',    limit    ] ) )
```

Parameters

Name	Type	Description	Required
is_valid	boolean	Instructs the system to only fetch valid or invalid URLs.	No.
offset	integer	The offset to start at.	No.
limit	integer	The number of URLs that should be fetched.	No.

Returns

An array of ezurl (page 1116) objects or FALSE.

Description

This function fetches URLs from the URL table.The URL table stores addresses that have been input using the URL (page 597) or the XML block (page 601) datatype. Please refer to the "URL storage" (page 134) section of the "Concepts and basics" chapter for more information about how the system handles URLs.

The "is_valid", "offset" and "limit" parameters are optional. While the "is_valid" parameter can be used to filter out only valid (TRUE) or invalid (FALSE) URLs, the "offset" and "limit" parameters can narrow down the result. If the "is_valid" parameter is omitted, both valid and invalid URLs will be fetched. The function returns an array of ezurl (page 1116) objects. If no URLs can be found, the function will return FALSE.

Examples

Example 1

5

```
{def $urls=fetch( 'url', 'list', hash( 'is_valid', true(),
                                        'offset',   0,
                                        'limit',    10 ) )}


{foreach $urls as $url}
    {$url.url} <br />
{/foreach}
```

Outputs the first ten valid URLs that are stored in the URL table.

5

list_count

Summary

Fetches the number of URLs that are stored in the URL table.

Usage

```
fetch( 'url', 'list_count', hash( [ 'is_valid', is_valid ] ) )
```

Parameters

Name	Type	Description	Required
is_valid	boolean	Instructs the system to count either valid or invalid URLs.	No.

Returns

The number of URLs (as an integer).

Description

This function fetches and counts the URLs that are stored in the URL table.The URL table stores addresses that have been input using the URL (page 597) or the XML block (page 601) datatype. Please refer to the "URL storage" (page 134) section of the "Concepts and basics" chapter for more information about how the system handles URLs.

The "is_valid" parameter is optional and can be used to filter out only valid (TRUE) or invalid (FALSE) URLs. If the "is_valid" parameter is omitted, both valid and invalid URLs will be counted. The function returns the number of found URLs as an integer.

Examples

Example 1

```
{def $valid_urls=fetch( 'url', 'list_count', hash( 'is_valid', true() ) )
Number of valid URLs: {$valid_urls}
```

Outputs the number of valid URLs.

Views

\begin{description} \item[edit]\refer{label519b6636a96eb7a5e101ffa0eee5d120}\ \\ Provides an interface for editing a URL.

\item[list]\refer{labelff299d98b3d70ecff0d55fd8b8f95fec}\ \\ Provides an interface for generating an overview of all URLs.

\item[views]\refer{labele740ec9c665e5002b0f3901fb122ea56}\ \\ Provides an interface for viewing a URL.

\end{description}

5

views

Summary

Provides an interface for viewing a URL.

5

list

Summary

Provides an interface for generating an overview of all URLs.

edit

Summary

Provides an interface for editing a URL.

5

5.4.21   user

Summary

Provides views for logging users in/out, password changing, etc.

Description

This module provides an interface to the permission system inside the eZ Publish kernel. It contains views that make it possible to log users in and out, register and activate new users, password changing, etc. A typical eZ Publish site that has login capabilities makes use the views that this module provides.

The module components are documented in the following sections:

\begin{itemize} \item{} Fetch functions\refer{label556b1c0de584d0275b2ed8dcf9a8048c} \item{} Views\refer{label870a118a1f98ec1df154e81a3ea0a242} \end{itemize}

5

Fetch functions

anonymous_count   (page 976)
    Fetches the number of anonymous users.

current_user   (page 977)
    Fetches the user that is currently logged in.

has_access_to   (page 978)
    Checks if a user has access to a certain function of a module.

is_logged_in   (page 980)
    Checks if a specific user is logged in.

logged_in_count   (page 981)
    Fetches the number of users that are logged in.

logged_in_list   (page 982)
    Fetches the names of the users that are logged in.

logged_in_users   (page 984)
    Fetches the users that are logged in.

member_of   (page 986)
    Fetches the roles that are assigned to a user.

user_role   (page 987)
    Fetches the policies that are available for a user.

5

anonymous_count

Summary

Fetches the number of anonymous users.

Usage

```
fetch( 'user', 'anonymous_count' )
```

Returns

The number of anonymous users as an integer.

Description

This function counts the number of anonymous users currently accessing the site and returns that count (as an integer). An anonymous user is considered to be active if the last access time within the range of the activity timeout. The timeout can be set using the "ActivityTimeout" directive in a configuration override for "site.ini". The default timeout is one hour.

Examples

Example 1

```
{def $visitors=fetch( 'user', 'anonymous_count' )}
There are {$visitors} anonymous users accessing the site.
```

Outputs the number of anonymous users that are currently accessing the site.

5

current_user

Summary

Fetches the user that is currently logged in.

Usage

```
fetch( 'user', 'current_user' )
```

Returns

An ezuser (page 1117) object.

Description

This function fetches the user object for the user that is currently logged in. If no user is logged in, the anonymous user will be returned. In both cases, the function will return an ezuser (page 1117) object.

Examples

Example 1

```
{def $user=fetch( 'user', 'current_user' )}
User:     {$user.contentobject.name} <br />
E-mail:   {$user.email} <br />
Username: {$user.login} <br />
Group(s): {$user.groups|implode(', ')} <br />
```

Outputs miscellaneous information about the user that is currently logged in.

5

has_access_to

Summary

Checks if a user has access to a certain function of a module.

Usage

```
fetch( 'user', 'has_access_to',
        hash( 'module',   module
              'function', function,
   [ 'user_id',  user_id ] ) )
```

Parameters

Name	Type	Description	Required
module	string	The name of the module.	Yes.
function	string	The name of the function.	Yes.
user_id	integer	The ID number of the user.	No.

Returns

TRUE if access is allowed, FALSE otherwise.

Description

This function checks if the current user has access to a certain function of a module. The name of the module and the function must be provided using the "module" and the "function" parameters. The optional "user_id" parameter can be used to check access for other users than the current user. The function returns TRUE if access is allowed, otherwise FALSE will be returned.

Examples

Example 1

```
{def $access=fetch( 'user', 'has_access_to',
                    hash( 'module',   'content',
                          'function', 'read',
                          'user_id',  128 ) )}

{if $access}
    Access is allowed.
{else}
```

```
    Access is denied.
{/if}
```

Reveals if user number 128 has access to the read function of the content module.

is_logged_in

Summary

Checks if a specific user is logged in.

Usage

```
fetch( 'user', 'is_logged_in', hash( 'user_id', user_id ) )
```

Parameters

Name	Type	Description	Required
user_id	integer	The ID number of the user that should be checked.	Yes.

Returns

TRUE if the specified user is logged in, FALSE otherwise.

Description

This function checks if a user is logged in or not. The desired user's ID number must be specified using the "user_id" parameter. The ID number of a user is the same as the ID number of the content object that represents that user. A user is considered to be active / logged in if the last access time is within the range of the activity timeout. The timeout can be set using the "ActivityTimeout" directive in a configuration override for "site.ini". The default timeout is one hour.

Examples

Example 1

```
{def $test=fetch( 'user', 'is_logged_in', hash( 'user_id', 256 ) )}

{if $test}
    User number 256 is currently logged in.
{else}
    User number 256 is not logged in.
{/if}
```

Outputs information that reveals whether user number 256 is logged in or not.

5

logged_in_count

Summary

Fetches the number of users that are logged in.

Usage

```
fetch( 'user', 'logged_in_count' )
```

Returns

The number of logged in users (as an integer).

Description

This function counts the number of logged in users (both anonymous and non-anonymous) and returns that count as an integer. A user is considered to be active / logged in if the last access time is within the range of the activity timeout. The timeout can be set using the "ActivityTimeout" directive in a configuration override for "site.ini". The default timeout is one hour.

Examples

Example 1

```
{def $users=fetch( 'user', 'logged_in_count' )}
There are currently {$users} active users on the system.
```

Outputs the number of currently active / logged in users.

5

logged_in_list

Summary

Fetches the names of the users that are logged in.

Usage

```
fetch( 'user', 'logged_in_list',
       hash( [ 'sort_by', sort_by, ]
             [ 'offset',  offset, ]
             [ 'limit',   limit   ] ) )
```

Parameters

Name	Type	Description	Required
sort_by	mixed	The field that should be used by the sorting mechanism.	No.
offset	integer	The offset to start at.	No.
limit	integer	The number of users that should be fetched.	No.

Returns

An associative array or FALSE.

Description

This function will fetch all the logged in users and return an associative array. The keys of the returned hash will be the user ID numbers; the values will be the users' names. If no users are logged in, FALSE will be returned. The "sort_by", "offset" and "limit" parameters are optional.

A user is considered to be active / logged in if the last access time is within the range of the activity timeout. The timeout can be set using the "ActivityTimeout" directive in a configuration override for "site.ini". The default timeout is one hour.

The "sort_by" parameter must be specified as an array. Each element of the array must be another array where the first element denotes the field (as a string) that the sorting mechanism should use. The second element specifies the direction of the sort (as a boolean). The following sorting fields can be used:

- user_id

- login

- activity

5

- email

Examples

Example 1

```
{def $users=fetch( 'user', 'logged_in_list',
                   hash( 'sort_by', array( array( 'login', true() ) ) ) )}

{foreach $users as $user}
    {$user} <br />
{/foreach}
```

Outputs the names of the users that are currently logged in (sorted by usernames).

5

logged_in_users

Summary

Fetches the users that are logged in.

Usage

```
fetch( 'user', 'logged_in_users',
       hash( [ 'sort_by', sort_by, ]
             [ 'offset',  offset,  ]
             [ 'limit',   limit    ] ) )
```

Parameters

Name	Type	Description	Required
sort_by	mixed	The field that should be used by the sorting mechanism.	No.
offset	integer	The offset to start at.	No.
limit	integer	The number of users that should be fetched.	No.

Returns

An array with ezuser (page 1117) objects or FALSE.

Description

This function will fetch all the logged in users and return an array containing ezuser (page 1117) objects. If no users are logged in, FALSE will be returned. The "sort_by", "offset" and "limit" parameters are optional.

A user is considered to be active / logged in if the last access time is within the range of the activity timeout. The timeout can be set using the "ActivityTimeout" directive in a configuration override for "site.ini". The default timeout is one hour.

The "sort_by" parameter must be specified as an array. Each element of the array must be another array where the first element denotes the field (as a string) that the sorting mechanism should use. The second element specifies the direction of the sort (as a boolean). The following sorting fields can be used:

- user_id

- login

- activity

- email

Examples

Example 1

```
{def $users=fetch( 'user', 'logged_in_users',
                   hash( 'sort_by', array( array( 'login', true() ) ) ) )}

{foreach $users as $user}
    {$user.contentobject.name} <br />
{/foreach}
```

Outputs the names of the users that are currently logged in (sorted by usernames).

member_of

Summary

Fetches the roles that are assigned to a user.

Usage

```
fetch( 'user', 'member_of', hash( 'id', id ) )
```

Parameters

Name	Type	Description	Required
id	integer	The ID number of the target user.	Yes.

Returns

An array with ezrole (page 1109) objects or FALSE.

Description

This function will fetch the roles that are assigned to a user. The desired user's ID number must be specified using the "id" parameter. The function will return an array of ezrole (page 1109) objects. If no roles are associated with the user, or if an invalid user ID is provided, the function will return FALSE.

Examples

Example 1

```
{def $roles=fetch( 'user', 'member_of', hash( 'id', 42 ) )}

{foreach $roles as $role}
    {$role.name} <br />
{/foreach}
```

Outputs the names of the roles that are assigned to user number 42.

5

user_role

Summary

Fetches the policies that are available for a user.

Usage

```
fetch( 'user', 'user_role', hash( 'user_id', user_id ) )
```

Parameters

Name	Type	Description	Required
user_id	integer	The user to fetch policies from	Yes.

Returns

An array of hashes or FALSE.

Description

This function will fetch the policies that are available for a user. The desired user's ID number must be specified using the "id" parameter. The function will return an array of policy structures or FALSE if no policies are available or if a non-existing user ID number is provided. The following table shows the structure of the hashes that make up the elements of the returned array.

Name	Type	Description
moduleName	string	The name of the module that the user has access to (* means all modules).
functionName	string	The name of the function that the user has access to (* means all functions).
limitation	string	The elements of the module and function that the user has access to (* means no limitations).

Examples

Example 1

```
{def $policies=fetch( 'user', 'user_role', hash( 'user_id', 42 ) )}

{foreach $policies as $policy}
   {$policy.moduleName} /
   {$policy.functionName} /
   {$policy.limitation} <br />
{/foreach}
```

Outputs information about the policies that are available for user number 42.

5

Views

activate   (page 990)
 Provides an interface for activating a user account.

forgotpassword   (page 991)
 Provides an interface for situations where a user forgets his/her password.

login   (page 992)
 Provides an interface for logging in a user.

logout   (page 993)
 Provides a mechanism that logs out a user.

password   (page 994)
 Provides an interface for changing the password for the current user.

preferences   (page 995)
 Provides an interface for managing the preferences of the current user.

register   (page 996)
 Provides an interface for registering a new user.

setting   (page 997)
 Provides an interface for tweaking user account settings.

success   (page 998)
 Provides an interface that is called upon a successful user registration.

5

activate

Summary

Provides an interface for activating a user account.

5

forgotpassword

Summary

Provides an interface for situations where a user forgets his/her password.

login

Summary

Provides an interface for logging in a user.

logout

Summary

Provides a mechanism that logs out a user.

password

Summary

Provides an interface for changing the password for the current user.

5

preferences

Summary

Provides an interface for managing the preferences of the current user.

register

Summary

Provides an interface for registering a new user.

setting

Summary

Provides an interface for tweaking user account settings.

success

Summary

Provides an interface that is called upon a successful user registration.

5

5.4.22   workflow

Summary

Provides views for managing workflows, workflow groups, workflow events, etc.

Description

This module provides an interface to the workflow engine inside the eZ Publish kernel. A workflow is a sequential list of events that is started by a trigger. This module contains views that make it possible to manipulate workflow groups, workflows and events. The administration interface makes use of the views that this module provides in order to allow the users to manage workflows (add new, remove, edit, etc.). Please refer to the "Workflows" (page 167) section of the "Concepts an basics" chapter for more information about workflows.

The module components are documented in the following sections:

- Views (page 1000)

Views

down   (page 1001)
     Provides an interface for moving an event to a lower position.

edit   (page 1002)
     Provides an interface for editing a workflow.

event   (page 1003)
     Not documented yet.

groupedit   (page 1004)
     Provides an interface for editing a workflow group.

grouplist   (page 1005)
     Provides an interface for generating a list of all available workflow groups.

process   (page 1006)
     Not documented yet.

run   (page 1007)
     Not documented yet.

up   (page 1008)
     Provides an interface for moving an event to a higher position.

view   (page 1009)
     Provides an interface for viewing a workflow.

workflowlist   (page 1010)
     Provides an interface for generating a list of workflows that belong to a group.

5

down

Summary

Provides an interface for moving an event to a lower position.

edit

Summary

Provides an interface for editing a workflow.

event

Summary

Not documented yet.

groupedit

Summary

Provides an interface for editing a workflow group.

grouplist

Summary

Provides an interface for generating a list of all available workflow groups.

process

Summary

Not documented yet.

5

run

Summary

Not documented yet.

up

Summary

Provides an interface for moving an event to a higher position.

5

view

Summary

Provides an interface for viewing a workflow.

5

workflowlist

Summary

Provides an interface for generating a list of workflows that belong to a group.

5

5.5   Views

The views are documented in the following sections:

- class (page 647)
- collaboration (page 666)
- content (page 679)
- error (page 822)
- ezinfo (page 823)
- form (page 828)
- infocollector (page 831)
- layout (page 836)
- notification (page 841)
- package (page 853)
- pdf (page 876)
- reference (page 880)
- role (page 881)
- rss (page 889)
- search (page 895)
- section (page 902)
- setup (page 918)
- shop (page 919)
- trigger (page 962)
- url (page 965)
- user (page 974)
- workflow (page 999)

5

5.6 Objects

\begin{description} \item[ezauthor]\refer{labelc45a3b282ac09950cc6f3bc1679361d3}\ \\ Contains information about authors.

\item[ezbasket]\refer{label61bb94347ab6dd5f5c7744e80a6718ec}\ \\ Contains information about a user's shopping basket.

\item[ezbinaryfile]\refer{label6f82366174cfab4de4cd8ca7620d37bf}\ \\ Contains information about a file.

\item[ezcontentbrowsebookmark]\refer{label9bfb0bcb9c0aff160879c8ad0222bb43}\ \\ Contains information about a bookmark.

\item[ezcontentbrowserecent]\refer{labelb3f474da0ca57fd09e8aee264c69b298}\ \\ Contains information about a node with recently edited children.

\item[ezcontentclass]\refer{label832173e036ed6a272ed64845c155fb75}\ \\ Contains information about a content class.

\item[ezcontentclassattribute]\refer{labele1c95c5f154f2c185ef9b1e97a88d7a1}\ \\ Contains information about an attribute of a content class.

\item[ezcontentclassclassgroup]\refer{label81914e96495671bd6674f72fb9981bf8}\ \\ Contains information about a class group assignment.

\item[ezcontentclassgroup]\refer{labelb754c7fc3c601fbf7969e8c7b7eb3a2b}\ \\ Contains information about a class group.

\item[ezcontentlanguage]\refer{labelbddb0ca5208c07b5974ac641d4f3b172}\ \\ Contains information about a language.

\item[ezcontentobject]\refer{label30d021727dc5815d4f84c748b21ead1b}\ \\ Contains information about a content object.

\item[ezcontentobjectattribute]\refer{label9eb92ac927b091ff74fd873bbc3bf82d}\ \\ Contains information about an attribute of a content object.

\item[ezcontentobjecttranslation]\refer{labelf2547dc9c47073e9769a6a08b323adc2}\ \\ DEPRECATED (Contains information about a translation.)

\item[ezcontentobjecttrashnode]\refer{labeldb83f33cb9ad386d633052a1e10fc8aa}\ \\ Contains information about a node moved to the trash.

\item[ezcontentobjecttreenode]\refer{label08b6bc1d4ca38502457c180194b3f092}\ \\ Contains information about a node within the content node tree.

\item[ezcontentobjectversion]\refer{labelf52f1868d9eb1bf9c1f4029f4473bbaf}\ \\ Contains information about a version of a content object.

\item[ezcurrencydata]\refer{label50e48e54d845e00b09cf57193142e59c}\ \\ Contains information about a currency.

\item[ezdate]\refer{label88deb7b831d78fdfad32e26644eb8718}\ \\ Contains information about a date.

5

\item[ezdatetime]\refer{label415d98e0eb750b6429a0ea8e5dc6dbb7}\ \\ Contains information about a date and time.

\item[ezimagealiashandler]\refer{label5e08d7fdec0635eec21c63389cf07e3c}\ \\ Contains information about an image.

\item[ezimagelayer]\refer{label172cc76fc1bf9ef17788f50e074495dc}\ \\ Contains information about an image layer.

\item[ezimageobject]\refer{label7a47602af204cec3419210ffa593a16d}\ \\ Contains information about an image.

\item[ezinformationcollection]\refer{label1a0af96d645b1a6383e1029f0d482dc3}\ \\ Contains information about a block of collected information.

\item[ezinformationcollectionattribute]\refer{label46149d0d9ecedcf69172dceda8bda9df}\ \\ Contains information about an attribute of a collection.

\item[ezkeyword]\refer{label043b0464a57ab0ef97ad12e73ad50026}\ \\ Contains information about keywords.

\item[ezlocale]\refer{label7ad00aa8e698f725d52e525ee156dafe}\ \\ Contains information about a locale.

\item[ezmatrix]\refer{labelec4893b2d828f3df11e0770b32b791de}\ \\ Contains information about a matrix.

\item[ezmedia]\refer{label4325f8911195eb043d3cdd49a1a5de4f}\ \\ Contains information about a video file.

\item[ezmultioption]\refer{label9d4f31733d1dc13a26eec044f942dc5c}\ \\ DEPRECATED (Contains information about multilpe options.)

\item[ezmultioption2]\refer{label2b47e2360e171c7f89e7949062a74fe5}\ \\ Contains information about a group of multi-options.

\item[ezmultiprice]\refer{labelc85a7c591d281763dad18c81de034a44}\ \\ Contains information about prices and currencies for a product.

\item[eznodeassignment]\refer{label0f450df271d69ea46295a8b9cd6e2cf6}\ \\ Contains information about a node assignment.

\item[ezoption]\refer{label49d22be4c1fc38667be8d61b77b1fea6}\ \\ Contains information about a collection of options.

\item[ezorder]\refer{label496ebcf1110645fd919b746f30bb41f1}\ \\ Contains information about an order.

\item[ezorderitem]\refer{label5b894f0c5cf2c56c6e592835dc0e2db9}\ \\ Contains information about order items.

\item[ezorderstatus]\refer{label59ede7121f798a4473196ed13bc7c627}\ \\ Contains information about an order status.

\item[ezpolicy]\refer{labelf304b08a1f98f3898b823c73c2df5b51}\ \\ Contains information about a policy.

\item[ezprice]\refer{labelddfc6748a170ce2f18237c030843d0fb}\   \\   Contains   information about a price.

\item[ezproductcategory]\refer{label49074d74cb80dccf2a8695a6dfa788e8}\   \\   Contains information about a product category.

\item[ezproductcollectionitem]\refer{label0c84f7f19e2adb99c03c38aab599fe1f}\   \\   Contains information about an item of a product collection.

\item[ezrangeoption]\refer{label3709e4889c008f9dfd52a1b040b07de9}\ \\ Contains information about a range of options.

\item[ezrole]\refer{label97cd0a21c4bc9772cfa7f4b6f9f822ca}\ \\ Contains information about a role.

\item[ezsection]\refer{labele0786bda75797a682450a3eea2305651}\ \\ Contains information about a section.

\item[ezsimplifiedxmlinput]\refer{labele0399c249d4cc14867e2b2946f54fc1c}\ \\ Contains information about XML data.

\item[ezsubtreenotificationrule]\refer{label5eb67180980e8413002e58458c1c2b38}\ \\ Contains information about a subtree notification rule.

\item[eztime]\refer{labelbf95ce9abfa8eee96e75eef22913e766}\   \\   Contains   information about a time value.

\item[ezurl]\refer{labeldb51b1323f429737b20abcca7e4bfad0}\ \\ Contains information about a URL.

\item[ezuser]\refer{label37074849b704944c869c20a2e954b503}\   \\   Contains   information about a user.

\item[ezvatrule]\refer{label7ccfa7f2cca9b8cea0dd136a8976ab13}\   \\   Contains   information about a VAT charging rule.

\item[ezvattype]\refer{label80354b55017749de2485c5ec22ddeddf}\ \\ Contains information about a VAT.

\item[ezxhtmlxmloutput]\refer{label6c28b63f94c9e841d6f7ef65709dd755}\   \\   Contains information about XML data.

\item[ezxmlinputhandler]\refer{label13f0d800dbb767d6d25fd3107b6e0b91}\ \\ Contains information about XML data.

\item[ezxmloutputhandler]\refer{label18d1e66e07c9869394c641327dc6dbe1}\   \\   Contains information about XML data.

\item[ezxmltext]\refer{labelc805f12b6b66d3d2f1e46dc4a0fd3621}\ \\ Contains information about an XML block.

\end{description}

5.6.1   ezauthor

Summary

Contains information about authors.

Attributes

Attribute	Type	Static*	Description
author_list	array	Yes.	Contains information about the authors. Each element in the array consists of a hash of strings. The keys are "id", "name" and "email".
name	NULL	Yes.	Not used.
is_empty	boolean	No.	Returns TRUE if the object does not contain any authors (if the "author_list" array is empty); otherwise FALSE is returned.

* Indicates whether the attribute provides a pre-fetched value or if it needs to be computed upon request.

5

5.6.2 ezbasket

Summary

Contains information about a user's shopping basket.

Attributes

Attribute	Type	Static*	Description
id	string	Yes.	The ID number of the shopping basket.
session_id	string	Yes.	The ID of the session that the basket belongs to.
productcollection_id	string	Yes.	The ID number of the product collection that belongs to the basket.
order_id	string	Yes.	The ID number of the order that belongs to the basket.
items	array	No.	An array of hashes containing information about the items. Each element consists of the following data: **Attribute:** id **Type:** string **Description:** The ID number of the item in the basket. **Attribute:** vat_value **Type:** string **Description:** The actual value of VAT (for example, 22). **Attribute:** item_count **Type:** string **Description:** The quantity.

5

| | | | **Attribute:** node_id **Type:** string **Description:** The ID number of the node that represents the item.

Attribute: object_name **Type:** string **Description:** The name of the object that represents the item.

Attribute: price_ex_vat **Type:** double **Description:** The price of the item excluding the VAT.

Attribute: price_inc_vat **Type:** string **Description:** The price of the item including the VAT.

Attribute: discount_percent **Type:** string **Description:** The discount percentage (if any).

Attribute: total_price_ex_vat **Type:** double **Description:** The total price excluding the VAT.

Attribute: total_price_inc_vat **Type:** double |

			Description: The total price including the VAT. **Attribute:** item_object **Type:** object **Description:** The item itself (as an ezproduct-collectionitem (page 1106) object).
total_ex_vat	float	No.	The total amount to be payed excluding the VAT.
total_inc_vat	float	No.	The total amount to be payed including the VAT.
is_empty	boolean	No.	Returns TRUE if there are no items in the basket, FALSE otherwise.
productcollection	object	No.	The product collection that belongs to the basket.

* Indicates whether the attribute provides a pre-fetched value or if it needs to be computed upon request.

5

5.6.3   ezbinaryfile

Summary

Contains information about a file.

Attributes

Attribute	Type	Static*	Description
contentobject_attribute_id	string	Yes.	The ID number of the content object attribute that the file belongs to.
version	string	Yes.	The version number of the object that the file belongs to.
filename	string	Yes.	The internal name of the file (generated by the system).
original_filename	string	Yes.	The original name of the file.
mime_type	string	Yes.	The MIME type of the file (for example "audio/wav").
download_count	string	Yes.	The number of times the file has been downloaded through the "download" (page 794) view of the "content" module.
filesize	integer	No.	The size of the file (number of bytes).
filepath	string	No.	The path to the file (including the filename).
mime_type_category	string	No.	The MIME type category (for example "audio").
mime_type_part	string	No.	The MIME type part (for example "wav").

* Indicates whether the attribute provides a pre-fetched value or if it needs to be computed upon request.

5.6.4   ezcontentbrowsebookmark

Summary

Contains information about a bookmark.

Attributes

Attribute	Type	Static*	Description
id	string	Yes.	The ID number of the bookmark.
user_id	string	Yes.	The ID number of the user that the bookmark belongs to.
node_id	string	Yes.	The ID number of the bookmarked node.
name	string	Yes.	The name of the bookmark (the same as the name of the node).
node	object	No.	The bookmarked node (as ezcontentobjecttreenode (page 1051) object).
contentobject_id	string	No.	The ID number of the object that is referenced by the bookmarked node.

* Indicates whether the attribute provides a pre-fetched value or if it needs to be computed upon request.

5

5.6.5   ezcontentbrowserecent

Summary

Contains information about a node with recently edited children.

Attributes

Attribute	Type	Static*	Description
id	string	Yes.	A unique ID number.
user_id	string	Yes.	The ID number of the user that the "browse recent" entry belongs to.
node_id	string	Yes.	The ID number of the node under which something was recently published.
created	string	Yes.	A UNIX timestamp pinpointing the exact date/time when the "browse recent" entry was created.
name	string	Yes.	The name of the node under which something was recently published.
node	object	No.	The actual node under which something was recently published (as an ezcontentobjecttreenode (page 1051) object).
contentobject_id	string	No.	The ID of the object which is encapsulated by the node under which something was recently published.

* Indicates whether the attribute provides a pre-fetched value or if it needs to be computed upon request.

5.6.6   ezcontentclass

Summary

Contains information about a content class.

Attributes

Attribute	Type	Static*	Description
id	string	Yes.	The ID number of the class.
version	string	Yes.	The version/status of the class (0=normal, 1=temporary, 2=modified).
name	string	Yes.	The name of the class (for example "News article").
identifier	string	Yes.	The identifier of the class (for example "news_article").
contentobject_name	string	Yes.	The pattern which controls how the names of the instances should be generated.
creator_id	string	Yes.	The ID number of the object that represents the user who created the class.
modifier_id	string	Yes.	The ID number of the object that represents the user who most modified the class last.
created	string	Yes.	A UNIX timestamp pinpointing the exact date/time when the class was created.
remote_id	string	Yes.	A global unique ID for the class. The remote ID is an MD5 hash of the time when the class was

			created plus a random value. Remote IDs are used to avoid collision of identical classes during an import.
modified	string	Yes.	A UNIX timestamp pinpointing the exact date/time when the class was last modified.
is_container	string	Yes.	Either 1 or 0. Reveals whether nodes referencing objects of this class should be considered as containers or not. Used by the administration interface to allow or disallow the creation of nodes under a node which references an object of this class.
always_available	string	Yes.	Either 1 or 0. Controls whether new instances (objects) of this class should be always available or not. Used on multi-language sites to make it possible to fetch objects which do not exist in any of the site languages. Note that this value simply dictates the default value for the "always_available" flag of the objects themselves and thus it only affects new instances.
data_map	array	No.	The attributes (as ezcontentclassat-

			tribute (page 1026) objects) that make up the class.
object_count	string	No.	The number of instances (objects) of the class.
version_count	string	No.	DEPRECATED - Similar to the "version" attribute, but will be 2 if the "version_count" is 2 or higher.
version_status	string	No.	DEPRECATED - The version count of the class if it has been determined, FALSE if not determined.
ingroup_list	array	No.	The class groups (as ezclassclassgroup (page 1029) objects) that the class is a member of.
ingroup_id_list	array	No.	The class groups (as ezclassgroup (page 1029) objects) that the class belongs to.
match_ingroup_id_list	array	No.	The class groups (as ezclassgroup (page 1029) objects) that the class belongs to. This variable is connected with a feature that makes it possible to create template overrides based on class groups. By default the "match_ingroup_id_list" always returns a boolean FALSE value because the class group override feature is turned off. It can be turned on by setting the "Enable-

5

			Class-GroupOverride" directive in the [ContentOverrideSettings] block of a configuration override for "content.ini" to "true".
group_list	array	No.	All the class groups (as ezclassclassgroup (page 1029) objects) that are present in the database.
creator	object	No.	The object (as ezcontentobject (page 1032) object) representing the user who created the class.
modifier	object	No.	The object (as ezcontentobject (page 1032) object) representing the user who last modified the class.
can_instantiate_languages	array	No.	Not documented yet.

* Indicates whether the attribute provides a pre-fetched value or if it needs to be computed upon request.

5

5.6.7 ezcontentclassattribute

Summary

Contains information about an attribute of a content class.

Attributes

Attribute	Type	Static*	Description
id	string	Yes.	The ID number of the class attribute.
name	string	Yes.	The name of the class attribute.
version	string	Yes.	The version number of the version that the attribute belongs to.
contentclass_id	string	Yes.	The ID number of the class that the attribute belongs to.
identifier	string	Yes.	The identifier of the class attribute (for example "first_name").
placement	string	Yes.	The location of the class attribute within the list of attributes.
is_searchable	string	Yes.	Either 1 or 0. 1 means that the content stored using this attribute will be indexed by the search engine, 0 means that the content will not be indexed.
is_required	string	Yes.	Either 1 or 0. 1 means that input is required, 0 means that empty inputs are allowed.
can_translate	string	Yes.	Either 1 or 0. 1 means that instances of the attribute can be translated to dif-

			ferent languages. 0 means that no translations (except for the default translation) can be made.
is_information_collector	string	Yes.	Either 1 or 0. 1 means that the attribute functions as an information collector. 0 means that the attribute is just a normal attribute and thus stores data in the default/normal way.
data_type_string	string	Yes.	The identifier string of the datatype that is used to represent the class attribute (for example "ezstring").
data_int1	string	Yes.	Integer 1.
data_int2	string	Yes.	Integer 2.
data_int3	string	Yes.	Integer 3.
data_int4	string	Yes.	Integer 4.
data_float1	string	Yes.	Float 1.
data_float2	string	Yes.	Float 2.
data_float3	string	Yes.	Float 3.
data_float4	string	Yes.	Float 4.
data_text1	string	Yes.	Text 1.
data_text2	string	Yes.	Text 2.
data_text3	string	Yes.	Text 3.
data_text4	string	Yes.	Text 4.
content	mixed	No.	Data for the datatype which this class attribute is made of, the actual data depends on the datatype.
temporary_object_attribute	object	No.	A temporary content object attribute (as ezcontentobjectattribute (page 1042) object) which does not exist in the

			database.
data_type	object	No.	The datatype that is used to represent the class attribute.
display_info	array	No.	Array of miscellaneous display parameters used by the system (for example whether the components of the edit interface should be grouped or not).

* Indicates whether the attribute provides a pre-fetched value or if it needs to be computed upon request.

5

5.6.8 ezcontentclassclassgroup

Summary

Contains information about a class group assignment.

Attributes

Attribute	Type	Static*	Description
contentclass_id	string	Yes.	The ID number of the class which belongs to the group.
contentclass_version	string	Yes.	The version (either 1 or 0).
group_id	string	Yes.	The ID number of the class.
group_name	string	Yes.	The name of the class group (for example "Media").

* Indicates whether the attribute provides a pre-fetched value or if it needs to be computed upon request.

5

5.6.9   ezcontentclassgroup

Summary

Contains information about a class group.

Attributes

Attribute	Type	Static*	Description
id	string	Yes.	The ID number of the class group.
name	string	Yes.	The name of the class group (for example "Media").
creator_id	string	Yes.	The ID number of the object representing the user who created the class group.
modifier_id	string	Yes.	The ID number of the object representing the user who last modified the class group.
created	string	Yes.	A UNIX timestamp pinpointing the exact date/time when the class group was created.
modified	string	Yes.	A UNIX timestamp pinpointing the exact date/time when the class group was last modified.
creator	object	No.	The user who created the class group (as ezuser (page 1117) object).
modifier	object	No.	The user who last modified the class group (as ezuser (page 1117) object).

* Indicates whether the attribute provides a pre-fetched value or if it needs to be computed upon request.

5.6.10 ezcontentlanguage

Summary

Contains information about a language.

Attributes

Attribute	Type	Static*	Description
id	string	Yes.	The ID number of the language (can be 2, 4, 8, 16, 32, etc.)
name	string	Yes.	The name of the language, for example "English (United Kingdom)".
locale	string	Yes.	The language's locale code (for example "eng-GB").
disabled	string	Yes.	Reserved for future use.
translation	object	No.	Returns reference to the ezcontentlanguage object itself.
locale_object	object	No.	The locale of this language (as ezlocale (page 1076) object).
object_count	string	No.	The number of objects translated to this language.

* Indicates whether the attribute provides a pre-fetched value or if it needs to be computed upon request.

5

5.6.11 ezcontentobject

Summary

Contains information about a content object.

Attributes

Attribute	Type	Static*	Description
id	string	Yes.	The ID number of the object.
section_id	string	Yes.	The ID number of the section that the object belongs to.
owner_id	string	Yes.	The ID number of the object representing the user who initially created the object.
contentclass_id	string	Yes.	The ID number of the content class which the object is an instance of.
is_published	string	Yes.	Either 1 or 0. 1 means that the object has been published. 0 means that the object has not yet been published.
published	string	Yes.	A UNIX timestamp pinpointing the exact date/time when the object was published for the first time.
modified	string	Yes.	A UNIX timestamp pinpointing the exact date/time when the object was last modified.
current_version	string	Yes.	The number of the currently published version.
status	string	Yes.	The status of the ob-

			ject (0=Draft, 1=Published, 2=Archived).
remote_id	string	Yes.	A global unique ID for the object. The remote ID is an MD5 hash of the time when the object was created plus a random value. Remote IDs are used in order to avoid collisions of identical objects during an import.
language_mask	string	Yes.	The sum of the ID numbers of the languages in which the last published version of the object exists. Please note that the ID numbers are powers of 2. 1 (2^0) is reserved for marking objects that should always be available. Objects that are always available will be fetched even if they do not exist in any of the site/ prioritized languages (the initial/main language of the object will be used).
initial_language_id	string	Yes.	The ID number of the initial/main language.
current	object	No.	The current version (as ezcontentobjectversion (page 1056)) of the object.
versions	array	No.	The object's versions

			(as ezcontentob-jectversion (page 1056) objects).
author_array	array	No.	Array of ezuser (page 1117) objects representing the different creators of the content object's versions.
class_name	string	No.	The name of the class which the content object is an instance of (for example "Consumer product").
content_class	object	No.	The content class (as ezcontentclass (page 1022)) which the content object is an instance of.
contentobject_attributes	array	No.	Array of ezcontento-bjectat-tribute (page 1042) objects representing the attributes of the content object.
owner	object	No.	An ezcontentobject (page 1032) that represents the user who initially created the object.
related_contentobject_array	array	No.	An array of ezcontento-bject (page 1032) objects that are related to this object. This attribute is deprecated. It is recommended to use the "related_objects (page 747)" fetch function instead.
related_contentobject_count	string	No.	The number of objects that are related to this object. This attribute is depre-

			cated. It is recommended to use the "related_objects_count (page 750)" fetch function instead.
reverse_related_contentobject_array	array	No.	An array of ezcontentobject (page 1032) objects that make use of this object (reverse relations). This attribute is deprecated. It is recommended to use the "reverse_related_objects (page 751)" fetch function instead.
reverse_related_contentobject_count	string	No.	The number of objects that are using this object. This attribute is deprecated. It is recommended to use the "reverse_related_objects_count (page 755)" fetch function instead.
linked_contentobject_array	array	No.	An array of ezcontentobject (page 1032) objects that are related to this object by means of the "XML linked" object relation.
linked_contentobject_count	string	No.	The number of objects that are related to this object by means of the "XML linked" object relation.
reverse_linked_contentobject_array	array	No.	An array of ezcontentobject (page 1032) objects that make use of this object by means of

			the "XML linked" object relation.
reverse_linked_ contentobject_count	string	No.	The number of objects that are using this object by means of the "XML linked" object relation.
embedded_ contentobject_array	array	No.	An array of ezcontentobject (page 1032) objects that are related to this object by means of the "XML embedded" object relation.
embedded_ contentobject_count	string	No.	The number of objects that are related to this object by means of the "XML embedded" object relation.
reverse_embedded_ contentobject_array	array	No.	An array of ezcontentobject (page 1032) objects that make use of this object by means of the "XML embedded" object relation.
reverse_embedded_ contentobject_count	string	No.	The number of objects that are using this object by means of the "XML embedded" object relation.
can_read	boolean	No.	Returns TRUE if the current user has read access to the object, FALSE otherwise.
can_pdf	boolean	No.	Not documented yet.
can_diff	boolean	No.	Not documented yet.
can_create	boolean	No.	Not documented yet.
can_create_class_list	array	No.	Not documented yet.
can_edit	boolean	No.	Returns TRUE if the

			current user has edit access to the object, FALSE otherwise.
can_translate	boolean	No.	Returns TRUE if the current user has permissions to translate the contents of the object, FALSE otherwise.
can_remove	boolean	No.	Returns TRUE if the current user has permissions to remove the object, FALSE otherwise.
can_move	boolean	No.	Not documented yet.
can_move_from	boolean	No.	Returns TRUE if the current user has permissions to move the main node of the object, FALSE otherwise.
can_view_embed	boolean	No.	Not documented yet.
data_map	array	No.	Array of ezcontentobjectattribute (page 1042) objects representing the actual attributes of the content object.
main_parent_node_id	string	No.	The ID number of the main node of the object encapsulated by the parent node.
assigned_nodes	array	No.	Array of nodes (as ezcontentobjecttreenode (page 1051) objects) that encapsulate the object.
parent_nodes	array	No.	An array of ID numbers of the parent nodes (as strings).
main_node_id	string	No.	The ID number of the object's main

5

			node.
main_node	object	No.	The object's main node (as ezcontentobjecttreenode (page 1051)).
default_language	string	No.	The default language for content objects (see ContentObjectLocale (page 2431)) if it is set or the most prioritized language (i.e. the first element of the "SiteLanguageList (page 2423)" array).
content_action_list	array	No.	An array of hashes revealing information about the content actions that can be performed on the object. The keys "name" and "action" con- tain the actual name (for example "Add to basket" - which should be value of the HTML input tag) and the action itself (for example "ActionAddToBasket" - which should be the name of the HTML input tag). The ar- ray is generated by a function that exam- ines the object's at- tributes. If a datatype used to represent an at- tribute provides sup- port for content ac- tions or if the at- tribute is an infor-

			mation collector, the supported actions will be added to the "content_action_list" array. This array can be used to automatically generate action buttons (standard HTML buttons) for content objects that either make use of special datatypes or have attributes that are marked as information collectors.
class_identifier	string	No.	The identifier of the class which the object is an instance of (for example "consumer_product").
class_group_id_list	array	No.	An array of ID numbers of the class groups which the class (that the object is an instance of) belongs to.
name	string	No.	The actual name of the object in the current language (for example "Liver sandwich").
match_ingroup_id_list	array	No.	The ID numbers of the class groups that the class (which the object 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

			"match_ingroup_id_list" always returns a boolean FALSE value because the class group override feature is turned off. It can be turned on by setting the "Enable-Class-GroupOverride" directive in the [ContentOverrideSettings] block of a configuration override for "content.ini" to "true".
current_language	string	No.	Not documented yet.
current_language_object	object	No.	Not documented yet.
initial_language	object	No.	The initial/main language as a ezcontentlanguage (page 1031) object.
initial_language_code	string	No.	The code of the initial/main language (for example "eng-GB").
available_languages	array	No.	The languages in which the last published version of the object exists (as an array of locale codes).
language_codes	array	No.	Not documented yet.
language_js_array	string	No.	Not documented yet.
languages	array	No.	The list of languages the object exists in, ordered by their priority (as an array of ezcontentlanguage (page 1031) objects).
can_edit_languages	array	No.	Not documented

			yet.
can_create_ languages	array	No.	Not documented yet.
always_available	boolean	No.	Returns TRUE if the object is marked as "always available" (i.e. the object will be fetched even if it does not exist in any of the site languages), FALSE otherwise.
state_id_array	array	No.	Array containing the states ID of the content object.
state_identifier_array	array	No.	Array containing the states identifiers of the content object.

* Indicates whether the attribute provides a pre-fetched value or if it needs to be computed upon request.

5

5.6.12   ezcontentobjectattribute

Summary

Contains information about an attribute of a content object.

Attributes

Attribute	Type	Static*	Description
id	string	Yes.	The ID number of the attribute.
contentobject_id	string	Yes.	The ID number of the content object that the attribute belongs to.
version	string	Yes.	The version number of the content object that the attribute belongs to.
language_code	string	Yes.	The code of the translation that the attribute belongs to (for example "eng-GB").
language_id	string	Yes.	The ID number of the translation language plus 1 if this is the language is initial/main and the object is always available.
contentclassattribute_id	string	Yes.	The ID number of the attribute.
attribute_original_id	string	Yes.	The original ID of the attribute.
sort_key_int	string	Yes.	Integer used for sorting.
sort_key_string	string	Yes.	Text used for sorting.
data_type_string	string	Yes.	The identifier string of the datatype (for example "ezstring").
data_text	string	Yes.	Text stored by the attribute.
data_int	string	Yes.	Integer stored by the

5

			attribute.
data_float	string	Yes.	Float stored by the attribute.
contentclass_attribute	object	No.	The class attribute (as an ezcontentclassattribute (page 1026) object).
contentclass_attribute_identifier	string	No.	The identifier of the content class attribute (for example "first_name").
contentclass_attribute_name	string	No.	The name of the content class attribute.
can_translate	string	No.	1 if the attribute is translatable, 0 if not.
is_information_collector	string	No.	1 if the attribute is an information collector, 0 if not.
is_required	string	No.	1 if the attribute is required, 0 if not.
content	any	No.	The actual content (what is returned when the ".content" notation is used).
has_http_value	boolean	No.	TRUE if the attribute has an HTTP value, FALSE otherwise.
value	any	No.	The HTTP input from the user (if submitted) or the contents of the object attribute from the database (same as the "content" attribute).
has_content	boolean	No.	TRUE if the attribute contains content, FALSE if it is empty.
class_content	any	No.	The content of the class attribute which this object attribute is made from (same as ".contentclass_

5

			attribute.content").
object	object	No.	The object that the attribute belongs to (as an ezcontentobject (page 1032) object).
object_version	object	No.	The version that the attribute belongs to (as an ezcontentobjectversion (page 1056) object).
view_template	string	No.	The name of the template that is used to display the view interface for the attribute (for example "ezstring").
edit_template	string	No.	The name of the template that is used to display the edit interface for the attribute (for example "ezstring").
result_template	string	No.	The name of the template that is used to display the information that was collected by the attribute (for example "ezstring").
has_validation_error	boolean	No.	TRUE if a validation error was detected, FALSE if everything is okay.
validation_error	NULL	No.	The validation error(s), NULL if none.
validation_log	NULL	No.	A log of the validation error(s), NULL if none.
language	object	No.	The original translation (as ezcontentobjectattribute (page 1042) object) of this content object attribute (it may just

5

			be the same object).
is_a	string	No.	Returns the identifier of the datatype that is used to represent the attribute (for example "ezstring").
display_info	array	No.	An array containing information about how the attribute should be displayed in different scenarios (for example if the information should be grouped, etc.).
class_display_info	array	No.	An array containing information about how the attribute should be displayed on the class level.

* Indicates whether the attribute provides a pre-fetched value or if it needs to be computed upon request.

5

5.6.13 ezcontentobjecttranslation

Summary

DEPRECATED (Contains information about a translation.)

Attributes

Attribute	Type	Static*	Description
contentobject_id	string	Yes.	The ID number of the object that the translation belongs to.
version	string	Yes.	The version number that the translation belongs to.
language_code	string	Yes.	The translation's language code (for example "eng-GB").
locale	object	No.	The locale (as ezlocale (page 1076) object) that the translation uses.

* Indicates whether the attribute provides a pre-fetched value or if it needs to be computed upon request.

5.6.14   ezcontentobjecttrashnode

Summary

Contains information about a node moved to the trash.

Attributes

Attribute	Type	Static*	Description
node_id	string	Yes.	The ID number of the node.
parent_node_id	string	Yes.	The ID number of the parent node.
main_node_id	string	Yes.	The ID number of the main node.
contentobject_id	string	Yes.	The ID number of the content object.
contentobject_version	string	Yes.	The number of the published version.
depth	string	Yes.	The depth of the node within the content node tree. The depth of a top level node is 1.
sort_field	string	Yes.	The sorting method used to sort the child nodes.
sort_order	string	Yes.	The sorting order used when sorting the node's children.
priority	string	Yes.	The node's priority (positive or negative integer).
modified_subnode	string	Yes.	A UNIX timestamp pinpointing the exact time a sub node was changed.
path_string	string	Yes.	The node's path string (for example "/1/2/44").
path_identification_string	string	Yes.	The node's path identification string (for example "company/about_us/my_

5

			article").
remote_id	string	Yes.	A unique ID for the node (avoids crashes when importing/ exporting nodes). A remote ID is an MD5 hash of the time when the node was generated plus a random value.
is_hidden	string	Yes.	The node's hidden status (0=visible, 1=hidden).
is_invisible	string	Yes.	The node's visibility status (0=visible, 1=hidden by superior).
name	string	No.	The name of the object the node encapsulates (for example "My article").
data_map	array	No.	The object's attributes as ezcontentobjectattribute (page 1042) objects.
object	object	No.	The actual content object (as ezcontentobject (page 1032)) that the node encapsulates.
contentobject_version_object	object	No.	The current version (as ezcontentobjectversion (page 1056)) of the object that the node encapsulates.
sort_array	array	No.	The node's sort array.
can_read	boolean	No.	Returns TRUE if the current user has read access to the node (FALSE otherwise).
can_create	boolean	No.	Returns TRUE if the current user can cre-

			ate nodes under this node (FALSE otherwise).
can_edit	boolean	No.	Returns TRUE if the current user has edit access to the node (FALSE otherwise).
can_remove	boolean	No.	Returns TRUE if the current user can remove the node (FALSE otherwise).
creator	object	No.	The object (as ezcontentobject (page 1032)) containing the user who created the node.
path_array	array	No.	Array of strings revealing the ID numbers of the nodes that make up the path. The current node is also included.
parent	object	No.	The parent node (as ezcontentobjecttreenode (page 1051)).
class_identifier	string	No.	The identifier of the class which the object encapsulated by the node is an instance of (for example "product_review").
class_name	string	No.	The name of the class which the object encapsulated by the node is an instance of (for example "Product review").
original_parent	object	No.	Returns the parent node (as ezcontentobjecttreenode (page 1051)) or NULL if the parent

			node has been re-moved or moved to another place in the content tree.
original_parent_path_id_string	string	No.	Returns the path identification string of the parent node (for example "company/about_us") regardless whether the par-ent node has been removed, moved to another place in the content tree or not.

* Indicates whether the attribute provides a pre-fetched value or if it needs to be computed upon request.

5

5.6.15 ezcontentobjecttreenode

Summary

Contains information about a node within the content node tree.

Attributes

Attribute	Type	Static*	Description
node_id	string	Yes.	The ID number of the node.
parent_node_id	string	Yes.	The ID number of the parent node.
main_node_id	string	Yes.	The ID number of the main node.
contentobject_id	string	Yes.	The ID number of the content object.
contentobject_version	string	Yes.	The number of the published version.
contentobject_is_published	string	Yes.	The published status of the object (0=not published, 1=published).
depth	string	Yes.	The depth of the node within the content node tree. The depth of a top level node is 1.
sort_field	string	Yes.	The sorting method used to sort the child nodes.
sort_order	string	Yes.	The sorting order used when sorting the node's children.
priority	string	Yes.	The node's priority (positive or negative integer).
modified_subnode	string	Yes.	A UNIX timestamp pinpointing the exact time a sub node was changed.
path_string	string	Yes.	The node's path string (for example "/1/2/44").
path_identification_	string	Yes.	The node's path

string			identification string in the initial/ main language (for example "company/ about_us").
remote_id	string	Yes.	A unique ID for the node (avoids crashes when importing/ exporting nodes). A remote ID is an MD5 hash of the time when the node was generated plus a random value.
is_hidden	string	Yes.	The node's hidden status (0=visible, 1=hidden).
is_invisible	string	Yes.	The node's visibility status (0=visible, 1=hidden by superior).
name	string	No.	The name of the object the node encapsulates (for example "My article").
data_map	array	No.	The object's attributes as ezcontentobjectattribute (page 1042) objects.
object	object	No.	The actual content object (as ezcontentobject (page 1032)) that the node encapsulates.
subtree	array	No.	All the nodes that are below this node as ezcontentobjecttreenode (page 1051) objects.
children	array	No.	Array of nodes that are directly below this node as ezcontentobjecttreenode (page

5

				[1051](#)) objects.
children_count	string	No.		The num- ber of nodes that are directly below this node.
view_count	integer	No.		The number of times the node has been viewed.
contentobject_ version_object	object	No.		The current ver- sion (as ezcontento- bjectversion (page [1056](#))) of the object that the node encapsulates.
sort_array	array	No.		The node's sort ar- ray.
can_read	boolean	No.		Returns TRUE if the current user has read access to the node (FALSE otherwise).
can_pdf	boolean	No.		Not documented yet.
can_create	boolean	No.		Returns TRUE if the current user can cre- ate nodes under this node (FALSE other- wise).
can_edit	boolean	No.		Returns TRUE if the current user has edit access to the node (FALSE otherwise).
can_hide	boolean	No.		Returns TRUE if the current user can modify the hid- den state of the node (FALSE otherwise).
can_remove	boolean	No.		Returns TRUE if the current user can remove the node (FALSE otherwise).
can_move	boolean	No.		Returns TRUE if the cur- rent user can move the node to another location (FALSE oth-

			erwise).
can_move_from	boolean	No.	Same as "can_move", returns TRUE if the current user has permissions to move node, FALSE otherwise.
can_add_location	boolean	No.	Not documented yet.
can_remove_location	boolean	No.	Not documented yet.
can_view_embed	boolean	No.	Not documented yet.
is_main	boolean	No.	Not documented yet.
creator	object	No.	The object (as ezcontentobject (page 1032)) containing the user who created the node.
path	array	No.	Array containing the nodes that make up the path as ezcontentobjecttreenode (page 1051) objects. The current node is not included.
path_array	array	No.	Array of strings revealing the ID numbers of the nodes that make up the path. The current node is also included.
parent	object	No.	The parent node (as ezcontentobjecttreenode (page 1051)).
url	string	No.	The URL of the node either as a virtual URL ("company/about_us") or a system URL ("content/view/full/13") depending on a con-

				figuration setting.
url_alias	string	No.		The virtual URL of the node in the current language (which is determined according to the list of prioritized languages), for example "company/about_us".
class_identifier	string	No.		The identifier of the class which the object encapsulated by the node is an instance of (for example "product_review").
class_name	string	No.		The name of the class which the object encapsulated by the node is an instance of (for example "Product review").
hidden_invisible_string	string	No.		The visibility status of a node ("-/-" = completely visible, "H/X" = hidden by user and thus invisible, "-/X" = hidden by superior).
hidden_status_string	string	No.		The visibility status of the node: "Visible", "Hidden" or "Hidden by superior".

* Indicates whether the attribute provides a pre-fetched value or if it needs to be computed upon request.

5.6.16   ezcontentobjectversion

Summary

Contains information about a version of a content object.

Attributes

Attribute	Type	Static*	Description
id	string	Yes.	The ID number of the version.
contentobject_id	string	Yes.	The ID number of the object that the version belongs to.
creator_id	string	Yes.	The ID number of the object that represents the user who created the version.
version	string	Yes.	The actual version number.
status	string	Yes.	The status of the version.
created	string	Yes.	A UNIX timestamp pinpointing the exact date/time when the version was created.
modified	string	Yes.	A UNIX timestamp pinpointing the exact date/time when the version was last modified.
workflow_event_pos	string	Yes.	DEPRECATED - was related to workflows.
user_id	string	Yes.	DEPRECATED - was related to workflows.
language_mask	string	Yes.	The sum of the ID numbers of the languages in which the version exists plus 1.
initial_language_id	string	Yes.	The ID number of the language which the ver-

5

			sion was edited in (for example 16).
creator	object	No.	The object (as ez-contentobject (page 1032) object) that represents the user who created the version.
name	string	No.	The name of the version (generated using the object name pattern).
version_name	string	No.	The name of the version (generated using the object name pattern).
main_parent_node_id	string	No.	The ID number of the main parent node that references the object which the version belongs to.
contentobject_ attributes	array	No.	The attributes (as ezcontento-bjectattribute (page 1042) objects) that the version consists of. Only attributes in the current language (which is determined according to the list of prioritized languages) will be fetched.
related_ contentobject_array	array	No.	An array of ezcontentobject (page 1042) objects representing the objects that are related to the object that the version belongs to.
reverse_related_ object_list	array	No.	An array of ezcontentobject (page 1042) objects representing the objects that are related to the object which

			the version belongs to.
parent_nodes	array	No.	The parent nodes (as eznodeassignment (page 1094) objects) of the nodes which reference the object that the version belongs to.
can_read	boolean	No.	Returns TRUE if the current user has read access to the version. Otherwise FALSE is returned.
can_remove	boolean	No.	Not documented yet.
data_map	array	No.	A hash containing the attributes (as ezcontentobjectattribute (page 1042) objects) that the version consists of. The keys of the hash are the identifiers of the attributes.
node_assignments	array	No.	An array of node assignments (as eznodeassignment (page 1094) objects) that are connected with the object which the version belongs to.
contentobject	object	No.	The object (as ezcontentobject (page 1032)) that the version belongs to.
initial_language	object	No.	The language which the version was edited in (as ezcontentlanguage (page 1031) object).
language_list	array	No.	The translations (as ezcontentobjecttranslation (page

			1046) objects) that belong to the version - including the default translation.
translation	object	No.	DEPRECATED - was related to old multi-language functionality.
translation_list	array	No.	The translations (as ezcontentobjecttranslation (page 1046) objects) that belong to the version - the default translation is not included.
complete_translation_list	array	No.	Same as the "translation_list" attribute.
temp_main_node	object	No.	A temporary node (as ezcontentobjecttreenode (page 1051) object) for the object that the version belongs to. The temporary node does not exist in the database.

* Indicates whether the attribute provides a pre-fetched value or if it needs to be computed upon request.

5.6.17   ezcurrencydata

Summary

Contains information about a currency.

Attributes

Attribute	Type	Static*	Description
id	string	Yes.	The ID number of the currency.
code	string	Yes.	The three-character currency code (for example "USD").
symbol	string	Yes.	A string that is used as the currency symbol (for example "$").
locale	string	Yes.	A locale that is used for price formatting (for example "eng-US").
status	string	Yes.	Returns 0 if the currency is inactive, 1 otherwise.
auto_rate_value	string	Yes.	The currency rate that is retrieved via automatic update of the exchange rates from the Internet.
custom_rate_value	string	Yes.	The currency rate that is specified manually (0.00000 by default).
rate_factor	string	Yes.	The rate factor value (1.00000 by default). If the custom rate is 0.00000 or not specified then the auto rate will be multiplied by the rate factor.
rate_value	string	No.	The actual rate value that will be used in calculations. The

			rate value will be the same as the custom rate if it is not zero. If the custom rate is 0.00000 or not specified then the auto rate will be multiplied by the rate factor.

* Indicates whether the attribute provides a pre-fetched value or if it needs to be computed upon request.

5

5.6.18   ezdate

Summary

Contains information about a date.

Attributes

Attribute	Type	Static*	Description
timestamp	string	Yes.	The date as a UNIX timestamp (for example "567990000") if the date is a valid date, NULL otherwise.
is_valid	boolean	Yes.	Returns TRUE if the date is a valid date, FALSE otherwise.
year	string	No.	The year (for example "1988").
month	string	No.	The month (for example "01").
day	string	No.	The day (for example "01").

* Indicates whether the attribute provides a pre-fetched value or if it needs to be computed upon request.

5.6.19   ezdatetime

Summary

Contains information about a date and time.

Attributes

Attribute	Type	Static*	Description
timestamp	string	Yes.	The date/time value as a UNIX timestamp (for example "1147719660") if the date/time is valid.
hour	string	No.	The hour (for example "21").
minute	string	No.	The minute (for example "01").
year	string	No.	The year (for example "2006").
month	string	No.	The month (for example "05").
day	string	No.	The day (for example "15").
is_valid	boolean	Yes.	Returns TRUE if the date/time value is valid, FALSE otherwise.

* Indicates whether the attribute provides a pre-fetched value or if it needs to be computed upon request.

5

5.6.20   ezimagealiashandler

Summary

Contains information about an image.

Attributes

Attribute	Type	Static*	Description
alternative_text	string	No.	The alternative image text (for example "Picture of an apple.").
original_filename	string	No.	The original name of the image file (for example "apple.png").
is_valid	string	No.	Returns either 1 or 0 (valid or invalid).
name of variation	array	No.	**Attribute:** name **Type:** string **Description:** The name of the variation (for example "original"). **Attribute:** width **Type:** string **Description:** The width as number of pixels (for example "320"). **Attribute:** height **Type:** string **Description:** The height as number of pixels (for example "256"). **Attribute:** mime_type **Type:** string **Description:** The MIME type (for ex-

5

ample "image/
png").

Attribute: filename
Type: string
Description: The
name of the file
(for example "my_
image.png").

Attribute: suffix
Type: string
Description: The
file suffix (for exam-
ple "png").

Attribute: dirpath
Type: string
Description: The
path
to the image (for ex-
ample "var/storage/
images/test/
199-2-eng-GB").

Attribute:
basename
Type: string
Description: The
basename of the im-
age file (for example
"apple").

Attribute:
alternative_text
Type: string
Description: The al-
ternative image text
(for example "Pic-
ture of an apple.").

Attribute: text
Type: string
Description: Con-
tains the
"alternative_text" of

5

the original image.

Attribute: original_
filename
Type: string
Description: The
name of the origi-
nal file (for example
"apple.png").

Attribute: url
Type: string
Description: Com-
plete path + name
of image file (for ex-
ample "var/storage/
images/test/
199-2-eng-GB/
apple.png").

Attribute: alias_key
Type: string
Description: A in-
ternal CRC32 value
which is used when
an alias is created.
This value is based
on the filters that
were used (param-
eters included) and
is checked when an
alias is accessed. If
this values differs
from the configured
filters (in image.ini
or an override), the
system will recreate
the alias.

Attribute: times-
tamp
Type: string
Description:
A UNIX timestamp
pinpointing the ex-
act date/time when

5

the alias was last modified. For the "original" alias, the timestamp will reveal the time when the image was originally uploaded.

Attribute: full_path
Type: string
Description: Complete path + name of image file (for example "var/storage/ images/test/ 199-2-eng-GB/ apple.png").

Attribute: is_valid
Type: string
Description: TRUE if the alias was created properly, that means all conversion and filters were applied without problems. It will be FALSE if the image manager is wrongly configured or if there no compatible image converters could be found.

Attribute: is_new
Type: boolean
Description: Will be set to TRUE the first time the alias is created, the next time (reload of the same page) it will be FALSE. It will also be set to TRUE every time the alias is re-

			created due to changes in filters (see alias_key).
			Attribute: filesize **Type:** integer **Description:** The number of bytes that the image consumes.
			Attribute: info **Type:** string **Description:** Contains extra information about the image, depending on the image type. It will typically contain EXIF information from digital cameras or information about animated GIFs. If there is no information, the info will be a boolean FALSE.

* Indicates whether the attribute provides a pre-fetched value or if it needs to be computed upon request.

5

5.6.21 ezimagelayer

Summary

Contains information about an image layer.

Attributes

Attribute	Type	Static*	Description
filepath	string	Yes.	The path (for example ”design/ example/images”).
filename	string	Yes.	The name of the image file (for example ”delorean.png”).
width	integer	Yes.	The width of the image (number of pixels).
height	integer	Yes.	The height of the image (number of pixels).
alternative_text	string	Yes.	The alternative image text.
imagepath	string	No.	The internal (eZ publish) path to the image (for example ”design/ example/images/ delorean.png”).
has_size	boolean	No.	TRUE if ”width” and ”height” is set, otherwise FALSE.

* Indicates whether the attribute provides a pre-fetched value or if it needs to be computed upon request.

5

5.6.22   ezimageobject

Summary

Contains information about an image.

Attributes

Attribute	Type	Static*	Description
filepath	string	Yes.	The path (for example "design/ example/images").
filename	string	Yes.	The name of the image file (for example "delorean.png").
width	integer	Yes.	The width of the image (number of pixels).
height	integer	Yes.	The height of the image (number of pixels).
alternative_text	string	Yes.	The alternative image text.
imagepath	string	No.	The internal (eZ publish) path to the image (for example "design/ example/images/ delorean.png").
has_size	boolean	No.	TRUE if "width" and "height" is set, otherwise FALSE.

* Indicates whether the attribute provides a pre-fetched value or if it needs to be computed upon request.

5

5.6.23   ezinformationcollection

Summary

Contains information about a block of collected information.

Attributes

Attribute	Type	Static*	Description
id	string	Yes.	The ID number of the information collection.
contentobject_id	string	Yes.	The ID number of the object that collected the information.
user_identifier	string	Yes.	An unique identifier of the user that submitted the information (an MD5 hash of the user's ID number for registered users; session identifier for anonymous users).
creator_id	string	Yes.	The ID number of the object representing the user who submitted the information (for example, 14).
created	string	Yes.	A UNIX timestamp revealing the exact date/time when the information was collected.
modified	string	Yes.	A UNIX timestamp revealing the exact date/time when the collection was last modified.
attributes	array	No.	An array of the collection attributes (as ezinformationcollectionattribute (page

				1073) objects).
data_map	array	No.		A hash containing the attributes (as ezinformationcollectionattribute (page 1073) objects) that the collection consists of. The keys of the hash are the identifiers of the attributes.
object	object	No.		The actual object that collected the information (as an ezcontentobject (page 1032) object).
creator	object	No.		The user who submitted the information (as ezuser (page 1117) object).

* Indicates whether the attribute provides a pre-fetched value or if it needs to be computed upon request.

5.6.24 ezinformationcollectionattribute

Summary

Contains information about an attribute of a collection.

Attributes

Attribute	Type	Static*	Description
id	string	Yes.	The ID number of the information collection attribute.
informationcollection_id	string	Yes.	The ID number of the information collection itself.
contentclass_attribute_id	string	Yes.	The ID number of the class attribute.
contentobject_attribute_id	string	Yes.	The ID number of the object attribute.
contentobject_id	string	Yes.	The ID number of the object.
data_text	string	Yes.	Collected text.
data_int	string	Yes.	Collected integer.
data_float	string	Yes.	Collected float.
contentclass_attribute_name	string	No.	The name of the attribute that collected the information.
contentclass_attribute	object	No.	The class attribute (as an ezcontentclassattribute (page 1026) object).
contentobject_attribute	object	No.	The object attribute (as an ezcontentobjectattribute (page 1042) object).
contentobject	object	No.	The content object (as an ezcontentobject (page 1032) object).
result_template	string	No.	The name of the result template (for example "ezstring").
has_content	boolean	No.	Returns TRUE if the attribute

			is not empty, FALSE otherwise.
content	string	No.	The actual content of the attribute.
class_content	any	No.	The content of the class attribute which the collection attribute is made from.

* Indicates whether the attribute provides a pre-fetched value or if it needs to be computed upon request.

5.6.25   ezkeyword

Summary

Contains information about keywords.

Attributes

Attribute	Type	Static*	Description
keywords	array	Yes.	An array of strings containing the keywords/phrases.
keyword_string	string	No.	The actual keyword string (comma separated keywords/ phrases).
related_nodes	array	No.	An array of nodes (as ezcontentobject-treenode (page 1051) objects) that have at least one of the same keywords.
related_objects	array	No.	(Deprecated from 3.6.1, use related_nodes) An array of nodes (as ezcontentobject-treenode (page 1051) objects) that have at least one of the same keywords.

* Indicates whether the attribute provides a pre-fetched value or if it needs to be computed upon request.

*Note: When an **ezkeyword** attribute is present in a content class, and a content object of this class has this attribute filled in, it is used to fill the 'keyworkd' HTML meta element.*

5

5.6.26 ezlocale

Summary

Contains information about a locale.

Attributes

Attribute	Type	Static*	Description
charset	string	Yes.	The character set that the locale uses (for example "iso-8859-1").
allowed_charsets	array	Yes.	An array of strings containing the allowed character sets.
country_name	string	Yes.	The name of the country that the locale belongs to (for example "United Kingdom").
country_comment	string	Yes.	A comment about the country that the locale belongs to (usually empty).
country_code	string	Yes.	The country code (for example "GB" for Great Britian).
country_variation	string	Yes.	The country variation.
language_name	string	Yes.	The native name of the language (for example "Norsk", "Magyar", etc.).
intl_language_name	string	Yes.	The international name of the language (for example "Norwegian", "Hungarian", etc.).
language_code	string	Yes.	The language code (for example "eng").
language_comment	string	Yes.	A comment about

			the language itself (usually empty).
locale_code	string	Yes.	The actual locale code (for example "eng-GB", "nor-NO", etc.).
locale_full_code	string	No.	The full locale code (for example "eng-GB").
http_locale_code	string	No.	The HTTP locale code (for example "eng-GB").
decimal_symbol	string	Yes.	The decimal symbol (for example a dot ".").
thousands_separator	string	Yes.	The character (if any) that is used to separate/split large numbers.
decimal_count	string	Yes.	The number of decimal digits that should be displayed.
negative_symbol	string	Yes.	The symbol used for displaying negative numbers (usually just a dash: "-").
positive_symbol	string	Yes.	The symbol used for displaying positive numbers (usually empty).
currency_decimal_ symbol	string	Yes.	The symbol used for separating the integer part from the decimal part of currency values.
currency_thousands_ separator	string	Yes.	The thousand separator used for currencies.
currency_decimal_ count	string	Yes.	The number of decimal digits that should be included when displaying currencies.
currency_negative_ symbol	string	Yes.	The symbol used for displaying negative currencies (usu-

			ally just a dash: "-").
currency_positive_symbol	string	Yes.	The symbol used for displaying positive currencies (usually empty).
currency_symbol	string	Yes.	The currency symbol (for example "£").
currency_name	string	Yes.	The name of the currency (for example "British Pound", "Norwegian Kroner", etc.).
currency_short_name	string	Yes.	A short/abbreviated name for the currency (for example "BSP", "NOK", etc.).
is_monday_first	boolean	Yes.	Returns TRUE if monday is considered to be the first day of the week, FALSE otherwise.
weekday_name_list	array	No.	An array of strings containing the weekday names (for example "Monday", "Tuesday", etc.).
weekday_short_name_list	array	No.	An array of strings containing abbreviated weekday names (for example "Mon", "Tue", etc.).
weekday_number_list	array	Yes.	An array of strings containing the weekday numbers (for example "0", "1", etc.).
month_list	array	Yes.	An array of strings containing the month digits (for example "1" for January, "2" for February, etc.).
month_name_list	array	Yes.	An array of strings containing the name of the months (for example "January",

5

			"February", etc.).
is_valid	boolean	Yes.	Returns TRUE if the locale is valid (successfully read from disk), FALSE otherwise (unknown locale).

* Indicates whether the attribute provides a pre-fetched value or if it needs to be computed upon request.

5.6.27 ezmatrix

Summary

Contains information about a matrix.

Attributes

Attribute	Type	Static*	Description
name	boolean	Yes.	Always FALSE (this attribute is currently not used).
rows	array	No.	A collection of miscellaneous structures that contain information about the rows. Currently there is only one structure, called "sequential". It is built up of an array of hashes. The following table reveals the structure of the array elements. **Name:** identifier **Type:** string **Description:** The identifier of the column (defined at the class level). **Name:** name **Type:** string **Description:** The name of the column (defined at the class level). **Name:** columns **Type:** array **Description:** An array of strings holding the actual contents of the columns.

| columns | array | No. | A collection of miscellaneous structures that contain information about the rows. |

Currently there are two types of structures: "sequential" and "id". The "sequential" structure is built up of an array hashes. The following table reveals the structure of the array elements.

Name: identifier
Type: string
Description: The identifier of the column (defined at the class level).

Name: index
Type: string
Description: The row index ("0", "1", and so on).

Name: name
Type: string
Description: The name of the column (defined at the class level).

Name: rows
Type: array
Description: An array of strings holding the actual contents of the rows.

The "id" structure consists of hash where the

			keys are the column identifiers. The following table shows the structure that is available for each column identifier. **Name:** identifier **Type:** string **Description:** The identifier of the column (defined at the class level).
			Name: index **Type:** string **Description:** The column index ("0", "1", and so on).
			Name: name **Type:** string **Description:** The name of the column (defined at the class level).
			Name: rows **Type:** array **Description:** An array of strings holding the actual contents of the rows.
cells	array	Yes.	A flat array of the cells that make up the matrix (from left to right, top to bottom).
matrix	array	Yes.	Consists of "rows", "columns" and "cells" (see above).
rowCount	integer	No.	The number of rows.
columnCount	integer	No.	The number of columns.

* Indicates whether the attribute provides a pre-fetched value or if it needs to be computed upon request.

5.6.28 ezmedia

Summary

Contains information about a video file.

Attributes

Attribute	Type	Static*	Description
contentobject_attribute_id	string	Yes.	The identification number of the content object attribute.
version	string	Yes.	The content object version.
filename	string	Yes.	The name of the file in the eZ publish var directory (for example "44b963c9e8d1ffa80cbb08e84d5
original_filename	string	Yes.	The original filename (for example "ezpublish-mediademo.avi").
mime_type	string	Yes.	The MIME type (for example "video/ x-msvideo").
width	string	Yes.	The playback width - in number of pixels (for example "640").
height	string	Yes.	The playback height - in number of pixels (for example "480").
has_controller	string	Yes.	Either 1 or 0 (show controller or do not show controller).
controls	string	Yes.	Either 1 or 0 - Real Media specific - controls the control-bar.
is_autoplay	string	Yes.	Either 1 or 0 (automatically start playback or not).
pluginspage	string	Yes.	A URL that leads to the plug-in that is required for proper

5

			playback.
quality	string	Yes.	Flash specific - controls the quality of the media.
is_loop	string	Yes.	Either 1 or 0 (looped playback or single-cycle).
filesize	integer	No.	The number of bytes the media file consumes.
filepath	string	No.	The path to the media file (for example "var/storage/ original/video/ 44b963c9e8d1ffa80cbb08e84d57
mime_type_category	string	No.	The MIME type category (for example "video").
mime_type_part	string	No.	The MIME type part (for example "x-msvideo").

* Indicates whether the attribute provides a pre-fetched value or if it needs to be computed upon request.

5

5.6.29 ezmultioption

Summary

DEPRECATED (Contains information about multilpe options.)

Attributes

Attribute	Type	Static*	Description
name	string	Yes.	The name of the entire multi-option set.
multioption_list	array	Yes.	An array of hashes where each hash consists of the following elements: **Attribute:** id **Type:** integer **Description:** The ID number of the option. **Attribute:** name **Type:** string **Description:** The name of the option (for example "Color"). **Attribute:** priority **Type:** string **Description:** The option's priority. **Attribute:** default_option_id **Type:** string **Description:** The ID number of the default option. **Attribute:** optionlist **Type:** array **Description:** Array of hashes - the structure is described in

			the documentation page for the ezoption (page 1096) object.

* Indicates whether the attribute provides a pre-fetched value or if it needs to be computed upon request.

5

5.6.30 ezmultioption2

Summary

Contains information about a group of multi-options.

Attributes

The ezmultioption2 objects can be used to store information about the following:

- Ordinary groups of multi-options (top level groups that are described in the reference documentation of the "Multi-option2 (page 563)" datatype; these groups can be managed from within the administration interface).

- A virtual root group that is located one level above the ordinary group (the parent of all groups); it is not visible from within the administration interface.

- Virtual subgroups of multi-options (such a group contains all multi-options that are located one level beneath an existing multi-option; subgroups are not visible from within the administration interface).

For example, let's say the site administrator creates the following multi-level structure with two groups of multi-options called "Exterior" and "Interior":

```
Exterior
- Body: Sedan, Combi, Hatchback
- Color: Black, Red
-- Bumpers: Same as body, Non-painted
-- Mirrors: Silver, Same as body, Non-painted

Interior
- Seats: Make your choice, Cloth, Leather
```

In this case, the system will create "ezmultioption2" objects that store information about top level groups ("Exterior" and "Interior"), their parent group (virtual root group), plus an unnamed subgroup that contains the "Bumpers" and "Mirrors" multi-options.

The following table reveals the structure of "ezmultioption2" objects.

Attribute	Type	Static*	Description
name	string	Yes.	The name of the group (for example "Exterior") if the object represents a top level group of multi-options; otherwise

5

			an empty string will be returned (subgroups and root groups are unnamed).
id	integer	Yes.	Not in use.
group_id	integer	Yes.	The ID number of the group/subgroup (for example 2). 0 means that the object represents a root group.
rules	array	Yes.	The dependency rules for nested multi-options as an array of arrays if the object represents a root group; otherwise an empty array will be returned.
multioption_list	array	Yes.	The multi-options that belong to the group/subgroup as an array of hashes. (If the object represents a root group, an empty array will be returned.) Each hash describes a multi-option located at the first level in the group/subgroup. The hash-structure consists of the following elements: **Attribute:** id **Type:** integer **Description:** The internal ID number of the multi-option (unique within the group/subgroup that the multi-option belongs to).

Attribute: multioption_id
Type: string
Description: The ID number of the multioption (for example 2).

Attribute: name
Type: string
Description: The name of the multioption (for example "Color").

Attribute: priority
Type: string
Description: Reserved for future use.

Attribute: default_option_id
Type: string
Description: The ID number of the default option.

Attribute: imageoption
Type: integer
Description: 1 if one or more images are added to the options in the multioption, FALSE otherwise.

Attribute: optionlist
Type: array
Description: The options that belong to the multioption as an array of hashes. Each hash

consists of the following elements:

Attribute: id
Type: integer
Description: The internal ID number of the option (unique within the multi-option that the option belongs to).

Attribute: option_id
Type: string
Description: The ID number of the option.

Attribute: value
Type: string
Description: The option text (for example "Red").

Attribute: additional_price
Type: string
Description: Additional price (for example "100").

Attribute: is_selectable
Type: string
Description: 1 if the option is not disabled, otherwise 0.

Attribute: child_group
Type: object
Description: The subgroup (as an ezmultioption2 object) that contains all multi-options lo-

			cated one level beneath this one. If there are no nested multi-options, this element will not be returned.
optiongroup_list	array	Yes.	The top level groups (as ezmultioption2 objects) if the object represents a root group; otherwise an empty array will be returned.

5

5.6.31   ezmultiprice

Summary

Contains information about prices and currencies for a product.

Attributes

Attribute	Type	Static*	Description
vat_type	array	No.	The available VAT types (as an array of ezvattype (page 1121) objects).
current_user	object	No.	The current user (as an ezuser (page 1117) object).
is_vat_included	boolean	Yes.	TRUE if the VAT is included, FALSE otherwise.
vat_percent	string	No.	The VAT percentage (without the percent symbol).
discount_percent	double	Yes.	The discount percentage.
has_discount	boolean	No.	TRUE if a discount rule affects the price, FALSE otherwise.
selected_vat_type	object	No.	The selected VAT type (as an ezvattype (page 1121) object).
price	string	No.	The price in the preferred currency.
inc_vat_price	string	No.	The price in the preferred currency including the VAT.
ex_vat_price	double	No.	The price in the preferred currency excluding the VAT.
discount_price_inc_vat	double	No.	The discounted price in the preferred currency including the VAT.
discount_price_ex_	double	No.	The discounted price

5

vat			in the preferred currency excluding the VAT.
currency	string	No.	The code of the preferred currency.
currency_list	array	No.	The available currencies (as an array of ezcurrencydata (page 1060) objects).
auto_currency_list	array	No.	The currencies that the auto prices are calculated for (as an array of ezcurrencydata (page 1060) objects).
price_list	array	No.	The prices in all currencies (both custom prices and auto prices).
auto_price_list	array	No.	The auto prices.
custom_price_list	array	No.	The custom prices.
inc_vat_price_list	array	No.	The prices including the VAT.
ex_vat_price_list	array	No.	The prices excluding the VAT.
discount_inc_vat_price_list	array	No.	The discounted prices including the VAT.
discount_ex_vat_price_list	array	No.	The discounted prices excluding the VAT.

* Indicates whether the attribute provides a pre-fetched value or if it needs to be computed upon request.

5

5.6.32   eznodeassignment

Summary

Contains information about a node assignment.

Attributes

Attribute	Type	Static*	Description
id	string	Yes.	The ID number of the node assignment.
remote_id	string	Yes.	The remote ID of the node assignment.
contentobject_id	string	Yes.	The ID number of the object that the node assignment belongs to.
contentobject_version	string	Yes.	The version number which the node assignment belongs to.
parent_node	string	Yes.	The ID number of the parent node.
sort_field	string	Yes.	The ID number of the method for sorting child nodes.
sort_order	string	Yes.	Either 1 (ascending) or 0 (descending). Reveals the order for sorting child nodes.
is_main	string	Yes.	Either 1 or 0. 1 means that this is the main node assignment for the object.
from_node_id	string	Yes.	The ID number of the original node. This attribute will only have a valid value if an existing node is moved, in which case the ”parent_node_id” will reveal the new parent node.

5

			If the attribute contains "0" or "-1", it means that the node is not meant to be moved.
parent_remote_id	string	Yes.	The remote ID of the parent node.
op_code	string	Yes.	Not documented yet.
parent_node_obj	object	No.	The parent node (as ezcontentobjecttreenode (page 1051) object).
parent_contentobject	object	No.	The object (as ezcontentobject (page 1032) object) that is referenced by the parent node.
node	object	No.	The actual node (as ezcontentobjecttreenode (page 1051)) that this assignment assigns to the object.
is_nop_operation	boolean	No.	Not documented yet.
is_create_operation	boolean	No.	Not documented yet.
is_move_operation	boolean	No.	Not documented yet.
is_remove_operation	boolean	No.	Not documented yet.
is_set_operation	boolean	No.	Not documented yet.
temp_node	object	No.	A temporary node (as ezcontentobjecttreenode (page 1051) object) for the object that the version belongs to. The temporary node does not exist in the database.

* Indicates whether the attribute provides a pre-fetched value or if it needs to be computed upon request.

5.6.33   ezoption

Summary

Contains information about a collection of options.

Attributes

Attribute	Type	Static*	Description
name	string	Yes.	The name of the option (for example "Color").
option_list	array	Yes.	An array of hashes where each hash consits of the following elements: **Attribute:** id **Type:** integer **Description:** The ID number of the option. **Attribute:** value **Type:** string **Description:** The option text (for example "Red", "Green", etc.). **Attribute:** additional_price **Type:** string **Description:** The addition price. **Attribute:** is_default **Type:** boolean **Description:** TRUE if it is the default option, FALSE otherwise.

* Indicates whether the attribute provides a pre-fetched value or if it needs to be computed upon request.

5.6.34 ezorder

Summary

Contains information about an order.

Attributes

Attribute	Type	Static*	Description
id	string	Yes.	The ID number of the order.
order_nr	string	Yes.	The number (count) of the order.
is_temporary	string	Yes.	1 if the order is temporary, 0 otherwise.
user_id	string	Yes.	The ID number of the user that the order belongs to.
productcollection_id	string	Yes.	The ID number of the product collection that belongs to the order.
data_text_1	string	Yes.	General purpose text block for the shop account handler. The default shop handler uses this attribute to store an XML structure with the customer information (name, address, etc.). Example:

			`<xml version="1.0" encoding="UTF-8"?> <shop_account>`
			
			`<first-name>Marty</first-name>`
			
			`<last-name>McFly</last-name>`
			
			`<email>marty@ez.no</email>`
			`<street1>7 Lyon Estates</street1>`
			
			`<street2>Outside Hill Valley</street2>`
			`<zip>55532</zip>`
			`<place>Hill Valley</place>`
			
			`<state>California</state>`
			`<country>USA</country>`
			`<comment>No comment.</comment>`
			`</shop_account>`

data_text_2	string	Yes.	General purpose text block #2 for the shop account handler. Similar to "data_text_1".
account_identifier	string	Yes.	The account identifier (for example "ez").
created	string	Yes.	A UNIX timestamp pinpointing the exact date/time when the order was cre-

			ated.
ignore_vat	string	Yes.	1 if the VAT should be ignored, 0 otherwise.
email	string	Yes.	The E-mail address of the buyer.
status_id	string	Yes.	The global ID number of the order's current status.
status_modified	string	Yes.	A UNIX timestamp pinpointing the exact date/time when the status was order's status was last modified.
status_modifier_id	string	Yes.	The ID number of the user who last modified the order's status.
is_archived	string	Yes.	Not documented yet.
status_name	string	No.	The name of the order's current status (for example "Pending").
status	object	No.	The actual order status (as an ezorderstatus (page 1102) object).
status_modification_list	array	No.	The status log as an array of ezorderstatus (page 1102) objects.
product_items	array	No.	Not documented yet.
order_items	array	No.	An array of order items that are associated with the order (as ezorderitem (page 1101) objects).
product_total_inc_vat	double	No.	Not documented yet.
product_total_ex_vat	double	No.	Not documented yet.
total_inc_vat	double	No.	Not documented

			yet.
total_ex_vat	double	No.	Not documented yet.
user	object	No.	Not documented yet.
account_view_ template	string	No.	Not documented yet.
account_information	array	No.	Not documented yet.
account_name	string	No.	Not documented yet.
account_email	string	No.	Not documented yet.
productcollection	object	No.	Not documented yet.

* Indicates whether the attribute provides a pre-fetched value or if it needs to be computed upon request.

5

5.6.35 ezorderitem

Summary

Contains information about order items.

Attributes

Attribute	Type	Static*	Description
id	string	Yes.	The ID number of the order item.
order_id	string	Yes.	The ID number of the order that the order item is associated with.
description	string	Yes.	The description of the order item.
price	string	Yes.	The price of the order item.
vat_value	string	Yes.	The VAT value.
is_vat_inc	string	Yes.	1 if the price includes the VAT, 0 otherwise.
type	string	Yes.	The type of the order item. This string reveals which part of the system that added the item (for example "ezsimple-shipping").
price_inc_vat	integer	No.	The price of the order item including the VAT.
price_ex_vat	integer	No.	The price of the order item excluding the VAT.

* Indicates whether the attribute provides a pre-fetched value or if it needs to be computed upon request.

5

5.6.36   ezorderstatus

Summary

Contains information about an order status.

Attributes

Attribute	Type	Static*	Description
id	string	Yes.	The ID number of the status.
status_id	string	Yes.	The global ID number of the status.
name	string	Yes.	The name of the status (for example "Delivered").
is_active	string	Yes.	1 if the status is active, 0 otherwise.
is_internal	boolean	No.	TRUE if the status is one of the built-in/internal statuses, FALSE if it is a custom status.

* Indicates whether the attribute provides a pre-fetched value or if it needs to be computed upon request.

5

5.6.37 ezpolicy

Summary

Contains information about a policy.

Attributes

Attribute	Type	Static*	Description
id	string	Yes.	The ID number of the policy.
role_id	string	Yes.	The ID number of the role the policy belongs to.
module_name	string	Yes.	The name of the module that the policy grants access to (for example "content").
function_name	string	Yes.	The name of the function that the policy grants access to (for example "read").
limitations	array	No.	The limitations (if any) of the policy.
role	object	No.	The role (as ezrole (page 1109) object) that the policy belongs to.
limit_identifier	string	No.	Not documented yet.
limit_value	string	No.	Not documented yet.
user_role_id	string	No.	Not documented yet.

* Indicates whether the attribute provides a pre-fetched value or if it needs to be computed upon request.

5

5.6.38   ezprice

Summary

Contains information about a price.

Attributes

Attribute	Type	Static*	Description
vat_type	array	No.	The available VAT types (as an array of ezvattype (page 1121) objects).
current_user	object	No.	The current user (as an ezuser (page 1117) object).
is_vat_included	boolean	Yes.	TRUE if the VAT is included, FALSE otherwise.
vat_percent	string	No.	The VAT percentage (without the percent symbol).
discount_percent	double	Yes.	The discount percentage.
has_discount	boolean	No.	TRUE if a discount rule affects the price, FALSE otherwise.
selected_vat_type	object	No.	The selected VAT type (as an ezvattype (page 1121) object).
price	string	Yes.	The entered price.
inc_vat_price	string	No.	The price including the VAT.
ex_vat_price	double	No.	The price excluding the VAT.
discount_price_inc_vat	double	No.	The discounted price including the VAT.
discount_price_ex_vat	double	No.	The discounted price excluding the VAT.
currency	string	Yes.	An empty string.

* Indicates whether the attribute provides a pre-fetched value or if it needs to be computed upon request.

5.6.39 ezproductcategory

Summary

Contains information about a product category.

Attributes

Attribute	Type	Static*	Description
id	integer	Yes.	The ID number of the product category.
name	string	Yes.	The name of the product category.

* Indicates whether the attribute provides a pre-fetched value or if it needs to be computed upon request.

5

5.6.40 ezproductcollectionitem

Summary

Contains information about an item of a product collection.

Attributes

Attribute	Type	Static*	Description
id	string	Yes.	Auto-incremented ID number (used on the database level).
productcollection_id	string	Yes.	The ID number of the product collection.
contentobject_id	string	Yes.	The ID number of the content object.
item_count	string	Yes.	The quantity.
price	string	Yes.	The price of the product (the object).
is_vat_inc	string	Yes.	1 if the price includes the VAT, 0 if not.
vat_value	string	Yes.	The VAT value.
discount	string	Yes.	Discount percentage.
name	string	Yes.	The name of the product (the object).
contentobject	object	No.	The actual content object (as an ezcontentobject (page 1032) object).
option_list	array	No.	An array representing the selected product options.

* Indicates whether the attribute provides a pre-fetched value or if it needs to be computed upon request.

5

5.6.41   ezrangeoption

Summary

Contains information about a range of options.

Attributes

Attribute	Type	Static*	Description
name	string	Yes.	The name of the range option (for example "Shoe size").
start_value	string	Yes.	The start value (for example "32").
stop_value	string	Yes.	The stop value (for example "40").
step_value	string	Yes.	The step value (for example "1").
option_list	array	Yes.	The generated options as an array of hashes where each hash consists of the following elements: **Attribute:** id **Type:** integer **Description:** The ID of the option. **Attribute:** value **Type:** string **Description:** The option value. **Attribute:** additional_price **Type:** integer **Description:** The additional price (will always be 0). **Attribute:** is_default **Type:** boolean **Description:** TRUE if it is the default option, FALSE other-

5

			wise.

* Indicates whether the attribute provides a pre-fetched value or if it needs to be computed upon request.

5

5.6.42   ezrole

Summary

Contains information about a role.

Attributes

Attribute	Type	Static*	Description
id	string	Yes.	The ID number of the role.
version	integer	Yes.	The current version of the role, 0 is the currently active role while any other values are temporary versions.
name	string	Yes.	The name of the role (for example "Anonymous").
is_new	integer	Yes.	The creation state of the role, will be 1 if the role just got created but have not been activated yet, 0 otherwise.
policies	array	No.	The policies (as ezpolicy (page 1103) objects) that make up the role.
limit_identifier	string	No.	The identifier of the limited assignment for the currently logged in user. Will be FALSE if the limited assignment is not used.
limit_value	string	No.	The value for the "limit_identifier" attribute (when the limitation feature is in use).
user_role_id	string	No.	Not documented yet.

5

* Indicates whether the attribute provides a pre-fetched value or if it needs to be computed upon request.

5

5.6.43   ezsection

Summary

Contains information about a section.

Attributes

Attribute	Type	Static*	Description
id	string	Yes.	The ID number of the section.
name	string	Yes.	The name of the section (for example "Standard").
navigation_part_ identifier	string	Yes.	The identifier of the navigation part that the section belongs to (for example "ezcontent-navigationpart").
locale	string	Yes.	Not in use.
identifier	string	Yes.	The identifier string of a section. Available since eZ Publish 4.4.

* Indicates whether the attribute provides a pre-fetched value or if it needs to be computed upon request.

5

5.6.44   ezsimplifiedxmlinput

Summary

Contains information about XML data.

Attributes

Attribute	Type	Static*	Description
input_xml	string	No.	The text that the end-user has input to the system.
edit_template_name	string	No.	The name of the template that is used when the object attribute is being edited. The default is "ezxmltext", but can be overridden for a custom handler (for example the Online Editor).
information_template_name	string	No.	The name of the template that will be used when the object attribute is collecting information. The default is "ezxmltext", but can be overridden for a custom handler (for example the Online Editor).
aliased_type	string	Yes.	Returns the name of the original handler. This will normally be FALSE (no alias) - the Online Editor takes control of the XML field using an alias.
alias_handler	string	No.	Returns the original handler if the "aliased_type" at-

5

			tribute is non-false.

* Indicates whether the attribute provides a pre-fetched value or if it needs to be computed upon request.

5

5.6.45   ezsubtreenotificationrule

Summary

Contains information about a subtree notification rule.

Attributes

Attribute	Type	Static*	Description
id	string	Yes.	The ID number of the subtree notification rule.
user_id	string	Yes.	The ID number of the user that the rule belongs to.
use_digest	string	Yes.	When this attribute is "0", the system will check if the user has digest settings set; otherwise the digest settings are ignored and the notification is sent immediately.
node_id	string	Yes.	The ID number of the subscribed node.
node	object	No.	The actual node (as an ezcontentobjecttreenode (page 1051) object).

* Indicates whether the attribute provides a pre-fetched value or if it needs to be computed upon request.

5

5.6.46 eztime

Summary

Contains information about a time value.

Attributes

Attribute	Type	Static*	Description
timestamp	integer	No.	The time as a UNIX timestamp or NULL.
time_of_day	integer	Yes.	Not documented yet.
hour	integer	No.	The hour.
minute	integer	No.	The minute.
is_valid	boolean	Yes.	TRUE if the time is valid, FALSE otherwise.

* Indicates whether the attribute provides a pre-fetched value or if it needs to be computed upon request.

5

5.6.47   ezurl

Summary

Contains information about a URL.

Attributes

Attribute	Type	Static*	Description
id	string	Yes.	The ID number of the URL.
url	string	Yes.	The actual address (for example "http://www.slashdot.org").
original_url_md5	string	Yes.	The MD5 sum of the URL.
is_valid	string	Yes.	Either 1 (valid) or 0 (invalid). Reveals if the URL is valid or not.
last_checked	string	Yes.	A UNIX timestamp revealing when the URL was validated by the system.
created	string	Yes.	A UNIX timestamp pinpointing the exact date/time when the URL was created.
modified	string	Yes.	A UNIX timestamp pinpointing the exact date/time when the URL last modified.

* Indicates whether the attribute provides a pre-fetched value or if it needs to be computed upon request.

5

5.6.48 ezuser

Summary

Contains information about a user.

Attributes

Attribute	Type	Static*	Description
contentobject_id	string	Yes.	The ID number of the object that represents the user.
login	string	Yes.	The username of the user.
email	string	Yes.	The E-mail address of the user (for example "marty@ez.no").
password_hash	string	Yes.	The encrypted version of the user's password.
password_hash_type	string	Yes.	The type of encryption that was used to obfuscate the user's password.
contentobject	object	No.	The actual object (as ezcontentobject (page 1032)) that represents the user.
groups	array	No.	The object ID numbers of the user groups that the user is a member of.
has_stored_login	boolean	No.	Returns TRUE if the user has a non-empty username stored in the database; otherwise FALSE will be returned.
original_password	string	No.	The password input by the user from the last page (from "/user/register" or

			”/content/edit”). It is only used for validation of the password. It will be FALSE if empty input was provided.
original_password_confirm	string	No.	The confirmation password for the ”original_password” attribute (FALSE if empty).
roles	array	No.	The roles (as ezrole (page 1109) objects) that are assigned to the user.
role_id_list	array	No.	The ID numbers of the roles that are assigned to the user.
limited_assignment_value_list	array	No.	Not documented yet.
is_logged_in	boolean	No.	Returns TRUE if the user is logged in, FALSE otherwise.
is_enabled	boolean	No.	Returns TRUE if the user is enabled, FALSE otherwise.
last_visit	string	No.	Not documented yet.
has_manage_locations	boolean	No.	Not documented yet.

* Indicates whether the attribute provides a pre-fetched value or if it needs to be computed upon request.

5.6.49 ezvatrule

Summary

Contains information about a VAT charging rule.

Attributes

Attribute	Type	Static*	Description
id	integer	Yes.	The ID number of the VAT charging rule.
country	string	Yes.	The name of the country.
vat_type	integer	Yes.	The ID number of the VAT type.
product_categories	array	No.	The product categories affected by this VAT rule as an array of hashes where each hash consists of the following elements: **Attribute:** id **Type:** integer **Description:** The ID number of the product category. **Attribute:** name **Type:** string **Description:** The name of the product category.
product_categories_string	string	No.	The names of the product categories affected by this VAT rule (separated by commas).
product_categories_ids	array	No.	The ID numbers of the product categories affected by this VAT rule.
product_categories_	array	No.	The names

names			of the product categories affected by this VAT rule.
vat_type_object	object	No.	The VAT type (as ezvattype (page 1121) object).
vat_type_name	string	No.	The name of the VAT type.

* Indicates whether the attribute provides a pre-fetched value or if it needs to be computed upon request.

5.6.50   ezvattype

Summary

Contains information about a VAT.

Attributes

Attribute	Type	Static*	Description
id	string	Yes.	The ID number of the VAT type.
name	string	Yes.	The name of the VAT type (for example "Standard").
percentage	string	Yes.	The actual VAT percentage value (without the percent symbol).
is_dynamic	boolean	No.	Returns TRUE if the VAT type is dynamic, FALSE otherwise.

* Indicates whether the attribute provides a pre-fetched value or if it needs to be computed upon request.

5

5.6.51 ezxhtmlxmloutput

Summary

Contains information about XML data.

Attributes

Attribute	Type	Static*	Description
output_text	string	No.	The resulting text which is suitable for the given output format, for example the rendered XHTML text.
view_template_name	string	No.	The name of the template that will be used when the object attribute is viewed. The default is "ezxmltext", but can be overridden for a handler (for example the Online Editor).
aliased_type	string	Yes.	Returns the name of the original handler. This will normally be FALSE (no alias) - the Online Editor takes control of the XML field using an alias.
aliased_handler	string	No.	Returns the original handler if the "aliased_type" attribute is non-false.

* Indicates whether the attribute provides a pre-fetched value or if it needs to be computed upon request.

5.6.52   ezxmlinputhandler

Summary

Contains information about XML data.

Attributes

Attribute	Type	Static*	Description
input_xml	string	Yes.	The text that the end-user has input to the system.
edit_template_name	string	No.	The name of the template that is used when the object attribute is being edited. The default is "ezxmltext", but can be overridden for a custom handler (for example the Online Editor).
information_template_name	string	No.	The name of the template that will be used when the object attribute is collecting information. The default is "ezxmltext", but can be overridden for a custom handler (for example the Online Editor).
aliased_type	string	Yes.	Returns the name of the original handler. This will normally be FALSE (no alias) - the Online Editor takes control of the XML field using an alias.
alias_handler	string	No.	Returns the original handler if the "aliased_type" at-

			tribute is non-false.

* Indicates whether the attribute provides a pre-fetched value or if it needs to be computed upon request.

5

5.6.53   ezxmloutputhandler

Summary

Contains information about XML data.

Attributes

Attribute	Type	Static*	Description
output_text	string	Yes.	The resulting text which is suitable for the given output format, for example the rendered XHTML text.
view_template_name	string	No.	The name of the template that will be used when the object attribute is viewed. The default is "ezxmltext", but can be overridden for a handler (for example the Online Editor).
aliased_type	string	Yes.	Returns the name of the original handler. This will normally be FALSE (no alias) - the Online Editor takes control of the XML field using an alias.
aliased_handler	string	No.	Returns the original handler if the "aliased_type" attribute is non-false.

* Indicates whether the attribute provides a pre-fetched value or if it needs to be computed upon request.

5.6.54 ezxmltext

Summary

Contains information about an XML block.

Attributes

Attribute	Type	Static*	Description
input	object	No.	Returns the current input handler, which will be an object of type ezxmlinputhandler (page 1123), default is ezsimplifiedxmlinput (page 1112).
output	object	No.	Returns the current output handler, which will be an object of type ezxmloutputhandler (page 1125), default is ezhtmlxmloutput (page 1122).
pdf_output	object	No.	Returns the PDF output handler.
xml_data	string	Yes.	Returns the internal XML structure as text.
is_empty	boolean	No.	Returns TRUE if there is no XML data, FALSE otherwise.

* Indicates whether the attribute provides a pre-fetched value or if it needs to be computed upon request.

5

5.7   **Workflow events**

\begin{description}   \item[Approve]\refer{label66f43a379dfcf49d0740d0a6a395afe6}\   \\
Makes it possible to have the contents of objects be approved by an editor.

\item[Multiplexer]\refer{labeldd4556cbc8c8bfa07967e7e22b406b0d}\ \\ Starts other work-
flows from within a workflow.

\item[Payment gateway]\refer{label006b19b4a148204eb6130718326f0bca}\ \\ Generic solu-
tion for handling payment redirections.

\item[Simple shipping]\refer{label837635f7832fc127c0c087df0fc236d8}\ \\ Adds shipping
costs to orders.

\item[Wait until date]\refer{labelbd538856a050ce042ea63de385a147ac}\ \\ Makes it possible
to delay the publishing of objects.

\end{description}

5

5.7.1 Approve

Summary

Makes it possible to have the contents of objects be approved by an editor.

Description

This event makes it possible to have the contents of objects be approved by an editor before they are actually published. A workflow using this event must be connected to the "content/publish/ before" trigger. The following screenshot shows the edit interface for this event.

(see figure 5.84)

Figure 5.84: *Edit interface for the "Approve" event.*

Affected sections

The "Affected sections" menu shows the sections (page 132) that are available on the system. It allows the administrator to filter objects that belong to specific sections. Only objects that belong to the selected sections will be affected by the event.

User who approves content

The "User who approves content" menu shows a list of the user accounts on the system. It allows the administrator to select which user(s) that should function as editors and thus have the power to approve or deny content.

Excluded users and groups

The "Excluded users and groups" menu allows the administrator to select users and user groups that should not be affected by the event. In other words, if one of the selected users or a user who is a member of a selected group publishes an object that normally would be affected by this event, it will not be affected.

This event makes use of the collaboration feature of eZ publish. A user's pending items (items waiting for approval) will be displayed under "Pending items" within the "My account" part of the administration interface. The "Collaboration" interface under "My account" provides the review/ approve/reject interface for the editors.

Please note that the "cronjobs/workflow.php" cronjob must be run periodically for this to work.

5

5.7.2 Multiplexer

Summary

Starts other workflows from within a workflow.

Description

This event makes it possible to spawn other workflows from within a workflow. It is compatible with all of the predefined triggers. The following screenshot shows the edit interface of this event.

(see figure 5.85)
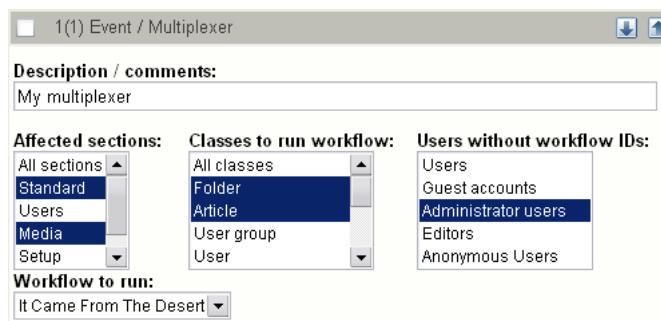
Figure 5.85: *Edit interface for the "Multiplexer" event.*

The "Affected sections" menu makes it possible to isolate the workflow so that it only affects content objects that belongs to the selected section(s). The "Classes to run workflow" allows a similar isolation on the class basis. In other words, only instances of the selected classes will be affected. The "Users without workflow IDs" makes it possible to select user groups that should not be affected by the multiplexer. In other words, if the current user (who initiated the workflow) is within one of the selected groups, the multiplexer event will simply be skipped by the system. The "Workflow to run" menu shows a list of the available workflows and thus allows the selection of the workflow that should be spawned by this event.

5

5.7.3 Payment gateway

Summary

Generic solution for handling payment redirections.

Description

This event is a general solution that is capable of handling different kinds of payment redirections. It allows several payment gateway mechanisms to be plugged into the system through a workflow. The workflow must be assigned to the "shop/checkout/before" trigger. The following screenshot shows the edit interface for this event.

(see figure 5.86)
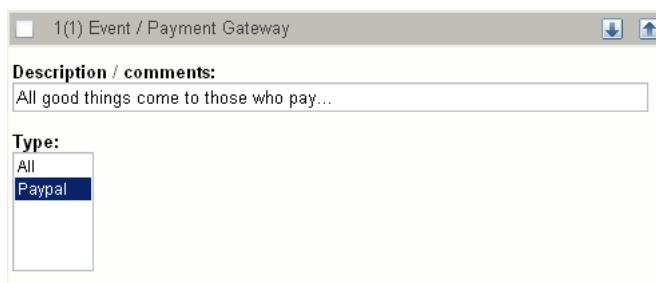
Figure 5.86: *Edit interface for the "Payment gateway" event.*

Type

The "Type" menu shows the payment gateway solutions that are available (installed and activated) as eZ publish extensions. It allows the administrator to select which payment gateway(s) that should be used. If several gateways are selected, the system will allow the customer to choose between the available gateways during the checkout process. The selection interface is presented using "/templates/workflow/selectgateway.tpl" located in the current or one of the fallback designs.

Execution

The following list shows the flow of execution when a payment gateway is used through the interface that this workflow event provides.

1. The customer initiates the checkout process.

2. The browser is redirected to the target payment server (PayPal, PayNet, etc.).

5

3. The customer attempts to pay for the products using the payment gateway.

4. The browser is sent back to eZ publish, the order will be either approved or rejected.

Between steps 3 and 4, the selected payment server will notify eZ publish about the purchase. The system will validate the authenticity (identity) of the message, and based on the reply it will either approve or reject the order. In case of no reply, the validation process will time-out and the customer will be asked to contact the shop owner.

This event provides a development framework which simplifies the implementation of payment solutions.

5

5.7.4 Simple shipping

Summary

Adds shipping costs to orders.

Description

This event adds shipping costs to an order (the order that is being processed for the current user). A workflow using this event must be assigned to the "shop/confirmorder/before" trigger. The following screenshot shows the edit interface for this event.

(see figure 5.87)
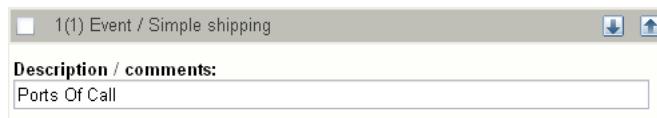
Figure 5.87: *Edit interface for the "Simple shipping" event.*

The cost and the description can be set using the "ShippingCost" and "ShippingDescription" directives within the "[SimpleShippingWorkflow]" group of a configuration override for "workflow.ini".

5

5.7.5   Wait until date

Summary

Makes it possible to delay the publishing of objects.

Description

This event makes it possible to delay the publishing of certain objects. It can be connected to either the "content/publish/before" trigger or the "content/publish/after" trigger. The following screenshot shows the edit interface for this event.

(see figure 5.88)
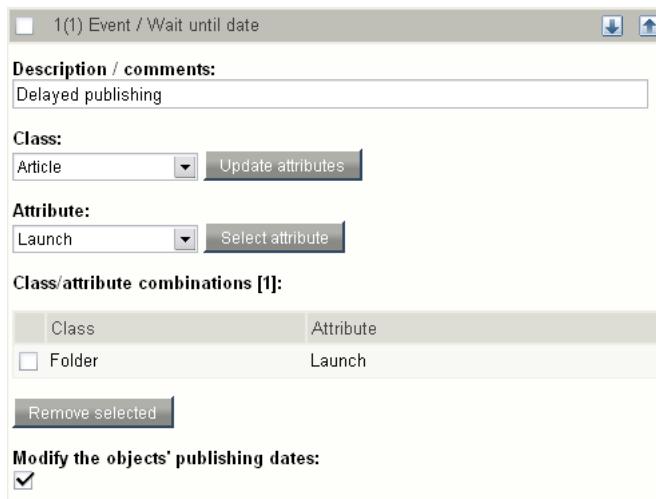
Figure 5.88: *Edit interface for the "Wait until date" event.*

Parameters and usage

The "Class" menu makes it possible to select a class that should be affected by the event. When a class is selected, the "Update attributes" button must be used to update the contents of the "Attribute" menu. It will allow the selection of the attribute which is used to enter the date and time when the object should be published. This attribute must be represented by the "Date and time" (page 524) datatype. Once the correct attribute it selected, the "Select attribute" button must be used to add it to the list below (labeled "Class / attribute combinations"). All objects that are instances of the classes added to this list will be affected by the event. The objects will be published automatically (by the way of the cronjob script) when the given date / time is reached.

If the "Modify the objects' publishing dates" checkbox is checked, the system will update the modification time of the objects when they are published by the system. If unchecked, the date / time when the objects were published by the users will be used.

5.8   Template operators

The template operators are documented in the following sections:

\begin{itemize} \item{} Arrays\refer{labele5f3e60e6a1e9b614f741e8e9b0f2994} \item{} Data and information extraction\refer{labelb11b374e2b9c10966f4d9518d35eb4ec} \item{} Formatting and internationalization\refer{labelf5bf2b0324a576c848c8ac11bdfa4d0e} \item{} Images\refer{label7a98e9205828502bd0d2671060b68ae7} \item{} Logical operations\refer{label36647d9386f86ee7bac22738c71ffea8} \item{} Mathematics\refer{label61b61778db2295265c0740dd26ac600b} \item{} Miscellaneous\refer{label0c368810ac24db56aced5514702270bb} \item{} Strings\refer{label7248a89c254bd9b611dca49d09c5734f} \item{} URLs\refer{label05be3258e931b25477a48b9e8ae6742e} \item{} Variable and type handling\refer{label7546b9dc35aeffbb8d9cc970c194b540} \end{itemize}

5

5.8.1 Arrays

\begin{description} \item[append]\refer{label88380ec5f94826fee084ac055a626fac}\ \\ Returns the input array with appended elements. \item[array]\refer{label1a81c3bc7a44633d1ea738a17cb4a1 \\ Creates and returns a new array. \item[array_sum]\refer{label313d749942c07787eef327540ea5454f}\ \\ Returns the sum of all elements in an array. \item[begins_ with]\refer{labeld2ba21d8c16ca51b7f8301cb392cdaef}\ \\ Checks if an array starts with a specific element/sequence. \item[compare]\refer{labelaa413063efa0c5ee82d817711dea7a92}\ \\ Compares the contents of two arrays. \item[contains]\refer{label5e702100b44d5c0787e832efe99cc7df}\ \\ Checks if an array contains a specific element. \item[ends_ with]\refer{labelc5d503295b8f81ed88b3324856bee45a}\ \\ Checks if an array ends with a specific element or sequence. \item[explode]\refer{labelde1df881ed0effc1c7eb7e9c6fdb1bc1}\ \\ Splits the input array and returns it as an array of sub-arrays. \item[extract]\refer{label61deb5182e347632afc69824b8c30711}\ \\ Returns a portion of the input array. \item[extract_left]\refer{label169c8d7965880cb5d070b9693256e172}\ \\ Returns a portion of the start of the input array. \item[extract_ right]\refer{label3bb2f8c40fada8684b12e2d9f3ba3fd2}\ \\ Returns a portion of the end of the input array. \item[hash]\refer{labelf167cb1c3b0e3b336a837c06ef8f8b55}\ \\ Creates and returns a new associative array (a hash). \item[implode]\refer{label977bdfcbc260ae4614f0938f3e11a96a}\ \\ Joins array elements with strings. \item[insert]\refer{label440ea349f32f4e3ab23d27e55cf7e7e8}\ \\ Inserts an element/sequence at specified position in an array. \item[merge]\refer{labelc375e5ab4a2cd644fe59158399666c8c}\ \\ Merges input and passed arrays into one array. \item[prepend]\refer{label7e13f4ceb0fbc97f92d973be7b9302d4}\ \\ Returns the input array prepended with specified elements. \item[remove]\refer{label6d0195d908c7dbf7b514e50022d20d37}\ \\ Returns the input array without some of the original elements. \item[repeat]\refer{label4ddda5c126539466b286359356d3577f}\ \\ Returns a repeated version of the input array. \item[replace]\refer{labela570c8493daa7d64b3c78ec07cc7c804}\ \\ Replaces elements in an array. \item[reverse]\refer{label0842e9addd50a891f94e7d8662b76fde}\ \\ Returns a reversed version of the input array. \item[unique]\refer{labelf5eb3558157db1599ab841387bc4 \\ Returns the input array without duplicate elements.

\end{description}

5

append

Summary

Returns the input array with appended elements.

Usage

```
input|append( element1 [, element2 [, ... ] ] )
```

Parameters

Name	Type	Description	Required
element1	any	Element to be appended to the input array.	Yes.
element2	any	Another element to be appended to the input array.	No.

Returns

An array consisting of the input array and the parameters.

Description

This operator appends the parameter value(s) at the end of the input array and returns the resulting array.

Examples

Example 1

```
{array( 1, 2, 3 )|append( 4, 5, 6 )}
```

The following array will be returned: (1, 2, 3, 4, 5, 6).

Example 2

```
{array( 1, 2, 3 )|append( array( 4, 5, 6 ))}
```

The following array will be returned: (1, 2, 3, (4, 5, 6)).

5

array

Summary

Creates and returns a new array.

Usage

```
array( element1 [, element2 [, ... ] ] )
```

Parameters

Name	Type	Description	Required
element1	any	Element / value of any kind.	Yes.
element2	any	Another element / value of any kind.	No.

Returns

An array containing the specified elements.

Description

This operator builds an array using the specified elements. The elements must be passed as parameters. The operator returns the resulting array.

Examples

Example 1

```
{array( 1, 2, 3, 4, 5, 6, 7 )}
```

The following array will be returned: (1, 2, 3, 4, 5, 6, 7).

Example 2

```
{array( 1, 2, 3, array( 4, 5, 6 ) )}
```

The following array will be returned: (1, 2, 3, (4, 5, 6)).

array_sum

Summary

Returns the sum of all elements in an array.

Usage

```
input|array_sum()
```

Returns

An integer representing the sum of the elements.

Description

This operator attempts to calculate and return the sum of the input array's elements.

Examples

Example 1

```
{array( 1, 2, 3, 4, 5, 6, 7 )|array_sum}
```

The following output will be produced: "28" - which is the sum of 1, 2, 3, 4, 5, 6, 7.

5

begins_with

Summary

Checks if an array starts with a specific element/sequence.

Usage

```
input|begins_with( element1 [, element2 [, ... ] ] )
```

Parameters

Name	Type	Description	Required
element1	any	An element that should be matched.	Yes.
element2	any	Another element that should be matched.	No.

Returns

TRUE or FALSE.

Description

This operator checks if the input array starts with a specified sequence of elements. If yes, the operator returns TRUE, otherwise FALSE will be returned.

Examples

Example 1

```
{array( 1, 2, 3, 4, 5, 6, 7 )|begins_with( 1, 2, 3 )}
```

Returns TRUE.

Example 2

```
{array( 1, 2, 3, 4, 5, 6, 7 )|begins_with( 2, 3, 4 )}
```

Returns FALSE.

compare

Summary

Compares the contents of two arrays.

Usage

```
input|compare( array )
```

Parameters

Name	Type	Description	Required
array	array	The array that should be compared with the input array.	Yes.

Returns

TRUE if arrays are equal, FALSE if not.

Description

This operator compares the contents of two arrays, the input array and an array that is provided as the first (and only) parameter. If the arrays are equal, the operator returns TRUE, otherwise FALSE will be returned.

Examples

Example 1

```
{array( 1, 2, 3, 4, 5 )|compare( array( 1, 2, 3, 4, 5 ))}
```

Returns TRUE.

Example 2

```
{array( 1, 2, 3, 4, 5 )|compare( array( 5, 4, 3, 2, 1 ))}
```

Returns TRUE.

5

Example 3

```
{array( 1, 2, 3, 4, 5 )|compare( array( 1, 2, 4, 3, 3 ))}
```

Returns FALSE.

5

contains

Summary

Checks if an array contains a specific element.

Usage

```
input|contains( element )
```

Parameters

Name	Type	Description	Required
element	any	The element that should be matched.	Yes.

Returns

TRUE if the element is found, FALSE if not.

Description

This operator checks if the input array contains a specific element (specified using the first parameter). If it does, the operator will return TRUE, otherwise FALSE will be returned.

Examples

Example 1

```
{array( 1, 2, 3, 4, 5 )|contains( 3 )}
```

Returns TRUE.

Example 2

```
{array( 1, array( 3, 4 ), 5 )|contains( array( 3, 4 ) )}
```

Returns TRUE.

Example 3

5

```
{array( 1, array( 3, 4 ), 5 )|contains( 3 )}
```

Returns FALSE.

5

ends_with

Summary

Checks if an array ends with a specific element or sequence.

Usage

```
input|ends_with( element1 [, element2 [, ... ] ] )
```

Parameters

Name	Type	Description	Required
element1	any	An element that should be matched.	Yes.
element2	any	Another element that should be matched.	No.

Returns

TRUE or FALSE.

Description

This operator checks if the input array ends with a specified sequence of elements. If yes, the operator returns TRUE, otherwise FALSE will be returned.

Examples

Example 1

```
{array( 1, 2, 3, 4, 5, 6, 7 )|ends_with( 5, 6, 7 )}
```

Returns TRUE.

Example 2

```
{array( 1, 2, 3, 4, 5, 6, 7 )|ends_with( 4, 5, 6 )}
```

Returns FALSE.

explode

Summary

Splits the input array and returns it as an array of sub-arrays.

Usage

```
input|explode( offset )
```

Parameters

Name	Type	Description	Required
offset	integer	The offset where the array should be split.	Yes.

Returns

An array containing the original array as two arrays.

Description

This operator splits the input array at an offset specified by the "offset" parameter. The operator will return an array containing the two arrays.

Examples

Example 1

```
{array( 1, 2, 3, 4, 5 )|explode( 3 )}
```

The following array will be returned: ((1, 2, 3), (4, 5)).

5

extract

Summary

Returns a portion of the input array.

Usage

```
input|extract( offset [, length ] )
```

Parameters

Name	Type	Description	Required
offset	integer	The offset to start at.	Yes.
length	integer	The number of elements that should be extracted.	No.

Returns

An array containing the extracted elements.

Description

This operator will return a portion of the input array. The desired portion must be defined by the "offset" and "length" parameters. If the "length" parameter is omitted, the rest of the array (from offset) will be returned.

Examples

Example 1

```
{array( 1, 2, 3, 4, 5, 6, 7 )|extract( 2 )}
```

The following array will be returned: (3, 4, 5, 6, 7).

Example 2

```
{array( 1, 2, 3, 4, 5, 6, 7 )|extract( 3, 3 )}
```

The following array will be returned: (4, 5, 6).

5

extract_left

Summary

Returns a portion of the start of the input array.

Usage

```
input|extract_left( length )
```

Parameters

Name	Type	Description	Required
length	integer	The number of elements that should be extracted.	Yes.

Returns

An array containig the extracted elements.

Description

This operator extracts a portion from the start of the input array. The "length" parameter must be used to specify the number of elements that should be extracted.

Examples

Example 1

```
{array( 1, 2, 3, 4, 5, 6, 7 )|extract_left( 3 )}
```

The following array will be returned: (1, 2, 3).

5

extract_right

Summary

Returns a portion of the end of the input array.

Usage

```
input|extract_right( length )
```

Parameters

Name	Type	Description	Required
length	integer	The number of elements that should be extracted.	Yes.

Returns

An array containing the extracted elements.

Description

This operator extracts a portion from the end of the input array. The "length" parameter must be used to specify the number of elements that should be extracted.

Examples

Example 1

```
{array( 1, 2, 3, 4, 5, 6, 7 )|extract_right( 3 )}
```

The following array will be returned: (5, 6, 7).

5

hash

Summary

Creates and returns a new associative array (a hash).

Usage

```
hash( key1, value1 [, key2, value2 [, ... ] ] )
```

Parameters

Name	Type	Description	Required
key1	string	The key of value1.	Yes.
value1	any	The value associated with key1.	Yes.
key2	string	The key of value2.	No.
value2	any	The value associated with key2.	No.

Returns

An associative array (a hash).

Description

This operator builds an associative array using the specified key/value pairs. Odd parameters are considered to be keys, even parameters will be values. The operator returns the generated hash.

Examples

Example 1

```
{hash( 1, 'Red Eyes', 2, 'Green Gremlins', 3, 'Blue Thunder' )}
```

The following hash will be returned:

Key	Value
1	Red Eyes
2	Green Gremlins
3	Blue Thunder

5

implode

Summary

Joins array elements with strings.

Usage

```
input|implode( separator )
```

Parameters

Name	Type	Description	Required
separator	string	The string that should be inserted between the elements.	Yes.

Returns

String containing array elements separated a string.

Description

This operator returns a string representation of the elements of the input array. Each element will be separated by the string specified using the "separator" parameter.

Examples

Example 1

```
{array( 1, 2, 3, 4, 5, 6, 7 )|implode( ', ' )}
```

The following string will be returned: "1, 2, 3, 4, 5, 6, 7".

Example 2

```
{array( 1, 2, 3, 4, 5, 6, 7 )|implode( '_-_' )}
```

The following string will be returned: "1 - 2 - 3 - 4 - 5 - 6 - 7".

insert

Summary

Inserts an element/sequence at specified position in an array.

Usage

```
input|insert( offset, element1 [, element2 [, ... ] ] )
```

Parameters

Name	Type	Description	Required
offset	integer	The offset where the element(s) should be inserted at.	Yes.
element1	any	An element that should be inserted into the existing array.	Yes.
element2	any	Another element that should be inserted into the existing array.	No.

Returns

An array containing a combination of the original array and the inserted elements.

Description

This operator inserts an element or a sequence of elements at a specified position within the input. The resulting array will be returned (original array with the inserted values).

Examples

Example 1

```
{array( 1, 2, 5 )|insert( 2, 3, 4 )}
```

The following array will be returned: (1, 2, 3, 4, 5).

5

merge

Summary

Merges input and passed arrays into one array.

Usage

```
input|merge( array1 [, array2 [, ... ] ] )
```

Parameters

Name	Type	Description	Required
array1	array	Array to be merged with the input array.	Yes.
array2	array	Another array to be merge with the input array.	No.

Returns

New array containing all arrays merged.

Description

This operator will merge the input array with all arrays passed as parameters. The resulting array will be returned.

Examples

Example 1

```
{array( 1, 2 )|merge( array( 3, 4 ), array( 5, 6, 7 ) )}
```

The following array will be returned: (1, 2, 3, 4, 5, 6, 7).

5

prepend

Summary

Returns the input array prepended with specified elements.

Usage

```
input|prepend( element1 [, element2 [, ... ] ] )
```

Parameters

Name	Type	Description	Required
element1	any	Element to be prepended to the input array.	Yes.
element2	any	Another element to be prepended to the input array.	No.

Returns

An array consisting of the parameters and the input array.

Description

This operator adds the parameter value(s) to the start of the input array and returns the resulting array.

Examples

Example 1

```
{array( 4, 5 )|prepend( 1, 2, 3 )}
```

The following array will be returned: (1, 2, 3, 4, 5).

5

remove

Summary

Returns the input array without some of the original elements.

Usage

```
input|remove( offset [, length ] )
```

Parameters

Name	Type	Description	Required
offset	integer	The offset to start removing elements.	Yes.
length	integer	The number of elements that should be removed.	No.

Returns

A cut-down version of the input array.

Description

This operator removes element(s) from the input array and thus a chopped/cut-down version of the input array will be returned. The "offset" parameter must be used to define the start of the portion that should be removed. The "length" parameter can be used to define the number of elements that should be removed. If the "length" parameter is omitted, only one elemen (specified by offset will be removed).

Examples

Example 1

```
{array( 1, 2, 3, 4, 5 )|remove( 2, 2 )}
```

The following array will be returned: (1, 2, 5).

Example 2

```
{array( 1, 2, 3, 4, 5 )|remove( 2 )}
```

The following array will be returned: (1, 2, 4, 5).

5

repeat

Summary

Returns a repeated version of the input array.

Usage

```
input|repeat( times )
```

Parameters

Name	Type	Description	Required
times	integer	The number of times the array should be repeated.	Yes.

Returns

An array containing the elements of the input array repeated n times.

Description

This operator returns a repeated version of the input array. The "times" parameter must be used to define the number of times the array should be repeated.

Examples

Example 1

```
{array( 1, 2, 3, 4 )|repeat( 3 )}
```

The following array will be returned: (1, 2, 3, 4, 1, 2, 3, 4, 1, 2, 3, 4).

5

replace

Summary

Replaces elements in an array.

Usage

```
input|replace( offset [, length [, element1 [, element2 [, ...]]] ] )
```

Parameters

Name	Type	Description	Required
offset	integer	The offset to start replacing elements.	Yes
length	integer	The number of elements that should be replaced.	No
element1	any	The new element that will replace the old one.	No
element2	any	The new element that will replace the old one.	No

Returns

A replaced version of the input array.

Description

This operator replaces elements(s) from the input array and a replaced version of input array will be returned. The "offset" parameter must be used to define the start of the portion that should be replaced. The "length" parameter can be used to define the number of elements that should be replaced. If length is omitted, the element(s) will be placed after the offset(like 'insert' operator).

Examples

Example 1

```
{array( 1, 2, 3, 4, 5 )|replace( 1, 2 )}
```

The following array will be returned: (1, 4, 5).

Example 2

```
{array( 1, 2, 3, 4, 5 )|replace( 3 )}
```

The following array will be returned: (1, 2, 3, 4, 5).

Example 3

```
{array( 1, 2, 3, 4, 5 )|replace( 1, 2, 'a', 'b', 'c' )}
```

The following array will be returned: (1, 'a', 'b', 'c', 4, 5).

Example 4

```
{array( 1, 2, 3, 4, 5 )|replace( 1, , 'a', 'b', 'c' )}
```

The following array will be returned: (1, 'a', 'b', 'c', 2, 3, 4, 5).

5

reverse

Summary

Returns a reversed version of the input array.

Usage

```
input|reverse()
```

Returns

A reversed version of the input array.

Description

This operator returns a reversed version (elements in reverse order) of the input array.

Examples

Example 1

```
{array( 1, 2, 3, 4 )|reverse}
```

The following array will be returned: (4, 3, 2, 1).

5

unique

Summary

Returns the input array without duplicate elements.

Usage

```
input|unique()
```

Returns

The input array containing only one occurance of every element.

Description

This operator removes duplicate elements from the input array.

Examples

Example 1

```
{array( 1, 2, 2, 3, 4, 4, 5 )|unique}
```

The following array will be returned: (1, 2, 3, 4, 5).

5

5.8.2 Data and information extraction

\begin{description} \item[currentdate]\refer{label71244e7a9ce1fcc226898f58e0648e3c}\ \\
Returns the timestamp of the current date/time. \item[ezhttp]\refer{label79ecebfefb0a5646d77466f9790d4
\\ Returns GET, POST, session and cookie variables. \item[ezhttp_
hasvariable]\refer{label512238517789fcffe2b3812ec4e11890}\ \\ Returns TRUE if the speci-
fied HTTP variable is set.   \item[ezini]\refer{label22878f055246b690715362fee49f34b7}\
\\ Provides read access to settings in the configuration files.
\item[ezini_hasvariable]\refer{labeldff9d3e7cb148a13d0f608174270ebd9}\
\\ Returns TRUE if the specified configuration directive is
set.   \item[ezmodule]\refer{labelbd9eb479669fff8c4d2cc80a0d0a00e8}\
\\ Returns TRUE if the specified module or view is enabled.
\item[ezpreference]\refer{label9ebeb42161bc68bc1f15ba384a2c16d6}\ \\ Provides access to a
user's preference values. \item[ezsys]\refer{labelc19f70bc99da31f9eb157ba2cf8498b9}\ \\ Re-
turns misc values such as wwwdir, sitedir, etc. \item[fetch]\refer{label1e0d1ef810f783802133c34f15a8454
\\ Provides access to the fetch functions of a module.   \item[module_
params]\refer{label71a9dd9ce3b4da0c9e68737a40be3b2f}\ \\ Extracts parameters from
the module that was run.

\end{description}

5

currentdate

Summary

Returns the timestamp of the current date/time.

Usage

```
currentdate()
```

Returns

The current date/time as a UNIX timestamp.

Description

This operator returns the UNIX timestamp for the current date/time.

Examples

Example 1

```
{def $timestamp=currentdate()}
Current timestamp: {$timestamp}
```

Outputs the current timestamp.

Example 2

```
{def $timestamp=currentdate()}
Current date/time: {$timestamp|l10n( 'shortdatetime' )}
```

Outputs the current date/time in a user friendly format.

5

ezhttp

Summary

Returns GET, POST, session and cookie variables.

Usage

```
ezhttp( name [, type [, check_existence ] ] )
```

Parameters

Name	Type	Description	Required
name	string	The HTTP variable that should be fetched.	Yes.
type	string	Source of variable. Default is is HTTP POST.	No.
check_existence	mixed	If the specified directive should be checked for existence.	No.

Returns

Variable value

Description

This operator makes it possible to inspect the contents of HTTP variables (POST, GET, session and cookie variables). The "type" parameter can be used to specify which type of variable that should be extracted. The following options are available:

- post (POST variable)

- get (GET variable)

- session (session variable)

- cookie (cookie variable)

The "check_existence" parameter can be set to TRUE or the string "hasVariable" in order to check if a directive is actually set. In this case, the operator will return either TRUE or FALSE. Note that using the "ezhttp_hasvariable" operator is a better option. (page 1165)

Examples

Example 1

5

```
{ezhttp( 'search' )}
```

Returns the "search" POST variable.

Example 2

```
{ezhttp( 'image', 'get' )}
```

Returns the "image" GET variable.

Example 3

```
{ezhttp( 'user_id', 'session' )}
```

Returns the "user_id" session variable.

Example 4

```
ezhttp( 'is_logged_in', 'cookie' )
```

Returns the 'is_logged_in' cookie variable.

5

ezhttp_hasvariable

Summary

Returns TRUE if the specified HTTP variable is set.

Usage

```
ezhttp_hasvariable( variable [, method ] )
```

Parameters

Name	Type	Description	Required
variable	string	The name of the variable that should be checked.	Yes.
method	string	Source of varialbe, default is HTTP POST.	No.

Returns

TRUE if the specified HTTP variable is set, FALSE otherwise.

Description

This operator makes it possible to check if an HTTP variable (GET, POST or session) specified using the "variable" parameter is set. The "method" parameter can be used to specify which type of variables the system should check (default is POST variables); the following options are available:

- post (POST variable)

- get (GET variable)

- session (session variable)

Examples

Example 1

```
{ezhttp_hasvariable( 'user_id', 'session' )}
```

Returns TRUE if the "user_id" session variable is set.

5

ezini

Summary

Provides read access to settings in the configuration files.

Usage

```
ezini( section, variable [, ini_file [, ini_path [, dynamic [, check_existence
] ] ] ] )
```

Parameters

Name	Type	Description	Required
section	string	Section to read value from.	Yes.
variable	string	The name of the directive that should be accessed.	Yes.
ini file	string	The target configuration file (default is "site.ini").	No.
ini_path	string	Custom path to the configuration file(s).	No.
dynamic	boolean	If the specified directive should be dynamically checked.	No.
check_existence	mixed	If the specified directive should be checked for existence.	No.

Returns

A string containing the value of a configuration setting.

Description

This operator makes it possible to access the settings in the configuration ("*.ini") files. The "ini_path" operator can be used to set a custom path where the configuration files are located (default is the settings directory). The "dynamic" parameter can be used to force the system to actually check the ini file, otherwise the value will be checked upon template compilation and "hardcoded" in the compiled template. The "check_existence" parameter can be set to TRUE or the string "hasVariable" in order to check if a directive is actually set. In this case, the operator will return either TRUE or FALSE. Note that using the "ezini_hasvariable" (page 1168) operator is a better option.

5

Examples

Example 1

```
{if eq(ezini( 'SomeSettings', 'Test', 'example.ini' ), 'hello' )}
    [...display something...]
{else}
    [...display something else...]
{/if}
```

Conditional branching based on an configuration setting.

5

ezini_hasvariable

Summary

Returns TRUE if the specified configuration directive is set.

Usage

```
ezini_hasvariable( section, variable [, ini_file [, ini_path [, dynamic ] ] ] )
```

Parameters

Name	Type	Description	Required
section	string	Section to read value from.	Yes.
variable	string	The name of the directive that should be accessed.	Yes.
ini_file	string	The target configuration file (default is "site.ini").	No.
ini_path	string	Custom path to the configuration file(s).	No.
dynamic	boolean	If the specified directive should be dynamically checked.	No.

Returns

TRUE if the specified configuration directive is set, FALSE otherwise.

Description

This operator can be used to check if a configuration directive is set (or if it is empty). The "section", "variable" and "ini_file" parameters should be used to pinpoint the directive to be checked. If the directive is not set, the operator will return FALSE, otherwise it will return TRUE. The "ini_path" variable can be used to set a custom path. The "dynamic" parameter can be used to force the system to actually check the ini file, otherwise the value will be checked only upon template compilation and "hardcoded" in the compiled templates.

Examples

Example 1

```
{ezini_hasvariable( 'VersionManagement', 'DeleteDrafts', 'content.ini' )}
```

Returns TRUE if "DeleteDrafts" has been set to something within the "[VersionManagement]" block of "content.ini" or an override; otherwise FALSE is returned.

ezmodule

Summary

Returns TRUE if the specified module or view is enabled.

Usage

```
ezmodule( name )
```

Parameters

Name	Type	Description	Required
name	string	The name of the module or view that should be checked (for example "ezinfo" or "content/search").	Yes.

Returns

TRUE if the specified module or view is enabled, FALSE otherwise.

Description

This operator checks if the specified module or view is enabled according to the "Rules (page 2485)" configuration array located in the [SiteAccessRules] section of "site.ini" or an override. Note that this operator can not be used to check whether the current user has access to a certain function of the specified module (this can only be done using the "has_access_to (page 978)" fetch function of the "user" module).

Examples

Example 1

```
{def $isShopEnabled = ezmodule( 'shop' )}
{if $isShopEnabled}
    The shop module is enabled.
{else}
    The shop module is disabled.
{/if}
```

Outputs information about whether the "shop" module is enabled or disabled.

Example 2

```
{ezmodule( 'content/tipafriend' )}
```

Returns TRUE if "tipafriend" view of the "content" module is enabled, FALSE otherwise.

5

ezpreference

Summary

Provides access to a user's preference values.

Usage

```
ezpreference( preference )
```

Parameters

Name	Type	Description	Required
preference	string	The name of the preference that should be extracted.	Yes.

Returns

A string containing the contents/value of the specified preference.

Description

This operator makes it possible to extract the preference values of the current user. The name of the desired preference must be provided as a parameter. The function returns the value/contents of the specified preference. The current user can set a preference by requesting a URL of the following type:

```
/user/preferences/set/[name_of_preference]/[value]
```

Examples

Example 1

```
{ezpreference( 'bookmark_menu' )}
```

Returns the value/contents of the "bookmark_menu" preference for the current user.

5

ezsys

Summary

Returns misc values such as wwwdir, sitedir, etc.

Usage

```
ezsys( system_variable )
```

Parameters

Name	Type	Description	Required
system_variable	string	The name of the desired variable.	Yes.

Returns

A string containing the requested variable.

Description

This operator gives read access to certain eZ Publish system variables. The "system_variable" parameter must be used to specify the name of the variable that should be returned. As of eZ Publish 4.6 the "ezsys()" template operator now has a new parameter available "querystring". The following names can be used:

- indexfile - name of index file

- indexdir - relative path and index file

- sitedir - local path of eZ Publish installation

- wwwdir - relative directory path of eZ Publish installation

- hostname - hostname of eZ Publish server

- querystring (as of 4.6)

Examples

Example 1

Example: news theme installed on server "ez.no", by user "tom" in "/home/tom/public_html/local/ez_publish"

```
wwwdir - {ezsys( 'wwwdir' )}<br />
sitedir - {ezsys( 'sitedir' )}<br />
indexfile - {ezsys( 'indexfile' )}<br />
indexdir - {ezsys( 'indexdir' )}<br />
magicQuotes - {ezsys( 'magicQuotes' )}<br />
hostname - {ezsys( 'hostname' )}<br />
querystring - {ezsys('querystring')}<br/>
```

The following output will be produced:

```
wwwdir - /~tom/local/ez_publish
 sitedir - /home/tom/public_html/local/ez_publish/
 indexfile - /index.php/news
 indexdir - /~tom/local/ez_publish/index.php/news
 magicQuotes -
 hostname - ez.no
querystring - ?param1=value1&param2=value2
```

5

fetch

Summary

Provides access to the fetch functions of a module.

Usage

```
fetch( ... )
```

Parameters

Name	Type	Description	Required
any	any	Please refer to the links below.	Yes.

Returns

The returned result depends on the actual fetch operation.

Description

The fetch operator provides access to the fetch functions of an eZ Publish module. Please refer to the "Information extraction" (page 221) section of the "Templates" chapter for more information about the operator itself. The actual fetch functions are documented under the "Template fetch functions" (page 1472) section of the "Reference" chapter.

5

module_params

Summary

Extracts parameters from the module that was run.

Usage

```
module_params()
```

Returns

An array containing module information.

Description

This operator extracts some generic information from the module that was run. It seems that it can only be called inside "pagelayout.tpl". Please refer to the example below. The operator does not take any parameters.

Examples

Example 1

```
{module_params()|attribute(show)}
```

If the requested URL is "/content/view/full/65" (or using URL alias that points to "/content/view/full/65"), the following output will be produced:

Attribute	Type	Value
module_name	string	'content'
function_name	string	'view'
parameters	array	Array(2)
-ViewMode	string	'full'
-NodeID	string	65

Please note that this operator can only be called in "pagelayout.tpl".

5.8.3 Formatting and internationalization

\begin{description} \item[d18n]\refer{label1310a5ac12b8d0b6dae9b62226f83e62}\ \\ Marks a string for translation. Valid from 4.3 \item[datetime]\refer{labelbdb054d799f602b8ccfda2326cb0cea4}\ \\ Formats dates/times according to settings in "datetime.ini". \item[i18n]\refer{label59b5bc3603dbcc5a1eabe5cd65ac97cd}\ \\ Marks a string for translation. \item[l10n]\refer{labelb22cd59075ce4b5420870be566c1efc4}\ \\ Formats misc. numbers (times, dates, currencies, numbers, etc.). \item[si]\refer{labelb96bec535ff3e7ae0b1895613370f44c}\ \\ Handles unit display of values (output formatting).

\end{description}

5

d18n

Summary

Marks a string for translation. Valid from 4.3

Usage

```
input|d18n( context [, comment [, arguments ] ] )
```

Parameters

Name	Type	Description	Required
context	string	The context to which the string belongs.	Yes.
comment	string	A comment describing the text.	No.
arguments	hash	An associative array of arguments in the input parameter.	No.

Returns

A string containing a translated version of the input parameter.

Description

This adds a new template operator "d18n", it's an alias for i18n() operator and behaves just like it, only difference is that ezlupdate (qt scripts that generates translation files) will ignore it so it can be used for dynamic strings (variables). This operator makes it possible to translate static strings that are defined in various templates. It is typically useful to ensure that the HTML interface is available in several languages in a multi-language scenario.

The operator takes one required parameter "context", and two optional parameters: "comment" and "arguments". The "context" parameter should be used to specify a group to which the input parameter should be related. This is typically useful when there are a lot of strings that need to be structured/grouped. The "comment" parameter makes it possible to add additional comment which can help the person responsible for doing the actual translation. A comment could for example be "Button label" - revealing what that mysterious string actually is. The "arguments" parameter makes it possible to mix dynamic text into the translations. Please refer to "Example 2" for a demonstration of this feature.

Note: This template operator was added from version 4.3 onwards.

5

Examples

Example 1

```
{"Are you sure you want to remove these items?"|d18n('design/standard/node')}
```

Outputs "Are you sure you want to remove these items?" translated to the current language. The translation is taken from the context block named "design/standard/node" located in the appropriate translation file.

For example, let's say that the "translation.ts" file located in the "share/translations/ita-IT/" directory of your eZ Publish installation contains the following lines:

```
<context>
<name>design/standard/node</name>
 ...
<message>
<source>Are you sure you want to remove these items?</source>
<translation>Sei sicuro di voler rimuovere questi elementi?</translation>
</message>
 ...
</context>
```

If your current system locale is "ita-IT" (as specified in the " Locale (page 2434)" setting located in the "[RegionalSettings]" section of the "settings/site.ini" configuration file or its override) then the following output is produced: "Sei sicuro di voler rimuovere questi elementi?"

Example 2

```
{def $number=5}
{"Please enter %number characters."|d18n('design/standard/
node', '', hash( '%number', $number ) )}
```

Outputs "Please enter 5 characters.", the %number will be dynamically replaced by the variable.

5

datetime

Summary

Formats dates/times according to settings in "datetime.ini".

Usage

```
{input|datetime( preset_format [, format ] )}
```

Parameters

Name	Type	Description	Required
preset_format	string	Preset datetime format set in "datetime.ini".	Yes.
format	string	Custom format (when preset format is set to "custom").	No.

Returns

A string representation of the input parameter.

Description

This operator takes care of formatting dates and times according to the setting defined in "datetime.ini" (or a configuration override). In addition, the operator also allows custom formats when the "preset_format" parameter is set to "custom". A custom format must be specified using the "format" parameter. The following table reveals the different elements that can be used in a custom format.

Element	Output	Description
%a	am	Lowercase Ante meridiem and Post meridiem.
%A	AM	Uppercase Ante meridiem and Post meridiem.
%c	2004-02-12T15:19:21+00:00	ISO 8601 date
%d	08	Day of the month, 2 digits with leading zeros.
%D	Wed	A short textual representation of a day, in accordance with the "[ShortDayNames]" section of the language .INI file located in the "share/ locale" directory.
%F	October	A full textual representation

		of a month, such as January or March.
%g	12	12-hour format of an hour without leading zeros.
%G	0	24-hour format of an hour without leading zeros.
%h	12	12-hour format of an hour with leading zeros.
%H	00	24-hour format of an hour with leading zeros.
%i	00	Minutes with leading zeros
%j	8	Day of the month without leading zeros
%l	Wednesday	A full textual representation of the day of the week.
%m	10	Numeric representation of a month, with leading zeros.
%M	Oct	A short textual representation of a month, in accordance with the "[ShortMonthNames]" section of the language .INI file located in the "share/locale" directory.
%n	10	Numeric representation of a month, without leading zeros.
%O	-0500	Difference to Greenwich time (GMT) in hours.
%r	Thu, 21 Dec 2000 16:01:07 +0200	RFC 2822 formatted date
%s	00	Seconds, with leading zeros.
%T	CDT	Timezone setting of this machine.
%U	1065589200	Seconds since the Unix Epoch (January 1 1970 00:00:00 GMT).
%w	3	Numeric representation of the day of the week.
%W	41	ISO-8601 week number of year, weeks starting on Monday.
%Y	2003	A full numeric representation of a year, 4 digits.
%y	03	A two digit representation of

		a year.
%z	280	The day of the year.
%Z	-18000	Timezone offset in seconds. The offset for timezones west of UTC is always negative, and for those east of UTC is always positive.

The date used to generate the contents of this table was "12:00 AM (Midnight) CDT on October 8, 2003".

Examples

Example 1

```
{currentdate()|datetime( 'mydate' )}
```

The following output will be produced: "13:15 6 Feb 2004" (according to the configuration settings).

Example 2

```
{currentdate()|datetime( 'custom', '%h:%i %a %d %F %Y' )}
```

The following output will be produced: "01:15 pm 06 February 2004" (custom format).

5

i18n

Summary

Marks a string for translation.

Usage

```
input|i18n( context [, comment [, arguments ] ] )
```

Parameters

Name	Type	Description	Required
context	string	The context to which the string belongs.	Yes.
comment	string	A comment describing the text.	No.
arguments	hash	An associative array of arguments in the input parameter.	No.

Returns

A string containing a translated version of the input parameter.

Description

This operator makes it possible to translate static strings that are defined in various templates. It is typically useful to ensure that the HTML interface is available in several languages in a multilanguage scenario.

The operator takes one required parameter "context", and two optional parameters: "comment" and "arguments". The "context" parameter should be used to specify a group to which the input parameter should be related. This is typically useful when there are a lot of strings that need to be structured/grouped. The "comment" parameter makes it possible to add additional comment which can help the person responsible for doing the actual translation. A comment could for example be "Button label" - revealing what that mysterious string actually is. The "arguments" parameter makes it possible to mix dynamic text into the translations. Please refer to "Example 2" for a demonstration of this feature.

Examples

Example 1

```
{"Are you sure you want to remove these items?"|i18n('design/standard/node')}
```

Outputs "Are you sure you want to remove these items?" translated to the current language. The translation is taken from the context block named "design/standard/node" located in the appropriate translation file.

For example, let's say that the "translation.ts" file located in the "share/translations/ita-IT/" directory of your eZ Publish installation contains the following lines:

```
<context>
<name>design/standard/node</name>
 ...
<message>
<source>Are you sure you want to remove these items?</source>
<translation>Sei sicuro di voler rimuovere questi elementi?</translation>
</message>
 ...
</context>
```

If your current system locale is "ita-IT" (as specified in the " Locale (page 2434)" setting located in the "[RegionalSettings]" section of the "settings/site.ini" configuration file or its override) then the following output is produced: "Sei sicuro di voler rimuovere questi elementi?"

Example 2

```
{def $number=5}
{"Please enter %number characters."|i18n('design/standard/
node', '', hash( '%number', $number ) )}
```

Outputs "Please enter 5 characters.", the %number will be dynamically replaced by the variable.

5

l10n

Summary

Formats misc. numbers (times, dates, currencies, numbers, etc.).

Usage

```
input|l10n( type [, locale [, symbol ] ] )
```

Parameters

Name	Type	Description	Required
type	string	The format that should be used.	Yes.
locale	string	The locale that should be used (when type is set to "currency").	No.
symbol	string	The currency symbol that should be used (when type is set to "currency").	No.

Returns

A string containing a formatted version of the input parameter.

Description

This operator formats/localizes miscellaneous numeric values according to the current locale settings. The value that should be formatted must be input using the input parameter. The "type" parameter must be used to select the desired format. The following list reveals the available formats/types.

- time
- shorttime
- date
- shortdate
- datetime
- shortdatetime
- currency
- clean_currency
- number

In addition, the operator also formats numeric values according to the specified "locale" and "symbol" parameters when the "type" parameter is set to "currency".

Examples

Example 1

```
{def $number=1234.567
     $timestamp=currentdate()}


time:          {$timestamp|l10n( 'time' )}
shorttime:     {$timestamp|l10n( 'shorttime' )}
date:          {$timestamp|l10n( 'date' )}
shortdate:     {$timestamp|l10n( 'shortdate' )}
datetime:      {$timestamp|l10n( 'datetime' )}
shortdatetime: {$timestamp|l10n( 'shortdatetime' )}
currency:      {$number|l10n( 'currency' )}
clean_currency: {$number|l10n( 'clean_currency' )}
number:        {$number|l10n( 'number' )}
```

If the current locale is "eng-GB", the following output will be produced:

```
time : 1:46:05 pm
shorttime : 1:46 pm
date : Friday 06 February 2004
shortdate : 06/02/2004
datetime : Friday 06 February 2004 1:46:05 pm
shortdatetime : 06/02/2004 1:46 pm
currency : £ 1,234.57
clean_currency : 1,234.57
number : 1,234.57
```

Example 2

```
{def $price=1234.57}
The price in local currency : {$price|l10n( 'currency' )} <br />
The price in Norwegian Krone : {$price|l10n( 'currency', 'nor-NO' )} <br />
The price in Norwegian Krone with specified symbol : {$price|l10n( 'currency',
'nor-NO', 'nok' )}
```

If the current locale is "eng-GB", the following output will be produced:

5

```
The price in local currency : £ 1,234.57
The price in Norwegian Krone : kr 1.234,57
The price in Norwegian Krone with specified symbol : nok 1.234,57
```

5

si

Summary

Handles unit display of values (output formatting).

Usage

```
input|si( unit [, prefix ] [, decimals ] [, symbol ] [, separator ] )
```

Parameters

Name	Type	Description	Required
unit	string	The unit that the input value should be converted to.	Yes.
prefix	string	The prefix used to represent the input value.	No.
decimals	integer	The number of decimal digits that should be shown.	No.
symbol	string	The symbol that should be used as the decimal separator.	No.
separator	string	The symbol that should be used as the thousand separator.	No.

Returns

The input value formatted according to the given parameters.

Description

This operator handles formatting of different kind of values (file sizes, lengths, weights, etc.). The value that should be formatted must be provided as the input parameter. The first parameter must reveal the type of the input value. The following list shows the types that can be used.

- meter

- gram

- second

- ampere

- kelvin

- mole

- candela

5

- byte

- bit

The second parameter is optional and can be used to specify a desired prefix. The rest of the parameters determine how the value should be formatted when it comes to decimals, separators, etc. If these parameters are omitted, the operator will use the settings of the current locale.

Custom units

Custom units can be configured by extending the entries of the "Base" group in a configuration override for "units.ini".

Prefix tables

The prefix is either the name of the size like *kilo* or one of these

Prefix	Description
binary	Calculate using 2 as base, e.g. 2^8
decimal	Calculate using 10 as base, e.g. 10^6
none	Show value as it is with just the unit appended
auto	Determine base from the type of unit (controlled by the "BinaryUnits" setting, default is "bit" and "byte").

The following table shows the proper binary prefixes.

Prefix	Power of 2	Symbol
kibi	2^10	Ki
mebi	2^20	Mi
gibi	2^30	Gi
tebi	2^40	Ti
pebi	2^50	Pi
exbi	2^60	Ei

The following table shows the commonly used binary prefixes (please note that they may be inaccurate).

Prefix	Power of 2	Symbol
kilo	2^10	k
mega	2^20	M
giga	2^30	G
tera	2^40	T
peta	2^50	P
exa	2^60	E

The following table shows the decimal prefixes.

Prefix	Power of 10	Symbol
yotta	10^24	Y
zetta	10^21	Z
exa	10^18	E
peta	10^15	P
tera	10^12	T
giga	10^9	G
mega	10^6	M
kilo	10^3	k
hecto	10^2	h
deca	10^1	da
deci	10^-1	d
centi	10^-2	c
milli	10^-3	m
micro	10^-6	
nano	10^-9	n
pico	10^-12	p
femto	10^-15	f
atto	10^-18	a
zepto	10^-21	z
yocto	10^-24	y

All of these values are defined in the "units.ini" configuration file.

Examples

Example 1

```
{1025|si( byte )}
{1025|si( byte, binary )}
{1025|si( byte, decimal )}
{1025|si( byte, none )}
{1025|si( byte, auto )}
{1025|si( byte, kibi )}
{1025|si( byte, kilo )}
```

If the "UseSIUnits" directive in a configuration override for "site.ini" is set to "false" (the default value), the following output will be produced:

```
1.00 kB
1.00 kB
1.02 kB
1025 B
1.00 kB
```

5

```
1.00 KiB
1.02 kB
```

If the "UseSIUnits" directive in a configuration override for "site.ini" is set to "true", the following output will be produced:

```
1.00 KiB
1.00 KiB
1.02 kB
1025 B
1.00 KiB
1.00 KiB
1.02 kB
```

5

5.8.4   Images

image   (page 1192)
> Creates and returns an image object.

imagefile   (page 1195)
> Loads an image from a file.

texttoimage   (page 1196)
> Renders a string as an image using a truetype font.

5

image

Summary

Creates and returns an image object.

Usage

```
image( value [,...] )
```

Parameters

Name	Type	Description	Required
value	mixed	Please refer to the description below.	Yes.

Returns

An ezimage object.

Description

This operator makes it possible to merge/flatten several images into one image. It returns the final image as an "ezimage" (page 1070) object. Please note that this operator only works if "ImageGD" is installed and enabled. It takes several parameters where each parameter can be a different type. The following types can be used:

- String

- Image layer (an "ezimagelayer" (page 1069) object).

- Array

String

If a parameter is a string, the contents of the string will be used as the alternative image text for the image object that is returned by the operator.

Image layer

If a parameter is an image layer (an "ezimagelayer" (page 1069) object), it will be merged with the other layers and thus become a part of the final result. An image layer may be generated by making use of the "imagefile" (page 1195) or the "texttoimage" (page 1196) operator.

5

Array

If a parameter is an array, the operator will assume that the first element (element number zero) is an image layer and that the second element is an associative array containing parameters that the system will use when the image layer is processed. The following table shows the parameters that can be used.

Name / key	Description
transparency	The image transparency as float value ranging from 0.0 (0% transparency) to 1.0 (100% transparency).
halign	Horizontal alignment as a string: "left", "right" or "center".
valign	Vertical alignment as a string: "top", "bottom" or "center".
x	Absolute horizontal placement as an integer (works with "left" and "right" horizontal alignment.)
y	Absolute vertical placement as an integer (works with "top" and "bottom" vertical alignment.)
xrel	Relative horizontal placement as a float value ranging from 0.0 to 1.0 (works with "left" and "right" horizontal alignment.)
xrel	Relative vertical placement as a float value ranging from 0.0 to 1.0 (works with "top" and "bottom" vertical alignment.)

Please note that the "x" and "xrel" parameters can not be used at the same time. The same goes for the "y" and the "yrel" parameters. When "right" or "bottom" alignment is used, the coordinate system will shift to accommodate the alignment. This is useful for doing alignment and placement since the placement is relative to the current coordinate system. Right alignment will start the axis at the right (0) and go on to the left (width). Bottom alignment will start the axis at the bottom (0) and go on to the top (height).

Template

If the operator is called directly, eZ Publish will display the resulting image using the "layer/ imageobject.tpl" template located in the "templates" directory of the current design or one of the fallback designs. It is possible to override this template using the template override system (page 226).

Examples

Example 1

```
{image( imagefile( 'design/example/images/test1.png' ),
        imagefile( 'design/example/images/test2.png' ),
        'My alternative image text...' )}
```

The "imagefile" (page 1195) operator loads the images from the filesystem and returns them as "ezimagelayer" (page 1069) objects. The "image" operator takes care of merging the images (layers) together into one single image. When the "image" operator is used directly (as in the example above), eZ Publish will insert the "imageobject.tpl" template (or an override) which in turn takes care of displaying the image. The last parameter is a string, which will be used as the alternative image text for the final image.

Example 2

```
{image( "It's A Kind Of Magic"|texttoimage( 'arial' ) )}
```

This example will render the string "It's A Kind Of Magic" using a truetype font specified within the "arial" style. The image layer produced by the "texttoimage" operator is wrapped inside an image object and displayed using the "imageobject.tpl" template (or an override).

Example 3

```
{image( "It's A Kind Of Magic",
        imagefile( 'design/example/images/test1.png' ),
        array( "It's A Kind Of Magic"|texttoimage( 'smartie' ),
               hash( 'transparency', 0.64,
                     'halign',        'right',
                     'valign',        'top' ) ) )}
```

This example will generate an image with 64% transparent text using the "smartie" style. The text will be aligned to the upper right corner of the "example1.png" image. The alternative image text will be "It's A Kind Of Magic" - which is the same as the merged in transparent text.

5

imagefile

Summary

Loads an image from a file.

Usage

```
imagefile( filename )
```

Parameters

Name	Type	Description	Required
filename	string	The image file that should be loaded.	Yes.

Returns

An ezimagelayer object.

Description

This operator loads an image from the filesystem and returns it as an "ezimagelayer" (page 1069) object. The location and the name of the file must be specified in the same way as it is shown in the example below (the whole path must be specified without a starting slash). Please note that this operator will only work if "ImageGD" is installed and enabled.

If this operator is called directly, eZ Publish will display the specified image using the "image.tpl" template located in the "templates/image/" directory of the current design or one of the fallback designs. It is possible to override this template using the template override system (page 226). An image layer can be provided as a parameter to the "image" (page 1192) operator.

Examples

Example 1

```
{imagefile( 'design/example/images/test.png' )}
```

This will load "test.png" from the specified directory and display it using the image layer template.

5

texttoimage

Summary

Renders a string as an image using a truetype font.

Usage

```
input|texttoimage( style )
```

Parameters

Name	Type	Description	Required
style	string	The name of the style that should be used.	Yes.

Returns

Image layer.

Description

This operator creates an image that contains the input text rendered using a truetype font. The style parameter must be used to specify the desired style. If the style parameter is omitted, the default style will be used. The operator returns an "ezimagelayer" (page 1069) object which can be used as a parameter to the "image" (page 1192) operator. If this operator is called directly, eZ Publish will display the specified image using the "layer/image.tpl" template located in the "templates" directory of the current design or one of the fallback designs. It is possible to override this template using the template override system (page 226). Please note that this operator will only work if "ImageGD" is installed and enabled.

An eZ Publish distribution comes with a small collection of truetype fonts that are used by the default styles. These fonts are located in the "/design/standard/fonts" directory. The fonts included in this directory are free when it comes to costs and distribution. Information about the author of a font is placed in a directory with the same name as the font itself.

Default styles

The default styles are defined in the "texttoimage.ini" configuration file. The following list reveals the names of the default styles.

- 1942

- a_d_mono

- archtura

- arial

- gallery

- object_text

- sketchy

- smartie

Please refer to the examples below to see the default styles in action.

Custom styles

It is possible to create custom styles and to make use of custom fonts. For each style, it is possible to configure the following settings:

- The name of the style

- Font family

- Point size

- Background color

- Text color

- Angle/rotation

- X adjustment

- Y adjustment

- Width adjustment

- Height adjustment

- Absolute width

- Absolute height

When using fonts that are located outside the "/design/standard/fonts" directory, for example "/design/example/fonts", the FontDir[] array in a configuration override for "texttoimage.ini" has to include an additional FontDir[] line that specifies the secondary font directory.

Examples

Example 1

```
{'Another World'|texttoimage( '1942' )}
```

The following output will be produced:

(see figure 5.89)

Figure 5.89: *Text rendered as image using the 1942 font.*

Example 2

```
{'Another World'|texttoimage( 'a_d_mono' )}
```

The following output will be produced:

(see figure 5.97)

Figure 5.90: *Text rendered as image using the a_d_mono font.*

Example 3

```
{'Another World'|texttoimage( 'archtura' )}
```

The following output will be produced:

(see figure 5.91)

Figure 5.91: *Text rendered as image using the archtura font.*

Example 4

```
{'Another World'|texttoimage( 'arial' )}
```

The following output will be produced:

(see figure 5.92)

Another World

<div align="center">Figure 5.92: Text rendered as image using the arial font.</div>

Example 5

```
{'Another World'|texttoimage( 'gallery' )}
```

The following output will be produced:

(see figure 5.93)

<div align="center">Another World</div>

<div align="center">Figure 5.93: Text rendered as image using the gallery font.</div>

Example 6

```
{'Another World'|texttoimage( 'object_text' )}
```

The following output will be produced:

(see figure 5.94)

<div align="center">Another World</div>

<div align="center">Figure 5.94: Text rendered as image using the object_text font.</div>

Example 7

```
{'Another World'|texttoimage( 'sketchy' )}
```

The following output will be produced:

(see figure 5.95)

5

ANOTHER WORLD

Figure 5.95: *Text rendered as image using the sketchy font.*

Example 8

```
{'Another World'|texttoimage( 'smartie' )}
```

The following output will be produced:

(see figure 5.96)

Another World

Figure 5.96: *Text rendered as image using the smartie font.*

Example 9

```
{'Another World'|texttoimage()}
```

The following output will be produced (the default font will be used):

(see figure 5.97)

ANOTHER WORLD

Figure 5.97: *Text rendered as image using the a_d_mono font.*

5

5.8.5 Logical operations

and (page 1202)
> Evaluates all parameters to check if any are evaluated as FALSE

choose (page 1204)
> Returns one of the parameters (pinpointed by the input parameter).

cond (page 1205)
> Returns the value of the first clause who's condition is TRUE.

eq (page 1207)
> Returns TRUE if the input equals the first parameter or if all parameters are equal.

false (page 1210)
> Creates and returns a boolean FALSE.

first_set (page 1211)
> Returns the first parameter that is set (or FALSE).

ge (page 1213)
> Returns TRUE if a parameter is greater than or equal to another parameter.

gt (page 1215)
> Returns TRUE if a parameter is greater than another parameter.

le (page 1217)
> Returns TRUE if a parameter is less than or equal to another parameter.

lt (page 1219)
> Returns TRUE if a parameter is less than another parameter.

ne (page 1221)
> Returns TRUE if one or more of the parameters do not match.

not (page 1223)
> Returns the opposite of the input or the first parameter (TRUE/FALSE).

null (page 1225)
> Returns TRUE if the input value is NULL (not the same as 0).

or (page 1226)
> Evaluates all parameters until one is found to be TRUE, returns that value.

true (page 1228)
> Creates and returns a boolean TRUE.

5

and

Summary

Evaluates all parameters to check if any are evaluated as FALSE

Usage

```
and( value1 [, value2 [, ... ] ] )
```

Parameters

Name	Type	Description	Required
value1	any	A variable/value to be evaluated.	Yes.
value2	any	Another variable/value to be evaluated.	No.

Returns

One of the parameters or TRUE (see below).

Description

This operator evaluates all parameters.
When one of them is found to be FALSE, the operator will then return that parameter. Under some circumstances shortcutting is then applied, which means that once a parameter in found to be FALSE, the evaluation of the rest of the parameters stops. However this is not always the case, and depends on personal settings.
If none of the parameters are found to be FALSE, the operator will return TRUE.

The following table shows how the different types are evaluated by this operator.

Type	Evaluation
Number	TRUE if the value is non-zero, FALSE otherwise.
String	TRUE if the string consists of at least one character, FALSE otherwise.
Boolean	TRUE if the boolean is a TRUE value, FALSE is otherwise.
Array	TRUE if the array has one or more elements, FALSE otherwise.
Object	TRUE if the object provides the "attributes" and the "attribute" methods, FALSE otherwise.
NULL	Always FALSE.
Other	Other types will be evaluated in the same way as PHP would do.

5

Examples

Example 1

```
{if and( false(), true(), false() )}
    The truth is out there.
{else}
    The day the earth stood still.
{/if}
```

The following output will be produced: "The day the earth stood still.".

Example 2

```
{def $a=array()
     $b=array( 1, 2, 3 )
     $c=array( 4, 5, 6 )}

{and( $a, $b, $c )}
```

The code above will return the empty array that is represented by $a.

5

choose

Summary

Returns one of the parameters (pinpointed by the input parameter).

Usage

```
input|choose( value1 [, value2 [, ... ] ] )
```

Parameters

Name	Type	Description	Required
value1	any	A variable/value of any kind.	Yes.
value2	any	Another variable/value of any kind.	No.

Returns

One of the parameters.

Description

This operator returns one of the parameters. The input parameter must be an integer that pinpoints exactly which parameter that should be returned. If the input parameter is zero, the operator will return the first parameter. If the input parameter is one, the operator will return the second parameter, and so on. If the offset is wrong, the operator will return FALSE.

Examples

Example 1

```
{0|choose( 'apples', 'bananas', 'coconuts' )}
```

The following output will be produced: "apples".

Example 2

```
{2|choose( 'apples', 'bananas', 'coconuts' )}
```

The following output will be produced: "coconuts".

5

cond

Summary

Returns the value of the first clause who's condition is TRUE.

Usage

```
cond( cond1, value1 [, cond2, value2 [, ... ] ] )
```

Parameters

Name	Type	Description	Required
cond1	boolean	Match condition 1.	Yes.
value1	any	Return value for condition 1.	Yes.
cond2	boolean	Match condition 2.	No.
value2	any	Return value for condition 2.	No.

Returns

One of the provided values or FALSE.

Description

This operator evaluates the provided conditions (odd numbered parameters). If one of the conditions evaluates to TRUE then the value which is associated with that condition is returned. If none of the conditions are TRUE, the operator will return FALSE. If an odd number of parameters are provided, the operator will return the last parameter if all conditions fail.

Examples

Example 1

```
{cond( true(), 'apples', true(), 'bananas' )}
```

The following output will be produced: "apples".

Example 2

```
{cond( false(), 'apples', true(), 'bananas' )}
```

The following output will be produced: "bananas".

Example 3

```
{cond( false(), 'apples', false(), 'bananas' )}
```

No output will be produced, the operator will return FALSE.

Example 4

```
{cond( false(), 'apples', 'bananas' )}
```

The following output will be produced: "bananas".

5

eq

Summary

Returns TRUE if the input equals the first parameter or if all parameters are equal.

Usage

```
input|eq( value1 [, value2 ] )
```

Parameters

Name	Type	Description	Required
value1	any	A variable/value that should be compared.	Yes.
value2	any	Another variable/value that should be compared.	No.

Returns

TRUE or FALSE (see below).

Description

This operator compares the contents of two or more variables and/or values. If the input parameter is used, the operator will compare it with the first parameter. If the provided variables/values match, the operator will return TRUE, otherwise FALSE will be returned. If more than one parameter is provided, the operator will compare all parameters. If all parameters are found to be equal, the operator will return TRUE, otherwise FALSE will be returned. If more than one parameter is provided, the operator will simply ignore the input parameter.

Note that "eq" compares the values in the same way as the '==' operator in PHP programming language (for example, 0.1 and 0.10 will be equal). Refer to the PHP reference documentation for more information. It is recommended to use the €compare (page 1297)€ template operator for string comparision.

Examples

Example 1

```
{if 1|eq( 1 )}
    The truth is out there.
{else}
    The day the earth stood still.
{/if}
```

5

The following output will be produced: "The truth is out there".

Example 2

```
{if 1|eq( 2 )}
    The truth is out there.
{else}
    The day the earth stood still.
{/if}
```

The following output will be produced: "The day the earth stood still".

Example 3

```
{if eq( 1, 1 )}
    The truth is out there.
{else}
    The day the earth stood still.
{/if}
```

The following output will be produced: "The truth is out there".

Example 4

```
{if eq( 1, 2 )}
    The truth is out there.
{else}
    The day the earth stood still.
{/if}
```

The following output will be produced: "The day the earth stood still".

Example 5

```
{if 1|eq( 1, 1 )}
    The truth is out there.
{else}
    The day the earth stood still.
{/if}
```

The following output will be produced: "The truth is out there".

5

Example 6

```
{if 2|eq( 1, 1 )}
    The truth is out there.
{else}
    The day the earth stood still.
{/if}
```

The following output will be produced: "The truth is out there.".

Example 7

```
{if 1|eq( 1, 2 )}
    The truth is out there.
{else}
    The day the earth stood still.
{/if}
```

The following output will be produced: "The day the earth stood still".

5

false

Summary

Creates and returns a boolean FALSE.

Usage

```
false()
```

Returns

FALSE.

Description

This operator creates and returns a boolean FALSE. It can be used to define a boolean variable and in logical comparisons.

Examples

Example 1

```
{def $my_boolean=false()}

{if $my_boolean}
    The truth is out there.
{else}
    The day the earth stood still.
{/if}
```

The following output will be produced: "The day the earth stood still.".

5

first_set

Summary

Returns the first parameter that is set (or FALSE).

Usage

```
first_set( value1 [, value2 [, ... ] ] )
```

Parameters

Name	Type	Description	Required
value1	any	A variable/value that should be evaluated.	Yes.
value2	any	Another variable/value that should be evaluated.	No.

Returns

The first value that is set or FALSE.

Description

This operator evaluates all parameters until one of them is found to be set. The parameter that is found to be set will be returned. If none of the parameters are set, the operator will return FALSE.

Examples

Example 1

```
{if first_set( $a, $b, $c )}
    The truth is out there.
{else}
    The day the earth stood still.
{/if}
```

As long as $a, $b and $c are undeclared/unset, the following output will be produced: "The day the earth stood still".

Example 2

```
{first_set( $a, 256, $b )}
```

As long as $a is undeclared/unset, the following output will be produced: "256".

5

ge

Summary

Returns TRUE if a parameter is greater than or equal to another parameter.

Usage

```
input|ge( value1 [, value2 ] )
```

Parameters

Name	Type	Description	Required
value1	any	A variable/value that should be compared.	Yes.
value2	any	Another variable/value that should be compared.	Only if the input parameter is omitted.

Returns

TRUE or FALSE (see below).

Description

This operator compares two parameters. It returns TRUE if the first parameter is greater than or equal to the second parameter; otherwise FALSE will be returned. If the input parameter is provided, the operator will compare it with the first parameter; otherwise it is the first and the second parameter that will be compared. The following table shows how the different types are treated.

Type	Value
Number	The value of the number is used.
String	The number of characters is used.
Boolean	FALSE means 0 and TRUE means 1.
Array	The number of elements is used.
Object	The number of object attributes is used.
Other	Always 0.

5

Examples

Example 1

```
{256|ge( 128 )}
```

or

```
{ge( 256, 128 )}
```

Returns TRUE.

Example 2

```
{128|ge( 256 )}
```

or

```
{ge( 128, 256 )}
```

Returns FALSE.

Example 3

```
{256|ge( 256 )}
```

or

```
{ge( 256, 256 )}
```

Returns TRUE.

Example 4

```
{128|ge( 256, 64 )}
```

Returns FALSE.

5

gt

Summary

Returns TRUE if a parameter is greater than another parameter.

Usage

```
input|gt( value1 [, value2 ] )
```

Parameters

Name	Type	Description	Required
value1	any	A variable/value that should be compared.	Yes.
value2	any	Another variable/value that should be compared.	Only if the input parameter is omitted.

Returns

TRUE or FALSE (see below).

Description

This operator compares two parameters. It returns TRUE if the first parameter is greater than the second parameter; otherwise FALSE will be returned. If the input parameter is provided, the operator will compare it with the first parameter; otherwise it is the first and the second parameter that will be compared. The following table shows how the different types are treated.

Type	Value
Number	The value of the number is used.
String	The number of characters is used.
Boolean	FALSE means 0 and TRUE means 1.
Array	The number of elements is used.
Object	The number of object attributes is used.
Other	Always 0.

5

Examples

Example 1

```
{256|gt( 128 )}
```

or

```
{gt( 256, 128 )}
```

Returns TRUE.

Example 2

```
{128|gt( 256 )}
```

or

```
{gt( 128, 256 )}
```

Returns FALSE.

Example 3

```
{256|gt( 256 )}
```

or

```
{gt( 256, 256 )}
```

Returns FALSE.

Example 4

```
{128|gt( 64, 256 )}
```

Returns TRUE.

5

le

Summary

Returns TRUE if a parameter is less than or equal to another parameter.

Usage

```
input|le( value1 [, value2 ] )
```

Parameters

Name	Type	Description	Required
value1	any	A variable/value that should be compared.	Yes.
value2	any	Another variable/value that should be compared.	Only if the input parameter is omitted.

Returns

TRUE or FALSE (see below).

Description

This operator compares two parameters. It returns TRUE if the first parameter is less than or equal to the second parameter; otherwise FALSE will be returned. If the input parameter is provided, the operator will compare it with the first parameter; otherwise it is the first and the second parameter that will be compared. The following table shows how the different types are treated.

Type	Value
Number	The value of the number is used.
String	The number of characters is used.
Boolean	FALSE means 0 and TRUE means 1.
Array	The number of elements is used.
Object	The number of object attributes is used.
Other	Always 0.

5

Examples

Example 1

```
{256|le( 128 )}
```

or

```
{le( 256, 128 )}
```

Returns FALSE.

Example 2

```
{128|le( 256 )}
```

or

```
{le( 128, 256 )}
```

Returns TRUE.

Example 3

```
{256|le( 256 )}
```

or

```
{le( 256, 256 )}
```

Returns TRUE.

Example 4

```
{128|le( 256, 64 )}
```

Returns TRUE.

5

lt

Summary

Returns TRUE if a parameter is less than another parameter.

Usage

```
input|lt( value1 [, value2 ] )
```

Parameters

Name	Type	Description	Required
value1	any	A variable/value that should be compared.	Yes.
value2	any	Another variable/value that should be compared.	Only if the input parameter is omitted.

Returns

TRUE or FALSE (see below).

Description

This operator compares two parameters. It returns TRUE if the first parameter is less than the second parameter; otherwise FALSE will be returned. If the input parameter is provided, the operator will compare it with the first parameter; otherwise it is the first and the second parameter that will be compared. The following table shows how the different types are treated.

Type	Value
Number	The value of the number is used.
String	The number of characters is used.
Boolean	FALSE means 0 and TRUE means 1.
Array	The number of elements is used.
Object	The number of object attributes is used.
Other	Always 0.

5

Examples

Example 1

```
{256|lt( 128 )}
```

or

```
{lt( 256, 128 )}
```

Returns FALSE.

Example 2

```
{128|lt( 256 )}
```

or

```
{lt( 128, 256 )}
```

Returns TRUE.

Example 3

```
{256|lt( 256 )}
```

or

```
{lt( 256, 256 )}
```

Returns FALSE.

Example 4

```
{128|lt( 256, 64 )}
```

Returns TRUE.

ne

Summary

Returns TRUE if one or more of the parameters do not match.

Usage

```
input|ne( value1 [, value2 [, ...] ] )
```

Parameters

Name	Type	Description	Required
value1	any	A variable/value that should be compared..	Yes.
value2	any	Another variable/value that should be compared.	Only if the input parameter is omitted.

Returns

TRUE or FALSE (see below).

Description

This operator compares all the provided parameters. If the parameters are not equal, the operator will return TRUE, otherwise FALSE will be returned. If more than one parameter is provided, the operator will ignore the input parameter.

Note that "ne" compares the values in the same way as the '!=' operator in PHP programming language (refer to the PHP reference documentation for more information). It is recommended to use the €compare (page 1297)€ template operator for string comparision.

Examples

Example 1

```
{128|ne( 128 )}
```

or

```
{ne( 128, 128 )}
```

Returns FALSE.

Example 2

```
{128|ne( 256 )}
```

or

```
{ne( 128, 256 )}
```

Returns TRUE.

Example 3

```
{256|ne( 256, 128 )}
```

Returns TRUE.

Example 4

```
{ne( 128, 128, 256 )}
```

Returns TRUE.

Example 5

```
{ne( 128, 128, 128 )}
```

Returns FALSE.

5

not

Summary

Returns the opposite of the input or the first parameter (TRUE/FALSE).

Usage

```
input|not( test )
```

Parameters

Name	Type	Description	Required
test	any	The variable/value that should be tested.	No.

Returns

TRUE or FALSE

Description

This operator returns TRUE if the input value is FALSE and vice versa. It is also possible to use the optional "test" parameter for evaluation. If both the input parameter and the "test" parameter are used, it is the input parameter that will be evaluated.

Examples

Example 1

```
{if false()|not()}
    The truth is out there.
{else}
    The day the earth stood still.
{/if}
```

The following output will be produced: "The truth is out there.".

Example 2

5

```
{if true()|not()}
    The truth is out there.
{else}
    The day the earth stood still.
{/if}
```

The following output will be produced: "The day the earth stood still.".

Example 3

```
{if not( false() )}
    The truth is out there.
{else}
    The day the earth stood still.
{/if}
```

The following output will be produced: "The truth is out there.".

Example 4

```
{if not( true() )}
    The truth is out there.
{else}
    The day the earth stood still.
{/if}
```

The following output will be produced: "The day the earth stood still.".

5

null

Summary

Returns TRUE if the input value is NULL (not the same as 0).

Usage

```
input|null()
```

Returns

TRUE or FALSE.

Description

This operator returns TRUE if the input value is NULL, otherwise FALSE will be returned. Please note that NULL is not the same as a numeric zero (0).

Examples

Example 1

```
{if 0|null()}
    The truth is out there.
{else}
    The day the earth stood still.
{/if}
```

The following output will be produced: "The day the earth stood still".

5

or

Summary

Evaluates all parameters until one is found to be TRUE, returns that value.

Usage

```
or( value1 [, value2 [, ... ] ] )
```

Parameters

Name	Type	Description	Required
value1	any	A variable/value to be evaluated.	Yes.
value2	any	Another variable/value to be evaluated.	No.

Returns

One of the parameters or FALSE (see below).

Description

This operator evaluates all parameters until one of them is found to be TRUE. The operator will then return that parameter and thus stop evaluating the rest of the parameters. If none of the parameters are found to be TRUE, the operator will return FALSE. The following table shows how the different types are evaluated by this operator.

Type	Evaluation
Number	TRUE if the value is non-zero, FALSE otherwise.
String	TRUE if the string consists of at least one character, FALSE otherwise.
Boolean	TRUE if the boolean is a TRUE value, FALSE is otherwise.
Array	TRUE if the array has one or more elements, FALSE otherwise.
Object	TRUE if the object provides the "attributes" and the "attribute" methods, FALSE otherwise.
NULL	Always FALSE.
Other	Other types will be evaluated in the same way as PHP would do.

5

Examples

Example 1

```
{if or( false(), true(), false() )}
    The truth is out there.
{else}
    The day the earth stood still.
{/if}
```

The following output will be produced: "The truth is out there.".

Example 2

```
{def $a=array()
     $b=array( 1, 2, 3 )
     $c=array( 4, 5, 6 )}

{or( $a, $b, $c )}
```

The code above will return the following array: (1, 2, 3).

5

true

Summary

Creates and returns a boolean TRUE.

Usage

```
true()
```

Returns

TRUE.

Description

This operator creates and returns a boolean TRUE. It can be used to define a boolean variable and in logical comparisons.

Examples

Example 1

```
{def $my_boolean=true()}

{if $my_boolean}
    The truth is out there.
{else}
    The day the earth stood still.
{/if}
```

The following output will be produced: "The truth is out there.".

5

5.8.6   Mathematics

abs   (page 1230)
>   Returns a positive value of either the input or the first parameter.

ceil   (page 1232)
>   Returns the next highest integer value of input or parameter.

dec   (page 1234)
>   Returns input or parameter decremented by one.

div   (page 1236)
>   Divides input or first parameter by the remaining parameters.

floor   (page 1238)
>   Returns the next lowest integer value of input or parameter.

inc   (page 1240)
>   Increments either the input or the first parameter with one.

max   (page 1242)
>   Returns the largest value of all parameters.

min   (page 1243)
>   Returns the smallest value of all parameters.

mod   (page 1244)
>   Returns the modulo of two parameters.

mul   (page 1246)
>   Multiplies all parameters and returns the result.

rand   (page 1248)
>   Returns a random integer.

round   (page 1250)
>   Returns a rounded version of the input or a parameter value.

sub   (page 1252)
>   Subtracts all remaining parameters from the first parameter.

sum   (page 1254)
>   Returns the sum of all parameters.

5

abs

Summary

Returns a positive value of either the input or the first parameter.

Usage

```
input|abs( value )
```

Parameters

Name	Type	Description	Required
value	number	Value to calculate absolute of.	Only if the input parameter is omitted.

Returns

Absolute value of input or parameter.

Description

Returns a positive value of either the input or the "value" parameter. If both are provided, it is the "value" parameter that will be used.

Examples

Example 1

```
{-16|abs}
```

or

```
{abs( -16 )}
```

The following output will be produced: "16".

5

Example 2

```
{abs( 256 )}
```

The following output will be produced: "256".

Example 3

```
{-64|abs( -128 )}
```

The following output will be produced: "128".

5

ceil

Summary

Returns the next highest integer value of input or parameter.

Usage

```
input|ceil( value )
```

Parameters

Name	Type	Description	Required
value	number	Value to be rounded up.	Only if the input parameter is omitted.

Returns

Integer (rounded up version of input/parameter).

Description

This operator returns the next highest integer value by rounding up either the input or the "value" parameter. If both are provided, it is the "value" parameter that will be used.

Examples

Example 1

```
{1.5|ceil}
```

or

```
{ceil( 1.5 )}
```

The following output will be produced: "2".

Example 2

```
{5.5|ceil( 8.2 )}
```

The following output will be produced: "9".

5

dec

Summary

Returns input or parameter decremented by one.

Usage

```
input|dec( value )
```

Parameters

Name	Type	Description	Required
value	number	The value that should be decremented.	Only if the input parameter is omitted.

Returns

input/parameter decremented with one.

Description

This operator decrements either the input or the "value" parameter with one and returns the result as an integer. If both are provided, it is the "value" parameter that will be used. Please note that this operator can not be used directly to decrement the value of a variable (please refer to the last example).

Examples

Example 1

```
{256|dec}
```

or

```
{dec( 256 )}
```

The following output will be produced: 255.

5

Example 2

```
{200|dec( 250 )}
```

The following output will be produced: 249.

Example 3

```
{def $i=256}
{set $i=dec( $i )}
{$i}
```

This example demonstrates how the value of a variable can be decremented using the "set" and the "dec" operators. The following output will be produced: "255".

5

div

Summary

Divides input or first parameter by the remaining parameters.

Usage

```
input|div( value1 [, value2 [, ...] ] )
```

Parameters

Name	Type	Description	Required
value1	number	Dividend or divisor.	Yes.
value2	number	Dividend.	Only if the input parameter is omitted.

Returns

The result of the division.

Description

This operator divides a the first parameter (can be the input parameter) by the rest of the parameters.

Examples

Example 1

```
{10|div( 5 )}
```

or

```
{div( 10, 5 )}
```

The following output will be produced: "2".

Example 2

```
{12|div( 3, 2 )}
```

or

```
{div( 12, 3, 2 )}
```

The following output will be produced: "2".

5

floor

Summary

Returns the next lowest integer value of input or parameter.

Usage

```
input|floor( value )
```

Parameters

Name	Type	Description	Required
value	number	Value to be rounded down.	Only if the input parameter is omitted.

Returns

Integer (rounded down version of input/parameter).

Description

This operator returns the next lowest integer by rounding down either the input or the "value" parameter. If both are provided, it is the "value" parameter that will be used.

Examples

Example 1

```
{256.7|floor}
```

or

```
{floor( 256.7 )}
```

The following output will be produced: "256".

5

Example 2

```
{999.2|floor( 256.7 )}
```

The following output will be produced: "256".

5

inc

Summary

Increments either the input or the first parameter with one.

Usage

```
input|inc( value )
```

Parameters

Name	Type	Description	Required
value	number	The value that should be incremented.	Only if the input parameter is omitted.

Returns

Number (input/parameter incremented with one).

Description

This operator increments either the input or the "value" parameter with one and returns the result as an integer. If both are provided, it is the "value" parameter that will be used. Please note that this operator can not be used directly to increment the value of a variable (please refer to the last example).

Examples

Example 1

```
{255|inc}
```

or

```
{inc( 255 )}
```

The following output will be produced: "256".

5

Example 2

```
{def $i=255}
{set $i=inc( $i )}
{$i}
```

The following output will be produced: "256".

5

max

Summary

Returns the largest value of all parameters.

Usage

```
max( value1, value2 [,...] )
```

Parameters

Name	Type	Description	Required
value1	any	A value that should be evaluated.	Yes.
value2	any	Another value that should be evaluated.	Yes.

Returns

The largest value of all parameters.

Description

This operator returns the largest value of all parameters. The input parameter is ignored.

Examples

Example 1

```
{max( 2, 3, 1 )}
```

The following output will be produced: "3".

Example 2

```
{max( array( 1, 2 ), array( 1, 2, 3 ) )}
```

The following array will be returned: (1, 2, 3).

min

Summary

Returns the smallest value of all parameters.

Usage

```
min( value1, value2 [,...] )
```

Parameters

Name	Type	Description	Required
value1	any	A value that should be evaluated.	Yes.
value2	any	Another value that should be evaluated.	Yes.

Returns

The smallest value of all parameters.

Description

This operator returns the smallest value of all parameters. The input parameter is ignored.

Examples

Example 1

```
{min( 10, 20, 6, 40, 50 )}
```

The following output will be produced: "6".

Example 2

```
{min( array( 1, 2 ), array( 1, 2, 3 ) )}
```

The following array will be returned: (1, 2).

5

mod

Summary

Returns the modulo of two parameters.

Usage

```
input|mod( value1 [,value2] )
```

Parameters

Name	Type	Description	Required
value1	number	Divisor or dividend.	Yes.
value2	number	Divisor or dividend.	Only if the input parameter is omitted.

Returns

Integer (the modulo of the supplied parameters).

Description

This operator returns the modulo (rest after division) of the first parameter divided by the second. The operator can also take an input parameter. If the input parameter is used, then it will be divided by the first parameter (and thus the second parameter will be ignored).

Examples

Example 1

```
{5|mod( 3 )}
```

or

```
{mod( 5, 3 )}
```

The following output will be produced: "2".

5

Example 2

```
{6|mod( 3, 7 )}
```

The following output will be produced: "0".

5

mul

Summary

Multiplies all parameters and returns the result.

Usage

```
input|mul( value1 [, value2] [, ...] )
```

Parameters

Name	Type	Description	Required
value1	number	Multiplicand.	Yes.
value2	number	Multiplicand.	Only if the input parameter is omitted.

Returns

The result of the multiplication (integer or float).

Description

This operator multiplies all parameters and returns the result. If an input parameter is provided, it will be included in the multiplication.

Examples

Example 1

```
{2|mul( 3 )}
```

or

```
{mul( 2, 3 )}
```

The following output will be produced: "6".

5

Example 2

```
{2|mul( 3, 4 )}
```

The following output will be produced: ”24”.

5

rand

Summary

Returns a random integer.

Usage

```
rand( [min_val, max_val] )
```

Parameters

Name	Type	Description	Required
min_val	integer	Lower limit (inclusive).	No.
max_val	integer	Upper limit (inclusive).	No.

Returns

Integer.

Description

This operator returns a random integer. The optional parameters "min_val" and "max_val" can be used to control the range from which the random number is picked. If you want a random integer between 3 and 13 (inclusive), use (3, 13) as parameters.

Examples

Example 1

```
rand()
```

Returns a random integer.

Example 2

```
rand( 5 )
```

Returns a random integer that is greater than 4.

Example 3

```
rand( 3, 13 )
```

Returns a random integer that is greater than 2 and less than 14.

5

round

Summary

Returns a rounded version of the input or a parameter value.

Usage

```
input|round( value )
```

Parameters

Name	Type	Description	Required
value	number	The number that should be rounded off.	Only if the input parameter is omitted.

Returns

A rounded off version of the provided value (an integer or float).

Description

This operator rounds off the value that was specified using either the input or the "value" parameter. If both are provided, it is the "value" parameter that will be used. The operator returns the rounded off value.

Examples

Example 1

```
{15.7|round}
```

or

```
{round( 15.7 )}
```

The following output will be produced: "16".

Example 2

```
{8.4|round( 9.7 )}
```

The following output will be produced: "10".

5

sub

Summary

Subtracts all remaining parameters from the first parameter.

Usage

```
input|sub( value [, ...] )
```

Parameters

Name	Type	Description	Required
value	number	A number that should be included in the subtraction.	Yes.

Returns

Number (result of subtraction).

Description

This operator subtracts all remaining parameters from the first parameter and returns the result. If an input parameter is provided, all other parameters will subtracted from it.

Examples

Example 1

```
{10|sub( 2 )}
```

or

```
{sub( 10, 2 )}
```

The following output will be produced: "8".

Example 2

```
{sub( 10, 2, 3 )}
```

The following output will be produced: "5".

Example 3

```
{10|sub( 10, 2, 3 )}
```

The following output will be produced: "-5".

5

sum

Summary

Returns the sum of all parameters.

Usage

```
input|sum( value [,...] )
```

Parameters

Name	Type	Description	Required
value	number	A value that should be added to the result.	Only if the input parameter is omitted.

Returns

Number (sum of all parameters + input).

Description

This operator adds up all the parameters (including the input parameter) and returns the result.

Examples

Example 1

```
{1|sum( 2 )}
```

or

```
{sum( 1, 2 )}
```

The following output will be produced: "3".

5

Example 2

```
{1|sum( 2, -3, 4)}
```

The following output will be produced: "4".

5.8.7   Miscellaneous

\begin{description} \item[action_icon]\refer{labelc5c41e3754f54b415ae37594b6de23dc}\ \\
Not documented yet.   \item[attribute]\refer{label42def8c8cec5b3b1010326258842f440}\
\\ Makes it possible to inspect the contents of arrays, hashes and ob-
jects.   \item[classgroup_icon]\refer{labela9681094855b2af265595db841074e99}\
\\ Outputs an image tag referencing a class group icon.   \item[class_
icon]\refer{label8f0b6739fd08886f2a6abf3d650a30bc}\ \\ Outputs an image tag referencing a
class icon. \item[content_structure_tree]\refer{label4718c24065715a7d210c2f699ca89ce3}\
\\ Not documented yet.   \item[dump]\refer{label1104e06f4294967d064e9074f9756e7b}\
\\ Makes it possible to inspect the contents of arrays, hashes and ob-
jects.   Alias of attribute() available in 4.7 and higher which is simpler to use.
\item[ezpackage]\refer{label39d162555979860c0d2b0a58945078ef}\ \\ Not documented yet.
\item[flag_icon]\refer{labela9baefde9440c94ee18c7a5ecd968709}\ \\ Outputs an image tag
referencing a flag icon.   \item[gettime]\refer{labelf31f3dac286db2b39b6a0dc2cd6a0b78}\
\\ Converts a UNIX timestamp to a human friendly structure.   \item[icon_
info]\refer{label28645f42eb410b24193e997f93e29531}\   \\   Not   documented   yet.
\item[makedate]\refer{label9a3fa564795bf9ce0e2475b1cb549dbf}\ \\ Generates the UNIX
timestamp of a given date. \item[maketime]\refer{labelfa96a252baed3e3594638bdc8449284b}\
\\ Generates the UNIX timestamp of a given date/time.   \item[mimetype_
icon]\refer{labeld6253a383bd104a6709eaa30788e1890}\   \\   Outputs
an   image   tag   referencing   a   MIME   type   icon.   \item[month_
overview]\refer{labelda9469e0730cff7272058d50f4c2039a}\ \\ Generates a structure that can
be used to build a calendar.   \item[pdf]\refer{label7f803a7218202074268a012c71daef62}\
\\   Provides   access   to   the   PDF   functions   (DEPRECATED).
\item[roman]\refer{label790e5c71d23340e3d35add861f1bd633}\ \\ Generates a roman rep-
resentation of a number. \item[topmenu]\refer{label811917cb343752d00fff26ea56ee6cf8}\ \\
Not documented yet. \item[treemenu]\refer{label77d0df26f34298feb19660c6a5d1a355}\ \\
Fetches a subtree of nodes for the purpose of menu generation.

\end{description}

5

action_icon

Summary

Not documented yet.

attribute

Summary

Makes it possible to inspect the contents of arrays, hashes and objects.

Usage

```
eZ Publish 4.5 and earlier releases:
input|attribute( [show_values [, level [, table ] ] ] )

eZ Publish 4.6 and later
input|attribute( [show_values [, level [, outputformat ] ] ] )
```

Parameters

Name	Type	Description	Required
show_values	string	Sets whether to extract values in addition to keys, names, etc. If "show" is passed, the values will be returned. Otherwise, the operator will not return any values.	No.
level	integer	The number of levels that should be processed (default is 2).	No.
table (pre 4.6)*	boolean	Return result as HTML table (default) or not.	No.
outputformat (from 4.6)*	string	Specify output format. Default output format is 'html', but 'text' is also supported. Other custom output formats can be written. See in the Note 2 below.	No.

Returns

A string revealing information about the target.

Description

This operator extracts all available keys, attribute names and/or methods that belong to the input parameter (must be either an object, an array or a hash). By default, the array keys, object attribute names and their types will be revealed. By passing "show" as the first parameter, the operator will also return the values. The second parameter can be used to control the number of levels/children that should be expanded and included in the result (the default setting is 2). A large level value may cause the system to be trapped in a recursive/infinite loop. The returned result is an HTML table containing the retrieved information. If "false()" is passed as the third parameter, the output will be a plain string instead of an HTML table. Please refer to the "Array

5

and object inspection" (page 205)section of the "Templates" chapter for more information about the use of this operator.

*__Note 1:__ As of version 4.6 the parameter "outputformat" has become a string. The string specifies the output format. The default output format is 'html', but 'text' is also supported. Other custom output formats can be written. See in "Description" field below.

*__Note 2:__ As of the 4.6 version of eZ Publish you can create your own output format handlers (page 1261).

Examples

Example 1

```
{def $example=hash( 'Name',  'John Doe',
                    'Age',   24,
                    'Phone', '555-3212' )}

{$example|attribute()}
```

The following output will be produced:

Attribute	Type
Name	string
Age	integer
Phone	string

Example 2

```
{def $example=hash( 'Name',  'Jane Doe',
                    'Age',   23,
                    'Phone', '555-3213' )}

{$example|attribute( 'show' )}
```

The following output will be produced:

Attribute	Type	Value
Name	string	'Jane Doe'
Age	integer	23
Phone	string	'555-3213'

Example 3

```
{def $example=hash( 'Name',  'Jane Doe',
                    'Age',   25,
                    'Phone', '555-3213' )}

{$example|attribute( 'something' )}
```

The following output will be produced:

Attribute	Type
Name	string
Age	integer
Phone	string

5

How to create a custom output formatter for the template operator attribute

eZ Publish 4.6 allows to create a custom output formatter for the attribute() template operator. This document explains process how to create a custom output formatter.

Create a custom extension folder

First we need to create a custom extension folder where we will place our code and configuration settings. Create a "*ezattributeformatter*" folder under extension folder inside eZ Publish root folder. You should have a following directory structure:
extension
€€ ezattributeformatter

   €€ classes
     €€ settings

Create php file

Next inside settings folder create a "*template.ini.append.php*" file with the following settings:

```
<?php /*

[AttributeOperator]
OutputFormatter[custom]=ezpAttributeOperatorCustomFormatter

*/ ?>
```

Key 'custom' for "*OutputFormatter*" is our custom attribute() template operator identifier. Value is a PHP class which has to implement "*ezpAttributeOperatorFormatterInterface*".

Implement PHP class

Once we have prepared our configuration we need to implement our custom formatter PHP class. In this example we will extend the base "*ezpAttributeOperatorFormatter*" class which provides for your convenient two useful methods which returns type and value of object, array, etc. Note that your class has to implement at least "*ezpAttributeOperatorFormatterInterface*".

```
class ezpAttributeOperatorBKMLFormatter extends ezpAttributeOperatorFormatter
implements ezpAttributeOperatorFormatterInterface
{

   /**
    * Formats header for the 'attribute' template operator output
    *
```

5

```php
     * @param string $value
     * @param bool $showValues
     * @return string
     */
    public function header( $value, $showValues )
    {
        $headers = "Attribute [Type]";
        if ( $showValues )
            ;$headers  .= ": Value";

        return "<h1>{$headers}</h1>{$value}";
    }


    /**
     * Formats single line for the 'attribute' template operator output
     *
     * @param string $key
     * @param mixed $item
     * @param bool $showValues
     * @param int $level
     * @return string
     */
    public function line( $key, $item, $showValues, $level )
    {
        $type = $this->getType( $item );
        $value = $this->getValue( $item );
        $spacing = str_repeat( "&gt;", $level );

        if ( $showValues )
            $output  = "<div><strong>{$spacing}{$key}</
strong> [ {$type} ]  : {$value}</div>";
        else
            $output  = "<div><strong>{$spacing}{$key}</
strong> [ {$type} ]  : {$value}</div>";

        return $output;
    }
}
```

The "*ezpAttributeOperatorFormatterInterface::header()*" method allows you to customize a header style and format while the "*ezpAttributeOperatorFormatterInterface::line()*" method allows you to customize a single line/row of the attribute() template operator output.

**Activate extension   **

Next we need to activate our extension. Add following line in to the settings/override/ site.ini.append.php:

```
[ExtensionSettings]
ActiveExtensions[]
[...]
ActiveExtensions[]=ezattributeformatter
```

Update autoload array

Once our extension is activated we need to update the PHP autoload array. In order to do so, execute following command from eZ Publish root folder:

```
php bin/php/ezpgenerateautoloads.php -e -p
```

Use the new output formatter

Now you we can use a new output formatter for attribute() template operator as follows:

```
{$node|attribute('show', 2, 'custom')}
```

5

classgroup_icon

Summary

Outputs an image tag referencing a class group icon.

Usage

```
input|classgroup_icon( [ size [, alt_text [, return_uri ] ] ] )
```

Parameters

Name	Type	Description	Required
size	string	The preferred icon size (small, medium, large, etc.).	No.
alt_text	string	The alternative image text.	No.
return_uri	boolean	The return format (image tag or just the address).	No.

Returns

A string containing an image tag.

Description

This operator generates an image tag that references a class group icon. The name of the icon must be provided using the input parameter. The "size", "alt_text" and "return_url" parameters are optional (see the description above). The operator uses the settings provided by the "icon.ini" configuration file (or an override).

Examples

Example 1

```
{'content'|classgroup_icon( 'small', 'Alternative image text' )}
```

The following output will be produced:

```
<img src="/share/icons/crystal-admin/16x16_indexed/filesystems/
folder_txt.png" width="16" height="16" alt="Content" title="Content" />
```

5

class icon

Summary

Outputs an image tag referencing a class icon.

Usage

```
input|class_icon( [ size, [, alt_text ] ] )
```

Returns

A string containing an image tag.

Description

This operator generates an image tag that references a class icon. The name of the icon must be provided using the input parameter. The "size" and "alt_text" parameters are optional (see the description above). The operator uses the settings provided by the "icon.ini" configuration file (or an override).

Examples

Example 1

```
{'folder'|class_icon( 'small', 'Alternative image text' )}
```

The following output will be produced:

5

content_structure_tree

Summary

Not documented yet.

dump

Summary

Makes it possible to inspect the contents of arrays, hashes and objects. Alias of attribute() available in 4.7 and higher which is simpler to use.

Usage

```
eZ Publish 4.7 and later
input|dump( [show_values [, level [, outputformat ] ] ] )
```

Parameters

Name	Type	Description	Required
show_values	string	Sets whether to extract values in addition to keys, names, etc. If "show" is passed, the values will be returned. Otherwise, the operator will not return any values. Default is "show"	No.
level	integer	The number of levels that should be processed (default is 1).	No.
outputformat	string	Specify output format. Default output format is 'html', but 'text' is also supported. Other custom output formats can be written.	No.

Returns

A string (html or raw text depending on "outputformat") revealing information about the target.

Description

This operator is an alias of the attribute() (page 1258) operator and is available in 4.7 and higher.

This operator extracts all available keys, attribute names and/or methods that belong to the input parameter (must be either an object, an array or a hash). By default, the array keys, object attribute names and their types will be revealed. By passing "show" as the first parameter, the operator will also return the values. The second parameter can be used to control the number of levels/children that should be expanded and included in the result (the default setting is 2). A large level value may cause the system to be trapped in a recursive/infinite loop. The returned result is an HTML table containing the retrieved information. If "text()" is passed as the third parameter, the output will be a plain string instead of an HTML table. Please refer to the "Array and object inspection" (page 205)section of the "Templates" chapter for more information about the use of this operator.

5

Examples

Example 1

```
{def $example=hash( 'Name',  'John Doe',
                    'Age',   24,
                    'Phone', '555-3212' )}

{$example|dump()}
```

The following output will be produced:

Attribute	Type
Name	string
Age	integer
Phone	string

Example 2

```
{def $example=hash( 'Name',  'Jane Doe',
                    'Age',   23,
                    'Phone', '555-3213' )}

{$example|dump( 'show' )}
```

The following output will be produced:

Attribute	Type	Value
Name	string	'Jane Doe'
Age	integer	23
Phone	string	'555-3213'

Example 3

```
{def $example=hash( 'Name',  'Jane Doe',
                    'Age',   25,
                    'Phone', '555-3213' )}

{$example|dump( 'something' )}
```

The following output will be produced:

5

Attribute	Type
Name	string
Age	integer
Phone	string

5

ezpackage

Summary

Not documented yet.

flag_icon

Summary

Outputs an image tag referencing a flag icon.

Usage

```
input|flag_icon()
```

Returns

A string containing an image tag.

Description

This operator generates an image tag that references a flag icon. The country code must be provided using the input parameter. This operator is frequently used by the administration interface.

Examples

Example 1

```
{'eng-GB'|class_icon()}
```

The following output will be produced:

```
<img src="/share/icons/flags/eng-GB.gif" alt="English (United Kingdom)" />
```

5

gettime

Summary

Converts a UNIX timestamp to a human friendly structure.

Usage

```
gettime( timestamp )
```

Parameters

Name	Type	Description	Required
timestamp	integer	A UNIX timestamp.	Yes.

Returns

An associative array (see below).

Description

This operator takes a UNIX timestamp as a parameter and returns an associative array that contains a human friendly representation of the provided timestamp. The following table shows the keys of the returned hash.

Key	Type	Description
seconds	integer	Number of seconds.
minutes	integer	Number of minutes.
hours	integer	Number of hours.
day	integer	The day of the month (1-31).
month	integer	The month number (1 is January).
year	integer	A four digit representation of the year (for example "1978").
weekday	integer	The day of the week (1-7).
weeknumber	string	The week of the year (1-52).
yearday	integer	The day of the year (1-365).
epoch	integer	The UNIX timestamp that was provided.

5

Examples

Example 1

```
{gettime( currentdate() )|attribute(show)}
```

The "currentdate" (page 1162) operator is used to generate the UNIX timestamp for the current date/time. This value is then fed to the "gettime" operator, which converts it to a human friendly structure. The array is viewed using the "attribute" (page 1258) operator. The following output will be produced:

Attribute	Type	Value
seconds	integer	11
minutes	integer	5
hours	integer	15
day	integer	13
month	integer	4
year	integer	2005
weekday	integer	4
weeknumber	string	15
yearday	integer	102
epoch	integer	1113397511

5

icon_info

Summary

Not documented yet.

makedate

Summary

Generates the UNIX timestamp of a given date.

Usage

```
makedate( [month [, day [, year [, dst ] ] ] ] )
```

Parameters

Name	Type	Description	Required
month	integer	The month of the year.	No.
day	integer	The day of the month.	No.
year	integer	The year.	No.
dst	integer	Daylight savings (on/off).	No.

Returns

A UNIX timestamp (an integer).

Description

This operator returns the UNIX timestamp corresponding to the provided parameters. The parameters may be left out in order from right to left. Parameters that are omitted will be set to the current value according to the local date and time. The "dst" parameter can be set to 1 if the time is during daylight savings time (DST), 0 if it is not, or -1 (the default) if it is unknown whether the time is within daylight savings time or not (the system will try to figure it out). If no parameters are given, the operator will return the timestamp for the current day.

Examples

Example 1

```
{makedate( 1, 2, 2005 )}
```

The following output will be produced: "1104620400" - which is the UNIX timestamp for "00:00:00, 2nd of January, 2005".

5

maketime

Summary

Generates the UNIX timestamp of a given date/time.

Usage

```
maketime( [hour [, minute [, second [, month [, day [, year [, dst ] ] ] ] ] ]
] )
```

Parameters

Name	Type	Description	Required
hour	integer	Hour of the day.	No.
minute	integer	Minute of the hour.	No.
second	integer	Second of the minute.	No.
month	integer	Month of the year.	No.
day	integer	Day of the month.	No.
year	integer	The year.	No.
dst	integer	Daylight savings (on/off).	No.

Returns

A UNIX timestamp (an integer).

Description

This operator returns the UNIX timestamp corresponding to the provided parameters. The parameters may be left out in order from right to left. Parameters that are omitted will be set to the current value according to the local date and time. The "dst" parameter can be set to 1 if the time is during daylight savings time (DST), 0 if it is not, or -1 (the default) if it is unknown whether the time is within daylight savings time or not (the system will try to figure it out). If no parameters are given, the operator will return the current timestamp.

Examples

Example 1

```
{maketime( 1, 2, 3, 4, 5, 2004 )}
```

The following output will be produced: "1081119723" - which is the UNIX timestamp for "01:02:03, 5th of April, 2004".

mimetype_icon

Summary

Outputs an image tag referencing a MIME type icon.

Usage

```
input|mimetype_icon( [ size, [, alt_text [, return_type ] ] ] )
```

Parameters

Name	Type	Description	Required
size	integer	The size of the icon (small, normal, large, etc.).	No.
alt_text	string	The alternative image text.	No.
return_url	boolean	The return type (FALSE=image tag, TRUE=address only).	No.

Returns

A string containing an image tag.

Description

This operator generates an image tag that references a MIME type icon. The name of the icon must be provided using the input parameter. The "size" and "alt_text" parameters are optional (see the description above). The operator uses the settings provided by the "icon.ini" configuration file (or an override).

Examples

Example 1

```
{'application/pdf'|mimetype_icon( 'small', 'Alternative image text' )}
```

This will output the MIME type icon that is associated with the ".pdf" file extension (defined in "icon.ini" or an override).

5

month_overview

Summary

Generates a structure that can be used to build a calendar.

Usage

```
input|month_overview( field,
                      today_timestamp,
                      hash( 'current',       current_timestamp,
                            'day_class',     day_class
                            'current_class', current_class,
                            'link',          link,
                            'month_link',    month_link,
                            'year_link',     year_link,
                            'day_link',      day_link,
                            'next',          hash( 'link', next_link ),
                            'previous',      hash( 'link', previous_link ) ) ) )
```

Parameters

Name	Type	Description	Required
field	string	The way the objects should be grouped.	Yes.
today_timestamp	integer	The UNIX timestamp for the day being searched.	Yes.
current_timestamp	integer	The UNIX timestamp of the current date/time.	No.
day_class	string	The CSS class that should be used for a normal day.	No.
current_class	string	The CSS class that should be used for the current day.	No.
link	string	Link to days containing objects ("content/view/full/node_id").	No.
month_link	string	Append "/month/MON" to links or not.	No.
year_link	string	Append "/year/YEAR" to links or not.	No.
day_link	string	Append "/day/DAY" to links or not.	No.
next_link	string	Link to the next month.	No.
previous_link	string	Link to the previous month.	No.

Returns

A complex structure (see below).

Description

This operator takes an array of content objects as input and analyzes and distributes them in a given month. It returns a complex structure (see below) that can be used by the "monthview.tpl" template, which will highlight days when at least one object has been published (see "design/ standard/templates/navigator/monthview.tpl"). The following structure will be returned:

```
.
|-- year
|-- month
|-- weekdays[]
|    |-- day
|    |-- class
|
|-- weeks[][]
|    |--day
|    |--link
|    |--class
|    |--highlight
|
|-- next
|    |--month
|    |--link
|
|-- previous
     |--month
     |--link
```

5

pdf

Summary

Provides access to the PDF functions (DEPRECATED).

Usage

```
{pdf( ... )}
```

Returns

The returned result depends on the actual PDF function.

Description

This operator provides access to the PDF functions that may be used to generate PDF data in a template. The actual PDF functions are documented in the "Template PDF functions" (page 1473) section of the "Reference" chapter.

5

roman

Summary

Generates a roman representation of a number.

Usage

```
input|roman( value )
```

Parameters

Name	Type	Description	Required
value	integer	A number that should be converted.	Only if the input parameter is omitted.

Returns

A string containing a roman number.

Description

This operator will convert either the input or the "value" parameter to a roman number. If both are provided, it is the "value" parameter that will be used.

Examples

Example 1

```
{8|roman()}
```

or

```
{roman( 8 )}
```

The following output will be produced: "VIII".

5

topmenu

Summary

Not documented yet.

5

treemenu

Summary

Fetches a subtree of nodes for the purpose of menu generation.

Usage

```
treemenu( path
         [, node_id           ]
         [, class_filter      ]
         [, depth_skip        ]
         [, max_level         ]
         [, is_selected_method ]
         [, indentation_level ]
         [, language          ] )
```

Parameters

Name	Type	Description	Required
path	array	An array of the path ($module_result.path).	Yes.
node_id	integer	DEPRECATED (The node ID number, i.e. the root of the subtree.)	No.
class_filter	array	An array of classes that should be filtered.	No.
depth_skip	integer	Number of levels that should be skipped.	No.
max_level	integer	The max depth that should be explored. (2 by default)	No.
is_selected_method	string	Sets whether "is_selected=TRUE" should be assigned to the parents of the current node as well.	No.
indentation_level	integer	The size of increment to use when calculating the indentation of the menu entries (15 by default).	No.
language	string/ array	The language(s) to use when fetching the nodes. Use either a string, e.g. 'nor-NO', or an array of strings, e.g. array('nor-NO', 'eng-GB').	No.

Returns

A complex structure that can be used to build a menu (see below).

5

Description

This operator fetches a subtree of nodes and returns a complex structure (an array of hashes) that can be used to generate a menu. Note that the optional "node_id" parameter is deprecated and no longer used (its value is ignored by the system, but the parameter itself is not removed because of backwards compatibility reasons). The root of the subtree is determined by the "path" parameter. If the provided path array points to a node, then a subtree of this node will be fetched. (Use "$module_result.path" to pass the path which leads to the page that is currently being viewed.) If the "path" parameter does not point to a node in the content tree, then a subtree of the "Content" top level node (page 125) will be fetched.

If the optional "class_filter" parameter is omitted, all nodes will be fetched without filtering. If an empty array is passed, then only folder nodes (class ID = 1) will be fetched.

The optional "language" parameter makes it possible to specify which languages to use when fetching the nodes. The languages will be prioritized according to the array (this overrides the language settings of the siteaccess).

The default value of the "is_selected_method" parameter is "tree". This value determines that the current node and its parent nodes will be considered as selected (is_selected=TRUE). If this parameter is set to "node", only the current node will be considered as selected.

The following table shows the hash-structure for each element in the array that will be returned.

Key	Type	Description
id	integer	The ID of the node.
level	integer	The depth of the node.
url_alias	string	The URL alias of the node.
url	string	The system URL of the node.
text	string	The name of the node.
is_selected	boolean	TRUE if the node is currently being viewed/selected, FALSE otherwise.

If the "path" parameter points to a node in the content tree, then the following additional keys will be included in the hash structure:

Key	Type	Description
data_map	array	The attributes (as ez-contentobjectattribute (page 1042) objects) of the actual content object encapsulated by the node.
class_name	string	The name of the class which the object encapsulated by the node is an instance of, for example "Folder".
is_main_node	boolean	TRUE if the node is a main

		node, FALSE otherwise.
has_children	boolean	TRUE if the node has children, FALSE otherwise.
indent	integer	The indentation of the menu entries. This value is calculated according to the depth of the node using "indentation_level" as increment. A menu item of the level N will have indent=(N-1)*indentation_level.

Examples

Example 1 (explanatory)

Let's use a small site with the following content structure (see the screenshot) for the purpose of demonstration

(see figure 5.98)
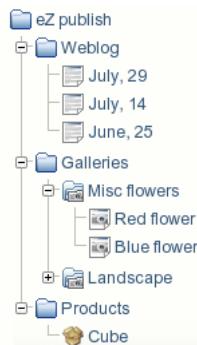
Figure 5.98: *The content tree*

and insert the following code into "pagelayout.tpl":

```
{def $mainMenu=treemenu( $module_result.path ) }
{foreach $mainMenu as $menu}
{$menu.level} - {$menu.text}<br />
{/foreach}
{undef $mainMenu}
```

If the "Weblog" node is being viewed, then the following output will be produced:

5

```
0 - Weblog
1 - July, 29
1 - July, 14
1 - June, 25
0 - Galleries
0 - Products
```

If the "Blue flower" node is being viewed, then the following output will be produced:

```
0 - Weblog
0 - Galleries
1 - Misc flowers
1 - Landscape
0 - Products
```

Since the "max_level" parameter is omitted, only two levels are explored. To set the "max_level" parameter to 3, change the first line of the previous code fragment in the following way:

```
{def $mainMenu=treemenu( $module_result.path, , , , 3 ) }
```

This will produce the following output for the "Blue flower" node:

```
0 - Weblog
0 - Galleries
1 - Misc flowers
2 - Red flower
2 - Blue flower
1 - Landscape
0 - Products
```

To skip the first level, set the "depth_skip" parameter to 1 by changing the first line of the code fragment as shown below:

```
{def $mainMenu=treemenu( $module_result.path, , , 1, 3 ) }
```

If the "Blue flower" node is being viewed, then the following output will be produced:

```
0 - Misc flowers
1 - Red flower
1 - Blue flower
0 - Landscape
```

Now, let's use another code fragment in order to see which items are selected:

5

```
{def $mainMenu=treemenu( $module_result.path, , , , 3 )}
{foreach $mainMenu as $menu}
    {if $menu.is_selected}
        {$menu.level} - {$menu.text} (selected) <br />
    {else}
        {$menu.level} - {$menu.text} <br />
    {/if}
{/foreach}
{undef $mainMenu}
```

Since the "is_selected_method" parameter is omitted, the "tree" mode will be used and the follow-
ing output will be produced for the "Blue flower" node:

```
0 - Weblog
0 - Galleries (selected)
1 - Misc flowers (selected)
2 - Red flower
2 - Blue flower (selected)
1 - Landscape
0 - Products
```

To set the "is_selected_method" parameter to "node", replace the first line of the last code fragment
by the following line:

```
{def $mainMenu=treemenu( $module_result.path, , , , 3, 'node' )}
```

If the "Blue flower" node is being viewed, then the following output will be produced:

```
0 - Weblog
0 - Galleries
1 - Misc flowers
2 - Red flower
2 - Blue flower (selected)
1 - Landscape
0 - Products
```

Example 2

These examples are from pagelayout.tpl.

Make a menu of folder and info_page classes. Skip the first level and maximum go to depth 6.

```
<ul>
{def $mainMenu=treemenu( $module_result.path, ,
                                array('folder','info_page'), 1, 6 )}
```

```
{foreach $mainMenu as $menu}
    <li class="level_{$menu.level}">

    {if $menu.is_selected}
        <div class="selected">
        <a href={$menu.url_alias|ezurl}>{$menu.text}</a>
        </div>
    {else}
        <a href={$menu.url_alias|ezurl}>{$menu.text}</a>
    {/if}

    </li>
{/foreach}
</ul>
```

Make a menu which shows the sub menu items when clicked on the parent menu item. Notice
that only the objectclass ids: 1, 9, and 17 are visible.

```
{def $docs=treemenu( $module_result.path, ,
                                   array(1, 9, 17), 0, 4)}
{def $depth=1 $last=0}

{foreach $docs as $menu}
    {if and($last | ne(0), $last.level|gt($menu.level))}
        </ul>
        </li>
    {/if}

    <li>

    {if and($last | ne(0), $last.level| lt($menu.level))}
        <ul>
        <li>
    {/if}

    <a {$menu.is_selected|choose('','class="selected"')}
        href={$menu.url_alias|ezurl}>{$menu.text|shorten(25)}</a>
    </li>

    {set last=$menu}
{/foreach}

{while $depth |gt(1)}
    </li>
    </ul>
```

```
    {set depth=$depth|sub(1)}
{/while}
```

5.8.8 Strings

\begin{description} \item[append]\refer{label0b664e1d1ed41934590b2e1433ed1ca2}\
\\ Returns the input string with a custom sequence appended to it.
\item[autolink]\refer{label4f72f9df3e83b421a1f217707f5f900e}\ \\ Returns
the input string with the addresses replaced by link tags. \item[begins_
with]\refer{labelec5a5218af5764bb89e176906be47459}\ \\ Checks if a string starts with
a specific character/sequence. \item[break]\refer{labelfaa6a97752cd743298dcf0f60a3f34ef}\
\\ Returns the input string with all newlines converted to HTML
breaks. \item[chr]\refer{labelff67c5baa934536e6a274ef2cca42153}\ \\ Gen-
erates a string based on the input array of ASCII/UNICODE values.
\item[compare]\refer{label16215ec53982e10ce30b7973fa338e6b}\ \\ Compares the contents
of two strings. \item[concat]\refer{labelc36ba3da27ffeec7ec67c3d04cc2b5db}\ \\ Merges sev-
eral strings into one string. \item[contains]\refer{label6b6bdabe7bebf9aaed07500d551089fa}\
\\ Checks if a string contains a specific element. \item[count_
chars]\refer{label790d0810c81d110dde678373b56f1345}\ \\ Returns the length of the
input string. \item[count_words]\refer{labeld041d9c4eb32822d16d851ba05c64d7c}\
\\ Returns the number of words that make up the input
string. \item[crc32]\refer{label0a5fe8eb3104db7eeb3bb8d168267b4b}\
\\ Returns the CRC32 polynomial of the input string.
\item[downcase]\refer{label224fd9937ac54b46492f1e895e9e2355}\ \\ Re-
turns a lowercased version of the input string. \item[ends_
with]\refer{label71e45cf72b664573c6f1e9bb6772229b}\ \\ Checks if a string ends with a
specific character/sequence. \item[explode]\refer{labelf54f4549dfc213a21a6724c91eddd2f8}\
\\ Splits the input string and returns it as an array of
strings. \item[extract]\refer{label6cc093a9b44d3196997b6b16c07b2759}\
\\ Returns a portion of the input string. \item[extract_
left]\refer{labelee9ec81254b4b675d69843db45b3d5f6}\ \\ Returns a portion of the start of
the input string. \item[extract_right]\refer{labelb55b6b86967611d8e4d978b04a44f4f0}\
\\ Returns a portion of the end of the input string.
\item[indent]\refer{labela71f0329e4d72304ec3240ecb9832eba}\ \\ Returns an indented
version of the input string. \item[insert]\refer{labeleeebad0f3a2254da994398865b78160f}\
\\ Returns the input string with additional text inserted at a specified position.
\item[md5]\refer{label822e9f907a160c5b74fc673f8ad2c3fb}\ \\ Returns the MD5 hash
of the input string. \item[nl2br]\refer{label81a85e421f993f650a9f84f599b9d855}\
\\ Returns the input string with all newlines converted to HTML
breaks. \item[ord]\refer{labelddfdf5b2346a64bb477ae1fdb76cba76}\ \\ Re-
turns an array containing the ASCII/UNICODE values of the input string.
\item[pad]\refer{labelcdbd9bf5af92ba07ac0fa9a5357d4858}\ \\ Returns a lengthened version
of the input string. \item[prepend]\refer{label2621dce3121f7e34459f0163ce488953}\
\\ Returns the input string prepended with a custom sequence.
\item[remove]\refer{labelf32199f1c490b4999603b8c7fdf489e6}\ \\ Returns a pruned version
of the input string. \item[repeat]\refer{labelda423d3018a79d060034ba4cc36949df}\ \\ Re-
turns a repeated version of the input string. \item[replace]\refer{label2d7be6dbb78ea109064b233f4dc4733
\\ Replaces elements in a string. \item[reverse]\refer{labelf1574fa13c86b0b4ab47de65a40e6e12}\
\\ Returns a reversed version of the input string. \item[rot13]\refer{labelf6fdc4e313622718f318e22215541

5

\\   Returns   a   ROT13   transformation   of   the   input   string.
\item[shorten]\refer{labelbfac4b5e2483aff3461fab2916c219b4}\ \\ Returns a shortened version of the input string. \item[simpletags]\refer{label29da39f8dd954ce3218b33e255b8a4c1}\
\\   Returns   a   partially   marked   up   version   of   the   input   string.
\item[simplify]\refer{label7be6201e2512f0d282e9cbd1fedcbf91}\ \\ Returns a simplified version of the input string. \item[trim]\refer{label522b0b5ffb0eab246fbcbdc3e9107df7}\ \\ Returns a stripped version of the input string. \item[upcase]\refer{label85e433f5fd74bc9e900831ceed0eed8e}
\\ Returns a capitalized version of the input string. \item[upfirst]\refer{label05a95561481ac1cb03d1b324d
\\   Returns   the   input   string   with   a   capitalized   initial   letter.   \item[upword]\refer{label694b6acbd2f4dd71716c05edc657f8ed}\
\\   Returns   the   input   string   with   capitalized   initial   letters.
\item[wash]\refer{label2f4d19fe94692c0ae9f4360ffce56a21}\ \\ Returns an HTML-safe version of the input string. \item[wordtoimage]\refer{label7bf484d1daa6c98eeb0f91dbe33d218d}\
\\   Returns   the   input   string   with   embedded   image   tags.
\item[wrap]\refer{labelbcd43f1e1d6a649c403db18ec3ea9fc2}\ \\ Returns a wrapped version of the input string.

\end{description}

append

Summary

Returns the input string with a custom sequence appended to it.

Usage

```
input|append( value1 [, value2 [, ... ] ] )
```

Parameters

Name	Type	Description	Required
value1	mixed	Text to be appended.	Yes.
value2	mixed	More text that should be appended.	No.

Returns

A string consisting of the input string and the parameters.

Description

This operator appends the parameter value(s) at the end of the input string and returns the resulting string.

Examples

Example 1

```
{'The '|append( 'Last ', 'Crusade ' )}
```

The following output will be produced: "The Last Crusade".

5

autolink

Summary

Returns the input string with the addresses replaced by link tags.

Usage

```
input|autolink()
```

Returns

A string with marked up links.

Description

This operator takes a string as an input. It will convert links and E-mail addresses to clickable entities using HTML markup. The resulting string (containing markup here and there) will be returned.

Examples

Example 1

```
{'Blah blah http://www.example.com blah hello@example.com blah.'|autolink()}
```

The following output will be produced:

```
Blah blah <a href="http://www.example.com">http://www.example.com</
a> blah <a href="mailto:hello@example.com">hello@example.com</a> blah.
```

5

begins_with

Summary

Checks if a string starts with a specific character/sequence.

Usage

```
input|begins_with( sequence )
```

Parameters

Name	Type	Description	Required
sequence	string	The string that should be matched.	Yes.

Returns

TRUE if there is a match, FALSE otherwise.

Description

This operator checks if the input string starts with a specified sequence of characters. If yes, the operator returns TRUE, otherwise FALSE will be returned.

Examples

Example 1

```
{'My cat is green.'|begins_with( 'My cat' )}
```

Returns TRUE.

Example 2

```
{'My cat is green.'|begins_with( 'My dog' )}
```

Returns FALSE.

5

break

Summary

Returns the input string with all newlines converted to HTML breaks.

Usage

```
input|break()
```

Returns

A string with HTML breaks.

Description

This operator takes a string as input. It does not replace newline characters/sequences, but it does insert HTML break tags (
) before all newlines in a string and returns a modified version of the input.

Examples

Example 1

```
{'The lazy
cat
jumps over
the quick rat.'|break()}
```

The following output will be produced:

```
The lazy<br />cat<br />jumps over<br />the quick rat.
```

5

chr

Summary

Generates a string based on the input array of ASCII/UNICODE values.

Usage

```
input|chr()
```

Returns

A string containing the requested characters.

Description

This operator creates and returns a string. The contents of the returned string is determined by the input parameter which must be an array of ASCII/UNICODE values (as integers).

Examples

Example 1

```
{array( 97, 98, 99 )|chr()}
```

The following output will be produced: "abc".

5

compare

Summary

Compares the contents of two strings.

Usage

```
input|compare( compare_with )
```

Returns

TRUE if the strings match, FALSE otherwise.

Description

This operator compares the contents of two strings and returns TRUE if they're identical, FALSE if they differ.

Examples

Example 1

```
{'Hello world'|compare( 'Hello world' )}
```

Returns TRUE.

Example 2

```
{'Hello world'|compare( 'Goodbye world' )}
```

Returns FALSE.

5

concat

Summary

Merges several strings into one string.

Usage

```
concat( value1, value2 [,...] )
```

Parameters

Name	Type	Description	Required
value1	mixed	A string that should be added.	Yes.
value2	mixed	Another string that should be added.	Yes.

Returns

A string consisting of all the parameters.

Description

This operator merges several strings into one and returns the resulting string.

Examples

Example 1

```
{concat( 'what', 'ever' )}
```

The following output will be produced: "whatever".

Example 2

```
{def $number=256}
{concat( 'The number is: ', $number, '!' )}
```

The following output will be produced: "The number is: 256!"

contains

Summary

Checks if a string contains a specific element.

Usage

```
input|contains( sequence )
```

Parameters

Name	Type	Description	Required
sequence	string	The string that should be matched.	Yes.

Returns

TRUE if there is a match, FALSE otherwise.

Description

This operator checks if the input string contains a specific sequence of characters. If a match is found, the operator will return TRUE, otherwise FALSE will be returned.

Examples

Example 1

```
{'Welcome to my homepage!'|contains( 'my' )}
```

Returns TRUE.

Example 2

```
{'Welcome to my homepage!'|contains( 'your' )}
```

Returns FALSE.

count_chars

Summary

Returns the length of the input string.

Usage

```
input|count_chars()
```

Returns

An integer revealing the string length.

Description

This operator counts and returns the number of characters (all of them, whitespaces included) that make up the input string.

Examples

Example 1

```
{'Testing 1 2 3'|count_chars()}
```

The following output will be returned: "13".

Example 2

```
{'Testing'|count_chars()}
```

The following output will be returned: "7".

5

count_words

Summary

Returns the number of words that make up the input string.

Usage

```
input|count_words()
```

Returns

An integer revealing the number of words that make up a string.

Description

This operator counts and returns the number of words that are found within the input string.

Examples

Example 1

```
{'Where do you want to publish today?'|count_words()}
```

The following output will be returned: "7".

5

crc32

Summary

Returns the CRC32 polynomial of the input string.

Usage

```
input|crc32()
```

Returns

The CRC32 polynomial of the input string.

Description

This operator calculates and returns the CRC32 polynomial of the input string.

Examples

Example 1

```
{'filename.txt'|crc32()}
```

The following output will be produced: ”-460339180”.

5

downcase

Summary

Returns a lowercased version of the input string.

Usage

```
input|downcase()
```

Returns

A lowercase version of the input string.

Description

This operator returns a lowercased version of the input string.

Examples

Example 1

```
{"My StriNG Is CoOl!"|downcase()}
```

The following output will be produced: "my string is cool!".

5

ends_with

Summary

Checks if a string ends with a specific character/sequence.

Usage

```
input|ends_with( sequence )
```

Parameters

Name	Type	Description	Required
sequence	string	The string that should be matched.	Yes.

Returns

TRUE if there is a match, FALSE otherwise.

Description

This operator checks if the input string ends with a specified sequence of characters. If yes, the operator returns TRUE, otherwise FALSE will be returned.

Examples

Example 1

```
{'Linux is great!'|ends_with( 'great!' )}
```

Returns TRUE.

Example 2

```
{'Linux is great!'|begins_with( 'Linux' )}
```

Returns FALSE.

5

explode

Summary

Splits the input string and returns it as an array of strings.

Usage

```
input|explode( separator )
```

Parameters

Name	Type	Description	Required
separator	string	Split sequence.	Yes.

Returns

An array of strings.

Description

This operator takes a string as input and returns an array of strings. Each element in the array will be a part of the input string extracted on the basis of the specified sequence of split characters.

Examples

Example 1

```
{'All-your-base-are-belong-to-us!'|explode( '-' )}
```

The following array will be returned: ('All', 'your', 'base', 'are', 'belong', 'to', 'us!').

5

extract

Summary

Returns a portion of the input string.

Usage

```
input|extract( offset [, length] )
```

Parameters

Name	Type	Description	Required
offset	integer	The offset to start at.	Yes.
length	integer	The number of characters that should be extracted.	No.

Returns

A string containing a portion of the input string.

Description

This operator will return a portion of the input string. The returned portion must be defined by the "offset" and "length" parameters. If the "length" parameter is omitted, the rest of the string (from offset) will be returned.

Examples

Example 1

```
{'I love monday mornings!'|extract( 7 )}
```

The following output will be produced: "monday mornings!".

Example 2

```
{'Big apples.'|extract( 4, 5 )}
```

The following output will be produced: "apple".

extract_left

Summary

Returns a portion of the start of the input string.

Usage

```
$input_string|extract_left( length )
```

Parameters

Name	Type	Description	Required
length	integer	The number of characters that should be extracted.	Yes.

Returns

A string containing a chunk of the input string.

Description

This operator extracts a portion from the start of the input string. The "length" parameter must be used to define the length (number of characters) of the portion.

Examples

Example 1

```
{'Gooooood morning Vietnam!'|extract_left( 8 )}
```

The following output will be produced: "Gooooood".

5

extract_right

Summary

Returns a portion of the end of the input string.

Usage

```
input|extract_right( length )
```

Parameters

Name	Type	Description	Required
length	integer	The number of characters that should be extracted.	Yes.

Returns

A string containing a chunk of the input string.

Description

This operator extracts a portion from the end of the input string. The "length" parameter must be used to define the length (number of characters) of the portion.

Examples

Example 1

```
{"Gooooood morning Vietnam!"|extract_right( 8 )}
```

The following output will be produced: "Vietnam!".

5

indent

Summary

Returns an indented version of the input string.

Usage

```
input|indent( count [, type [, filler ] ] )
```

Parameters

Name	Type	Description	Required
count	integer	Number of indentations.	Yes.
type	string	Type of indentation ("space", "tab" or "custom").	No.
filler	string	Custom indentation.	No.

Returns

An indented version of the input string.

Description

This operator indents the input string and returns it. The indentation type can be set to "space", "tab" or "custom". The default indentation is "space". If the indentation type is set to "custom", the "filler" parameter must be set to the desired indentation string.

Examples

Example 1

```
{'This is my text'|indent( 1 )}
```

The following string will be returned: " This is my text".

Example 2

```
{'This is my second line'|indent( 3, 'custom', '.' )}
```

The following string will be returned: "...This is my second line".

insert

Summary

Returns the input string with additional text inserted at a specified position.

Usage

```
input|insert( offset, sequence )
```

Parameters

Name	Type	Description	Required
offset	integer	The position where the sequence should be inserted.	Yes.
sequence	string	The string that should be inserted.	Yes.

Returns

A string consisting of the input and the inserted sequence.

Description

This operator inserts a sequence of characters at a specified position of the input string. The resulting string will be returned.

Examples

Example 1

```
{'My string is simple.'|insert( 3, 'static ' )}
```

The following output will be produced: "My static string is simple.".

5

md5

Summary

Returns the MD5 hash of the input string.

Usage

```
input|md5()
```

Returns

The MD5 hash of the input string.

Description

This operator calculates and returns the MD5 hash of the input string.

Examples

Example 1

```
{'Desktop computer'|md5()}
```

The following output will be produced: "8cae7108f44d1958b9febc65e44cbbc8".

5

nl2br

Summary

Returns the input string with all newlines converted to HTML breaks.

Usage

```
input|nl2br()
```

Returns

A string with HTML breaks.

Description

This operator takes a string as input. It replaces newline characters/sequences with HTML break tags and returns a modified version of the input.

Examples

Example 1

```
{'The lazy
cat
jumps over
the quick rat.'|nl2br()}
```

The following output will be produced:

```
The lazy<br />cat<br />jumps over<br />the quick rat.
```

5

ord

Summary

Returns an array containing the ASCII/UNICODE values of the input string.

Usage

```
input|ord()
```

Returns

An array with ASCII/UNICODE values.

Description

This operator returns an array containing the ASCII/UNICODE values of the characters that make up the input string.

Examples

Example 1

```
{'abcdef'|ord()}
```

The following array will be returned: (97, 98, 99, 100, 101, 102).

5

pad

Summary

Returns a lengthened version of the input string.

Usage

```
input|pad( length [, padding ] )
```

Parameters

Name	Type	Description	Required
length	integer	The desired length of the string.	Yes.
padding	character	Custom character to be used for padding.	No.

Returns

A padded version of the input string.

Description

This operator makes sure that the input string is at least "length" characters long by inserting extra characters at the end. It is possible to specify a custom character using the "padding" parameter (default is space). The operator returns a padded version of the input string.

Examples

Example 1

```
{'Too short!'|pad( 15 )}
```

The following string will be produced: "Too short! ".

Example 2

```
{'Too short!'|pad( 16, '-' )}
```

The following string will be produced: "Too short!-----".

prepend

Summary

Returns the input string prepended with a custom sequence.

Usage

```
input|prepend( sequence )
```

Parameters

Name	Type	Description	Required
sequence	string	The string that should be prepended.	Yes.

Returns

A string consisting of the first parameter and the input string.

Description

This operator puts the "sequence" parameter at the start of the input string and returns the resulting string.

Examples

Example 1

```
{'Weaver'|prepend( 'Sigourney ')}
```

The following string will be produced: "Sigourney Weaver".

5

remove

Summary

Returns a pruned version of the input string.

Usage

```
input|remove( offset, length )
```

Parameters

Name	Type	Description	Required
offset	integer	The offset to start at.	Yes.
length	integer	The number of characters that should be removed.	Yes.

Returns

A pruned version of the input string.

Description

This remove operator removes characters from the input string and returns the pruned version. The "offset" and "length" parameters must be used to define the start and length of the portion that should be removed.

Examples

Example 1

```
'My string is simple.'|remove( 3, 2 )
```

The following string will be produced: "My ring is simple.".

5

repeat

Summary

Returns a repeated version of the input string.

Usage

```
input|repeat( count )
```

Parameters

Name	Type	Description	Required
count	integer	The number of repeats.	Yes.

Returns

A repeated version of the input string.

Description

This operator returns a repeated version of the input string. The "count" parameter must be used to define the desired number of repetitions.

Examples

Example 1

```
{'*DJ Cat Show*'|repeat( 2 )}
```

The following string will be produced: "*DJ Cat Show* *DJ Cat Show* ".

5

replace

Summary

Replaces elements in a string.

Usage

```
input|replace( offset [, length [, sequence1 [, sequence2 [, ...]]] ] )
```

Parameters

Name	Type	Description	Required
offset	integer	The offset to start replacing elements.	Yes
length	integer	The number of characters that should be replaced.	No
sequence1	any	The sequence that will replace characters in input string.	No
sequence2	any	The sequence that will replace characters in input string.	No

Returns

A replaced version of the input string.

Description

This operator replaces character(s) from the input string and a replaced version of input string will be returned. The "offset" parameter must be used to define the start of the portion that should be replaced. The "length" parameter can be used to define the number of characters that should be replaced. If length is omitted, the sequence(s) will be placed after the offset(like 'insert' operator).

Examples

Example 1

```
{'12345'|replace( 1, 2 )}
```

The following string will be returned: '145'.

Example 2

5

```
{'12345'|replace( 1 )}
```

The following array will be returned: '12345' .

Example 3

```
{'12345'|replace( 1, 2, 'ab' )}
```

The following array will be returned: '1ab45' .

Example 4

```
{{'12345'|replace( 1, 2, 'a', 'b', 'c' )}
```

The following array will be returned: '1abc45' .

Example 5

```
{'12345'|replace( 1, , 'abc' )}
```

The following array will be returned: '1abc2345' .

5

reverse

Summary

Returns a reversed version of the input string.

Usage

```
input|reverse()
```

Returns

A reversed version of the input string.

Description

This operator returns a reversed version of an input string.

Examples

Example

```
{"Hello World"|reverse}
```

The following output will be produced: "dlroW olleH".

5

rot13

Summary

Returns a ROT13 transformation of the input string.

Usage

```
input|rot13()
```

Returns

A rotated version of the input string.

Description

Returns a ROT13 transformation of the input string.

Examples

Example 1

```
{'Hello World'|rot13()}
```

The following output will be produced: "Uryyb Jbeyq".

Example 2

```
{'Uryyb Jbeyq'|rot13()}
```

The following output will be produced: "Hello world".

5

shorten

Summary

Returns a shortened version of the input string.

Usage

```
input|shorten( [ length [, sequence [, trim_type ] ] ] )
```

Parameters

Name	Type	Description	Required
length	integer	The desired length of the returned string.	No.
sequence	string	Custom trailing/end-sequence.	No.
trim_type	string	Controls the type of trimming: "right" (default) or "middle".	No.

Returns

A shortened version of the input string.

Description

This operator shortens the input string to "length" characters and adds a trailing sequence. Please note that the "length" parameter also includes the length of the trailing sequence. If the input string is shorter than "length", it will not be shortened. The default length is 80, the default trailing sequence is three dots: "...". The third parameter controls the type of trimming, it can be set to either "right" (default) or "middle".

Examples

Example 1

```
{'Led Zeppelin rocks!'|shorten( 15 )}
```

The following output will be produced: "Led Zeppelin...".

Example 2

```
{"eZ Systems"|shorten( 7, '...' , 'middle' )}
```

The following output will be produced: "eZ...ms".

simpletags

Summary

Returns a partially marked up version of the input string.

Usage

```
input|simpletags( [ taglist ] )
```

Parameters

Name	Type	Description	Required
taglist	string	The name of the custom tag group that should be used.	No.

Returns

A partially marked up version of the input string.

Description

This operator returns a partially marked up version of the input string. It can be used to allow the usage/pass-through of a small subset of HTML/custom tags (other/disallowed tags will be removed). This operator is typically useful when it comes to allowing some kind of formatting in for example comments (instances of a class that does not support formatting through the XML block datatype). The optional "taglist" parameter can be used to select the list of allowed tags (the default is "TagList"). The tags/groups must be defined in a configuration override for "template.ini". The following table shows the tags that are allowed by default.

Custom tag	HTML replacement
literal	
	`<pre>...</pre>`
code	
	`<pre class="code">...</pre>`
strong	
	`<b>...</b>`
emphasize	
	`<i>...</i>`

5

Examples

Example 1

```
{'Back To <strong>The</strong> Future'|simpletags()}
```

The following output will be produced:

```
Back To <b>The</b> Future
```

5

simplify

Summary

Returns a simplified version of the input string.

Usage

```
input|simplify( [char] )
```

Parameters

Name	Type	Description	Required
char	character	The character that should be simplified.	No.

Returns

A simplified version of the input string.

Description

This operator takes a string as the input parameter. It transforms multiple consecutive characters into one. The operator can be used to remove the duplicates as it leaves only a single copy of each character. It is possible to specify only one character that should be simplified, this can be done using the optional "char" parameter. By default, the operator removes multiple spaces. Special characters must be specified using regular expression style, please refer to the table below.

Character	Description
\t	Tab (HT, TAB)
\n	Newline (LF, NL)
\r	Return (CR)
\f	Form feed (FF)
\a	Alarm / bell (BEL)
\e	Escape / think troff (ESC)

Examples

Example 1

```
{'We    don't    need   no whitespaces!'|simplify()}
```

The following output will be produced: "We don't need no whitespaces!".

Example 2

```
{'This____string__is___annoying.'|simplify( '_' )}
```

The following output will be produced: "This_string_is_annoying.".

5

trim

Summary

Returns a stripped version of the input string.

Usage

```
input|trim( [char_list] )
```

Parameters

Name	Type	Description	Required
char_list	string	Characters that should be removed.	No.

Returns

A stripped version of the input string.

Description

This operator removes characters from the beginning and the end of the input string. By default, it will get rid of the following characters:

Character	ASCII value (dec)	ASCII value (hex)	Description
	32	0x20	An ordinary space.
\t	9	0x09	A tab.
\n	10	0x0A	A new line (line feed).
\r	13	0x0D	A carriage return.
\0	0	0x00	The NUL-byte.
\x0B	11	0x0B	A vertical tab.

Examples

Example 1

```
{'   Gizmo is not a gremlin.   '|trim()}
```

The following output will be produced: "Gizmo is not a gremlin."

5

upcase

Summary

Returns a capitalized version of the input string.

Usage

```
input|upcase()
```

Returns

A capitalized version of the input string.

Description

This operator returns a capitalized version of the input string.

Examples

Example 3

```
{'This is my string.'|upcase()}
```

The following output will be produced: "THIS IS MY STRING.".

5

upfirst

Summary

Returns the input string with a capitalized initial letter.

Usage

```
input|upfirst()
```

Returns

The input string with a capitalized initial letter.

Description

This operator converts the first character of the input string to a capital letter. The resulting string is returned.

Examples

Example 1

```
{'good bye!'|upfirst()}
```

The following string will be returned: "Good bye!".

5

upword

Summary

Returns the input string with capitalized initial letters.

Usage

```
input|upword()
```

Returns

The input string with capitalized initial letters.

Description

This operator returns the input string with capitalized initial letters.

Examples

Example 3

```
{'good bye lenin!'|upword()}
```

The following output will be produced: "Good Bye Lenin!".

5

wash

Summary

Returns an HTML-safe version of the input string.

Usage

```
input|wash( [type] )
```

Parameters

Name	Type	Description	Required
type	string	The type of text that should be washed.	No.

Returns

An HTML-safe version of the input string.

Description

This operator translates the input string into an HTML friendly version. It will take care of converting bogus characters to HTML-friendly replacements. The "type" parameter can be used to specify the washing type, it can be set to either "xhtml" or "email" (the default is "xhtml"). E-mail washing can be controlled using the setting of the [WashSettings] configuration block of "template.ini". All strings that may break the HTML should always be washed using this operator.

Examples

Example 1

```
{'Bogus & stuff &lt;'|wash()}
The following output will be produced: "Bogus &amp; stuff &lt;".
```

Example 2

```
{'hello@example.com'|wash( 'email' )}
```

The following output will be produced:

```
hello<span class="spamfilter">SPAMFILTER</span>@example.com
```

5

Example 3

```
{'hello@example.com'|wash( 'email' )}
```

If a configuration override for "template.ini" exists and contains...

```
[WashSettings]
EmailDotText=[dot]
EmailAtText=[at]
```

...the following output will be produced: "hello[at]example[dot]com".

5

wordtoimage

Summary

Returns the input string with embedded image tags.

Usage

```
input|wordtoimage()
```

Returns

The input string with embedded image tags.

Description

This operator looks for special character sequences in the input string. When a match is found, it will be replaced by an image tag. For example, the sequence ":-)" will be replaced by a small image of a smiling face. The character sequences that should be replaced and the images that should be used are defined in the "wordtoimage.ini" configuration file.

Examples

Example 1

```
{'No problemo... :-)'|wordtoimage()}
```

This would return the input string where the ":-)" sequence would be replaced by an image tag referencing a small image of a smiling face.

5

wrap

Summary

Returns a wrapped version of the input string.

Usage

```
input|wrap( [ width [, break_sequence [, cut ] ] ] )
```

Parameters

Name	Type	Description	Required
width	integer	The width at which the text should be wrapped.	No.
break_sequence	string	A custom break/newline sequence.	No.
cut	boolean	TRUE (force wrap) or FALSE (do not force wrap).	No.

Returns

A wrapped version of the input string.

Description

This operator returns a wrapped version of the input string. The string will be wrapped at either the default width (80 characters) or at a width specified using the optional "width" parameter. It inserts newline characters ("\n") or a character/sequence which is specified using the optional "break_sequence" parameter. The "cut" parameter can be set to either TRUE or FALSE - it controls whether the string should always be wrapped at the specified width or not (a word that is larger than the desired width, it will be broken apart).

Examples

Example 1

```
{'Hello world'|wrap( 5 )}
```

The following output will be produced:

Hello
world

5.8.9   URLs

exturl   (page 1337)
    Not documented yet.

ezdesign   (page 1338)
    Returns the input string prepended with the current design directory.

ezimage   (page 1339)
    Returns the input string prepended with the current image directory.

ezroot   (page 1341)
    Same as "ezurl" without "index.php" and the siteaccess name in the returned address.

ezurl   (page 1342)
    Returns a working version of an eZ Publish URL (provided as input).

5

exturl

Summary

Not documented yet.

ezdesign

Summary

Returns the input string prepended with the current design directory.

Usage

```
input|ezdesign( [ quote ] )
```

Parameters

Name	Type	Description	Required
quote	string	Quote style: "no", "single" or "double" (default).	No.

Returns

The input string prepended with the current image directory.

Description

This operator returns the input string prepended with the current design directory. If the operator is unable to find the specified file within the current design, it will attempt to locate it in the fallback designs or the standard design. The "ezdesign" operator should always be used when a design related file is included in a template. It will make sure that the path to the file is always correct, regardless of the location of the eZ Publish directory, the access method, the environment, and so on.

By default, this operator returns a double-quoted string. The optional "quote" parameter can be used to control the way the address is returned: "no" (no quotes), "single" (single quotes) or "double" (double quotes, the default). Dropping quotes is useful when specifying CSS files in the following way:

```
<style type="text/css">
    @import url({'stylesheets/core.css'|ezdesign( 'no' )});
    @import url({'stylesheets/ezmain.css'|ezdesign( 'no' )});
    @import url({'stylesheets/ezsystems.css'|ezdesign( 'no' )});
</style>
```

5

ezimage

Summary

Returns the input string prepended with the current image directory.

Usage

```
input|ezimage( [ quote [, slash_skip ] ] )
```

Parameters

Name	Type	Description	Required
quote	string	Quote style: "no", "single" or "double" (default).	No.
skip_slash	boolean	Include (FALSE, default) or skip (TRUE) the first slash.	No.

Returns

The input string prepended with the current image directory.

Description

This operator prepends the input string with the location of the image directory used by the current design. If the operator is unable to find the specified file within the "images" subdirectory of the current design, it will attempt to locate it in the "images" subdirectory of the fallback designs or the standard design. The "ezimage" operator should always be used when an image is included in a template. It will make sure that the path to the image is always correct, regardless of the location of the eZ Publish directory, the access method, the environment, and so on.

By default, this operator returns a double-quoted string. The optional "quote" parameter can be used to control the way the address is returned: "no" (no quotes), "single" (single quotes) or "double" (double quotes, the default). The optional "skip_slash" parameter can be used to get rid of the first slash within the string that is being returned (when set to false()).

Examples

Example 1

In this example, the design "my_company" is used by the siteaccess. Images should be included in the following way:

5

```
<img src={'banner.jpg'|ezimage()} alt="My banner" ...>
```

The following output will be produced:

```
<img src="/design/my_company/images/banner.jpg" alt="My company" ... />
```

If eZ Publish is unable to find the image within the images directory of the current design directory, it will attempt to find it within the images subdirectory of the additional designs. At last, it will fallback to the standard design. In this case, the output will be the following:

```
<img src="/design/standard/images/banner.jpg" alt="My company" ...>
```

5

ezroot

Summary

Same as "ezurl" without "index.php" and the siteaccess name in the returned address.

Usage

```
input|ezroot( [ quote [, type ] ] )
```

Parameters

Name	Type	Description	Required
quote	string	Quote style: "no", "single" or "double" (default).	No.
type	string	URL type: "full" or "relative" (default).	No.

Returns

A string containing a working version of the input address.

Description

This operator works almost in the same way as the "ezurl" (page 1342) operator. The only difference is that it does not include "index.php" or the name of the siteaccess in the returned address. In other words, it returns an address which points to the root of the eZ Publish directory.

5

ezurl

Summary

Returns a working version of an eZ Publish URL (provided as input).

Usage

```
input|ezurl( [ quote [, type ] ] )
```

Parameters

Name	Type	Description	Required
quote	string	Quote style: "no", "single" or "double" (default).	No.
type	string	URL type: "full" or "relative" (default).	No.

Returns

A quoted string containing a valid / working version of the input URL.

Description

This operator takes an eZ Publish URL as input (either a system URL or a virtual URL); based on the location of the eZ Publish folder, the access settings and the environment, it will produce a valid address. All eZ Publish URLs that are specified in templates should always be piped through this operator; it will make sure that the URLs work regardless where eZ Publish is installed, which access method is used, and so on.

By default, this operator returns a relative URL as a double-quoted string. The optional "quote" parameter can be used to control the way the address is returned: "no" (no quotes), "single" (single quotes) or "double" (double quotes, the default). The optional "type" parameter controls whether relative or full URL is returned.

Examples

Example 1

Let's say that we're running a site called "my_company" (name of the siteaccess) and that we wish to create a link to the full view of node number 1024. Instead of specifying the entire URL (domain and all included) in the link tag, we pipe "/content/view/full/1024" or the virtual URL (for example "/test") through the "ezurl" operator:

```
<a href={'/content/view/full/1024'|ezurl(,'full')}>Test</a>
<a href={'/test'|ezurl(,'full')}>Test</a>
```

The operator will take care of translating the URLs into valid addresses depending on the setup and the environment eZ Publish is running in. If eZ Publish is running in a virtual host environment and uses the host access method, the following type of URLs will be produced:

"http://www.example.com/content/view/full/1024"
"http://www.example.com/test"

The "index.php" part of the URL will be supressed by the virtual host configuration. The name of the siteaccess will not appear in the URL because eZ Publish will use the domain/host to figure out which siteaccess to use.

If eZ Publish is running in a non-virtual host environment and uses the uri access method, the following URLs will be produced:

"http://www.example.com/index.php/my_company/content/view/full/1024"
"http://www.example.com/index.php/my_company/test"

Example 2

```
<a href={'/content/view/full/1024'|ezurl()}>Test</a>
<a href={'/test'|ezurl()}>Test</a>
```

If eZ Publish is running in a virtual host environment and uses the host access method, the following type of URLs will be produced:

"/content/view/full/1024"
"/test"

The "index.php" part of the URL will be supressed by the virtual host configuration. The name of the siteaccess will not appear in the URL because eZ Publish will use the domain/host to figure out which siteaccess to use.

If eZ Publish is running in a non-virtual host environment and uses the uri access method, the following URLs will be produced:

"/index.php/my_company/content/view/full/1024"
"/index.php/my_company/test"

5.8.10 Variable and type handling

\begin{description} \item[count]\refer{label3ee84c6a1347bf9d5310ceeefe568cc9}\ \\ Returns the count of the input parameter. \item[float]\refer{label4c3bf8a76498a5ea6e5efe1d823039bb}\ \\ Converts the input parameter to a float. \item[get_ class]\refer{label51d1195ddeed346a0277363f13e22b3b}\ \\ Returns the class name of an object. \item[get_type]\refer{labelaa5eb6bbb8236fc87f149856a44ab9d5}\ \\ Returns the type of the provided variable. \item[int]\refer{label23a106215b2edce10d1b424db9f690d2}\ \\ Converts the input parameter to an integer. \item[is_ array]\refer{label0f86e27955cfcd2413c8d0570689f470}\ \\ Returns TRUE if the provided variable is an array. \item[is_boolean]\refer{labelf9453be60c0328afa57508cf4b243308}\ \\ Returns TRUE if the provided variable is a boolean. \item[is_ class]\refer{labelde0ee1f24ded38638089fbc356493964}\ \\ Returns TRUE if an object is an instance of a specific class. \item[is_ float]\refer{labelf3321b85d23be46917109e15ead6c69a}\ \\ Returns TRUE if the provided variable is a float. \item[is_integer]\refer{labele366bec10a3e5e0069111f9e6347b759}\ \\ Returns TRUE if the provided variable is an integer. \item[is_ null]\refer{labeld8b079a47afdfd6e1ed4c8ede8549e56}\ \\ Returns TRUE if the provided variable is NULL. \item[is_numeric]\refer{labelcc86e79115105cde80fe9ecaf3f66db5}\ \\ Returns TRUE if the provided variable is a number. \item[is_ object]\refer{label8c01162a70df93c20b7214776e0c1ba2}\ \\ Returns TRUE if the target variable is an object. \item[is_set]\refer{label3812048c8ed47e03d358fdb113176e6e}\ \\ Returns TRUE if the value of the provided variable is set. \item[is_ string]\refer{label443b6589f75345102be8ea51176ac7a3}\ \\ Returns TRUE if the provided variable is a string. \item[is_unset]\refer{label96c19d6215c2c043085d7dc15f827f3b}\ \\ Returns TRUE if the provided variable is not set (has no value).

\end{description}

5

count

Summary

Returns the count of the input parameter.

Usage

```
input|count()
```

Returns

An integer revealing the count.

Description

This operator returns the count of the input parameter. The following table shows how the operator works with different variable types.

Type	Description
Array	The number of elements is returned.
Object	The number of object attributes is returned.
String	The length of the string is returned.
Number	The value itself is returned.
Boolean	FALSE results in 0 and TRUE results in 1.
Other	0 is returned.

Examples

Example 1

```
{array( 1, 2, 5 )|count()}
```

The following output will be produced: "3".

5

float

Summary

Converts the input parameter to a float.

Usage

```
input|float()
```

Returns

A float representation of the input parameter.

Description

This operator attempts to convert the input parameter to a float. It returns the converted value as a float. If the operator is unable to do the conversion, it will return a value of zero (0).

Examples

Example 1

```
{def $pi='3.1415'}
Value: {$pi|float()}
```

Converts the string "3.1415" and returns it as a float. The following output will be produced: "Value: 3.1415".

Example 2

```
{def $pi='three point fourteen'}
Value: {$pi|float()}
```

The following output will be produced: "Value: 0".

5

get_class

Summary

Returns the class name of an object.

Usage

```
input|get_class( target )
```

Parameters

Name	Type	Description	Required
target	any	The target variable.	Only if the input parameter is omitted.

Returns

A string containing the class name or FALSE.

Description

This operator gets the class of the input parameter or the target variable. It returns the PHP class name as a string. If both the input parameter and the target variable are provided, it is the target variable that will be evaluated. If the provided variable is not an object then the operator will return FALSE.

Examples

Example 1

```
{def $my_variable="Test"}

{if get_class( $my_variable )}
    Class detected.
{else}
    There is no class.
{/if}
```

5

The following output will be produced: "There is no class.".

Example 2

```
{get_class( $node )}
```

If $node is an actual content node, the following output will be produced: "ezcontentobjecttreenode".

5

get_type

Summary

Returns the type of the provided variable.

Usage

```
input|get_type( target )
```

Parameters

Name	Type	Description	Required
target	any	The target variable.	Only if the input parameter is omitted.

Returns

A string containing the type of the provided variable.

Description

This operator can be used to check the type of a variable. It returns the type of the input parameter or the target variable as a string. If both the input parameter and the target variable are provided, it is the target variable that will be evaluated.

- If the data is an object, the string "object" and the name of the class will be returned.

- If the data is an array, the string "array" and the number of elements will be returned.

- If the data is a string, the string "string" and the length of the string will be returned.

Examples

Example 1

```
{def $my_variable='ich bin'}
{$my_variable|get_type()}
```

The following output will be produced: "string".

5

Example 2

```
{def $my_variable='ich bin'
     $your_variable=array( 'du', 'bist' )}

{$my_variable|get_type( $your_variable )}
```

The following output will be produced: "array[2]".

5

int

Summary

Converts the input parameter to an integer.

Usage

```
input|int()
```

Returns

An integer representation of the input parameter.

Description

This operator attempts to convert the input parameter to an integer. It will return the converted value as an integer. If the operator is unable to do the conversion, it will return the value of zero (0).

Examples

Example 1

```
{def $number='57'}
Value: {$number|int()}
```

Converts the string "57" and returns it as an integer. The following output will be produced: "Value: 57".

Example 2

```
{def $number='fiftyseven'}
Value: {$number|int()}
```

The following output will be produced: "Value: 0".

5

is_array

Summary

Returns TRUE if the provided variable is an array.

Usage

```
input|is_array( target )
```

Parameters

Name	Type	Description	Required
target	any	The target variable.	Only if the input parameter is omitted.

Returns

TRUE or FALSE.

Description

This operator checks if the input parameter or the target variable is an array. If it is, the operator will return TRUE, otherwise FALSE will be returned. If both the input parameter and the target variable are provided, it is the target variable that will be evaluated.

Examples

Example 1

```
{def $my_variable=array( 1, 2, 3 )}

{if $my_variable|is_array()}
    It is an array.
{else}
    It is not an array.
{/if}
```

The following output will be produced: "It is an array.".

5

Example 2

```
{def $my_variable=array( 1, 2, 3 )}

{if is_array( $my_variable )}
    It is an array.
{else}
    It is not an array.
{/if}
```

The following output will be produced: "It is an array.".

Example 3

```
{def $a=array( 1, 2, 3)
     $b='Mobile instrument.'}

{if $a|is_array( $b )}
    It is an array.
{else}
    It is not an array.
{/if}
```

The following output will be produced: "It is not an array.".

5

is_boolean

Summary

Returns TRUE if the provided variable is a boolean.

Usage

```
input|is_boolean( target )
```

Parameters

Name	Type	Description	Required
target	any	Target variable.	Only if the input parameter is omitted.

Returns

TRUE or FALSE.

Description

This operator checks if the input parameter or the target variable is a boolean. If it is, the operator will return TRUE, otherwise FALSE will be returned. If both the input parameter and the target variable are provided, it is the target variable that will be evaluated.

Examples

Example 1

```
{def $my_variable=true
{if $my_variable|is_boolean()}
    It is a boolean.
{else}
    It is not a boolean.
{/if}
```

The following output will be produced: "It is a boolean.".

Example 2

```
{def $my_variable=true
{if is_boolean( $my_variable )}
    It is a boolean.
{else}
    It is not a boolean.
{/if}
```

The following output will be produced: "It is a boolean.".

Example 3

```
{def $my_variable=true
     $your_variable='BOFID'}
{if $my_variable|is_boolean( $your_variable )}
    It is a boolean.
{else}
    It is not a boolean.
{/if}
```

The following output will be produced: "It is not a boolean.".

5

is_class

Summary

Returns TRUE if an object is an instance of a specific class.

Usage

```
input|is_class( name [, object ] )
```

Parameters

Name	Type	Description	Required
name	string	The class name that should be matched.	Yes.
object	object	An object of any type (instead of the input parameter).	Only if the input parameter is omitted.

Returns

TRUE or FALSE.

Description

This operator attempts to find out the class name of the object that is provided either as the input parameter or the "object" parameter. If the name of the class matches the name provided using the "name" parameter, the function will return TRUE; otherwise FALSE will be returned.

Examples

Example 1

```
{if $node|is_class( 'ezcontentobjecttreenode' )}
Everything is okay.
{else}
Something is wrong.
{/if}
```

As long as the $node refers to an instance of the "ezcontentobjecttreenode" class, the following output will be produced: "Everything is okay.".

5

Example 2

```
{if is_class( 'ezcontentobjecttreenode', $node )}
Everything is okay.
{else}
Something is wrong.
{/if}
```

As long as the $node refers to an instance of the "ezcontentobjecttreenode" class, the following output will be produced: "Everything is okay.".

5

is_float

Summary

Returns TRUE if the provided variable is a float.

Usage

```
input|is_float( target )
```

Parameters

Name	Type	Description	Required
target	any	The target variable.	Only if the input parameter is omitted.

Returns

true of false, see description for details.

Description

This operator checks if the input parameter or the target variable is a float. If it is, the operator will return TRUE, otherwise FALSE will be returned. If both the input parameter and the target variable are provided, it is the target variable that will be evaluated.

Examples

Example 1

```
{def $my_variable=3.1415}

{if $my_variable|is_float()}
    It is a float.
{else}
    It is not a float.
{/if}
```

The following output will be produced: "It is a float.".

Example 2

```
{def $my_variable=3.1415}

{if is_float( $my_variable )}
    It is a float.
{else}
    It is not a float.
{/if}
```

The following output will be produced: "It is a float.".

Example 3

```
{def $a=3.1415
     $b='Mobile instrument.'}

{if $a|is_float( $b )}
    It is a float.
{else}
    It is not a float.
{/if}
```

The following output will be produced: "It is not a float.".

5

is_integer

Summary

Returns TRUE if the provided variable is an integer.

Usage

```
input|is_integer( target )
```

Parameters

Name	Type	Description	Required
target	any	The target variable.	Only if the input parameter is omitted.

Returns

TRUE or FALSE.

Description

This operator checks if the input parameter or the target variable is an integer. If it is, the operator will return TRUE, otherwise FALSE will be returned. If both the input parameter and the target variable are provided, it is the target variable that will be evaluated.

Note: "is_integer" works like "is_int" in php, in other words a string with number in it will return false.

Examples

Example 1

```
{def $my_variable=3}

{if $my_variable|is_integer()}
    It is an integer.
{else}
    It is not an integer.
{/if}
```

The following output will be produced: "It is an integer.".

Example 2

```
{def $my_variable=3}

{if is_integer( $my_variable )}
    It is an integer.
{else}
    It is not an integer.
{/if}
```

The following output will be produced: "It is an integer.".

Example 3

```
{def $a=3
     $b='Mobile instrument.'}

{if $a|is_integer( $b )}
    It is an integer.
{else}
    It is not an integer.
{/if}
```

The following output will be produced: "It is not an integer.".

5

is_null

Summary

Returns TRUE if the provided variable is NULL.

Usage

```
input|is_null( target )
```

Parameters

Name	Type	Description	Required
target	any	The target variable.	Only if the input parameter is omitted.

Returns

TRUE or FALSE.

Description

This operator checks if the input parameter or the target variable is NULL. If it is, the operator will return TRUE, otherwise FALSE will be returned. If both the input parameter and the target variable are provided, it is the target variable that will be evaluated. Please note that an integer with a value of zero is not the same as NULL.

Examples

Example 1

```
{def $my_variable=3}

{if $my_variable|is_null()}
    It is NULL.
{else}
    It is not NULL.
{/if}
```

5

The following output will be produced: "It is not NULL.".

Example 2

```
{def $my_variable=3}

{if is_null( $my_variable )}
    It is NULL.
{else}
    It is not NULL.
{/if}
```

The following output will be produced: "It is not NULL.".

Example 3

```
{def $a=3
     $b='Mobile instrument.'}

{if $a|is_null( $b )}
    It is NULL.
{else}
    It is not NULL.
{/if}
```

The following output will be produced: "It is not NULL.".

5

is_numeric

Summary

Returns TRUE if the provided variable is a number.

Usage

```
input|is_numeric( target )
```

Parameters

Name	Type	Description	Required
target	any	The target variable.	Only if the input parameter is omitted.

Returns

TRUE or FALSE.

Description

This operator checks if the input parameter or the target variable is a number or a numeric string (a string containing a number). If it is, the operator will return TRUE, otherwise FALSE will be returned. If both the input parameter and the target variable are provided, it is the target variable that will be evaluated.

Examples

Example 1

```
{def $my_variable=3}

{if $my_variable|is_numeric()}
    It is a number.
{else}
    It is not a number.
{/if}
```

The following output will be produced: "It is a number.".

Example 2

```
{def $my_variable='256'}

{if is_numeric( $my_variable )}
    It is a number.
{else}
    It is not a number.
{/if}
```

The following output will be produced: "It is a number.".

Example 3

```
{def $a=3
     $b='Mobile instrument.'}

{if $a|is_numeric( $b )}
    It is a number.
{else}
    It is not a number.
{/if}
```

The following output will be produced: "It is not a number.".

5

is_object

Summary

Returns TRUE if the target variable is an object.

Usage

```
input|is_object( target )
```

Parameters

Name	Type	Description	Required
target	any	The target variable.	Only if the input parameter is omitted.

Returns

TRUE or FALSE.

Description

This operator checks if the input parameter or the target variable is an object (as opposed to simple types like integer, string, etc.). If it is, the operator will return TRUE, otherwise FALSE will be returned. If both the input parameter and the target variable are provided, it is the target variable that will be evaluated.

Examples

Example 1

```
{def $my_variable=3}

{if $my_variable|is_object()}
    It is an object.
{else}
    It is not an object.
{/if}
```

5

The following output will be produced: "It is not an object.".

Example 2

```
{def $my_variable='256'}

{if is_object( $my_variable )}
    It is an object.
{else}
    It is not an object.
{/if}
```

The following output will be produced: "It is not object.".

Example 3

```
{def $a=3
     $b='Mobile instrument.'}

{if $a|is_object( $b )}
    It is an object.
{else}
    It is not an object.
{/if}
```

The following output will be produced: "It is not an object.".

5

is_set

Summary

Returns TRUE if the value of the provided variable is set.

Usage

```
is_set( target )
```

Parameters

Name	Type	Description	Required
target	any	The target variable.	Yes.

Returns

TRUE or FALSE.

Description

This operator checks if the value of the target parameter is a non-false value (meaning that it is set). If it is, the operator will return TRUE, otherwise FALSE will be returned. Please note that this operator does not take an input parameter.

Examples

Example 1

```
{if is_set( $whatever )}
    It is set.
{else}
    It is not set.
{/if}
```

The following output will be produced: "It is not set." - because $whatever is not declared and/ or defined.

Example 2

5

```
{def $whatever='We need more rocket fuel!'}
{if is_set( $whatever )}
    It is set.
{else}
    It is not set.
{/if}
```

The following output will be produced: "It is set.".

5

is_string

Summary

Returns TRUE if the provided variable is a string.

Usage

```
input|is_string( target )
```

Parameters

Name	Type	Description	Required
target	any	The target variable.	Only if the input parameter is omitted.

Returns

TRUE or FALSE.

Description

This operator checks if the input parameter or the target variable is a string. If it is, the operator will return TRUE, otherwise FALSE will be returned. If both the input parameter and the target variable are provided, it is the target variable that will be evaluated.

Examples

Example 1

```
{def $my_variable='Commodore'}

{if $my_variable|is_string()}
    It is a string.
{else}
    It is not a string.
{/if}
```

The following output will be produced: "It is a string.".

Example 2

```
{def $my_variable='Amiga'}

{if is_string( $my_variable )}
    It is a string.
{else}
    It is not a string.
{/if}
```

The following output will be produced: "It is a string.".

Example 3

```
{def $a='C64'
     $b=128}

{if $a|is_string( $b )}
    It is a string.
{else}
    It is not a string.
{/if}
```

The following output will be produced: "It is not a string.".

5

is_unset

Summary

Returns TRUE if the provided variable is not set (has no value).

Usage

```
is_unset( target )
```

Parameters

Name	Type	Description	Required
test	any	The target variable.	Yes.

Returns

TRUE or FALSE.

Description

This operator checks if the target variable is set (has a value). If it is, the operator returns FALSE, otherwise TRUE will be returned.

Examples

Example 1

```
{def $my_variable=true()}
{if is_unset( $my_variable )}
    Yes.
{else}
    No.
{/if}
```

The following output will be produced: "No.".

Example 2

```
{def $my_variable=false()}
{if is_unset( $my_variable )}
    Yes.
```

5

```
{else}
    No.
{/if}
```

The following output will be produced: "Yes.".

5

5.9   Template functions

The template functions are documented in the following sections:

\begin{itemize}   \item{}   Debugging\refer{label04881a8ca3637dc5860aea163100e1e4}
\item{}   Miscellaneous\refer{label2807c0fb3a746eb1f6277409985b5153}
\item{}   Variables\refer{labela37615988375bc4c85b9472863202b67}   \item{}
Visualization\refer{labelccffced82f5c1b9c5c99f8e46f4788c1} \end{itemize}

5

5.9.1 Debugging

debug-accumulator (page 1376)
Generates debug statistics for a block of template code.

debug-log (page 1378)
Generates a debug log entry for the specified variables.

debug-timing-point (page 1380)
Measures the time it takes to process a block of template code.

debug-trace (page 1381)
Generates an XDebug dump which can be traced/analyzed.

5

debug-accumulator

Summary

Generates debug statistics for a block of template code.

Usage

```
{debug-accumulator [ id=id ] [ name=name ]}
...
{/debug-accumulator}
```

Parameters

Name	Type	Description	Required
id	string	A unique identifier string for an accumulator.	No.
name	string	A name that will be used in the debug output.	No.

Description

This mechanism generates some debug statistics based on the code that is encapsulated by
"{debug-accumulator}" and "{/debug-accumulator}". The encapsulated code will be processed
normally. The number of calls, total time and average time will be shown in the debug output.

The "id" and the "name" parameters are optional. The "id" parameter can be used to uniquely
identify one accumulator. This means that if the same accumulator is used at multiple locations,
the values will be accumulated and appended. The "name" parameter can be used to assign a
name which will be used in the debug output.

Examples

Example 1

```
{debug-accumulator}

{def $nodes=fetch( 'content', 'tree', hash( 'parent_node_id', 2 ) )}

{foreach $nodes as $node}
    {$node.name|wash()}
{/foreach}

{/debug-accumulator}
```

This example demonstrates how the "debug-accumulator" mechanism can be used to generate some debug statistics based on the encapsulated template code.

5

debug-log

Summary

Generates a debug log entry for the specified variables.

Usage

```
{debug-log [ var=variable ] [ msg=message ]}
```

Parameters

Name	Type	Description	Required
var	mixed	A variable to be outputted to the debug-log	No.
msg	string	A label for the debug message.	No.

Description

This function creates a debug log entry based on the input variables to the function. If none of the parameters are used, no log output will be shown. If both "var" and "msg" parameters are given, "msg" will be the label of the log entry, and "var" will be the variable to be outputted. "var" can be any variable, eg, a string, an array or an object. If the php extension, xdebug is installed, it will be used to render the variable, as per normal eZDebug behavior.

If only one parameter is used, the function will treat both parameters equally and the content of the variable used will be outputted to the debug-log.

Examples

Example 1

```
{debug-log var=$object msg='object contents'}
```

This example demonstrates the "debug-log" with both parameters used. "msg" will be used as label and "var" as output.

Example 2

```
{debug-log msg='hello world'}
```

This example demonstrates the "debug-log" with only the "msg" parameter used. "msg" will be used as output.

Example 3

```
{debug-log var=array(1,2,3)}
```

This example demonstrates the "debug-log" with only the "var" parameter used. "var" will be used as output.

5

debug-timing-point

Summary

Measures the time it takes to process a block of template code.

Usage

```
{debug-timing-point [ id=id ]}
...
{/debug-timing-point}
```

Parameters

Name	Type	Description	Required
id	string	An identification string for the timing point.	No.

Description

This mechanism starts a timer, executes the template code that is encapsulated by "{debug-timing-point...}" and "{/debug-timing-point}", finally it stops the timer. It can be used to figure out how much time it takes to process a block of template code and/or to isolate debug messages that are generated between the timing points. The measurements will show up in the debug message. The optional "id" parameter can be used to assign an identification string to the block.

Examples

Example 1

```
{debug-timing-point id='test'}

{def $nodes=fetch( 'content', 'tree', hash( 'parent_node_id', 2 ) )}

{foreach( $nodes as $node )}
    {$node.name|wash()}
{/foreach}

{/debug-timing-point}
```

This example demonstrates how the "debug-timing-point" mechanism can be used to measure the amount of time it takes to fetch and print the names of all nodes that are below node number 2.

5

debug-trace

Summary

Generates an XDebug dump which can be traced/analyzed.

Usage

```
{debug-trace [ id=id ]}
...
{/debug}
```

Parameters

Name	Type	Description	Required
id	string	The name of the debug file.	No.

Description

This mechanism makes it possible to trace a block of code using "XDebug". The result will be a trace file made by XDebug which can be analyzed. If XDebug is not installed and enabled, this mechanism will not do anything. The "id" parameter can be used to name the trace file. The file extension will be ".xt". The default ID/name is "template-debug". Please note that the trace file will be reset every time a debug-trace is encountered; it is a good idea to have one unique ID per entry.

Examples

Example 1

```
{debug-trace id='fetch-trace'}

{def $nodes=fetch( 'content', 'tree', hash( 'parent_node_id', 2 ) )}

{foreach( $nodes as $node )}
    {$node.name|wash()}
{/foreach}

{/debug-trace}
```

This will generate an XDebug trace file called "fetch-trace.xt".

5

5.9.2 Miscellaneous

\begin{description} \item[cache-block]\refer{labelb8b91dcebaf9f0042e3f77e02213a2a6}\ \\
Caches the contents of a template block to static content for fast retrieval. Useful to avoid fetching content over and over from database when it is possible to cache it within a set of parameters. Note: cache-blocks also introduces some overhead, so do not have too many of them on a page, let them span parts of your page like e.g. header, footer and left menu.

\item[fetch_alias]\refer{label428c4c99946d68f1f365c7099dc49250}\ \\ Executes a fetch based on configuration settings.

\item[include]\refer{label52163a3eec940d3fa00283a506e944e5}\ \\ Includes a file.

\item[ldelim]\refer{label34c1292da666a01510559ddf5efa0516}\ \\ Outputs a left curly bracket, "\{".

\item[literal]\refer{label8f49ba0a5580438fb69da18c18c75174}\ \\ Instructs the parser to ignore a block of template code.

\item[rdelim]\refer{label641d2e6a68779094eb41e7564b1b6833}\ \\ Outputs a right curly bracket, "\}".

\item[run-once]\refer{labela0d0e780e34e37d8e0bc242d4c420bd7}\ \\ Assures that a block of template code is run only once within a page view.

\end{description}

5

cache-block

Summary

Caches the contents of a template block to static content for fast retrieval. Useful to avoid fetching content over and over from database when it is possible to cache it within a set of parameters. Note: cache-blocks also introduces some overhead, so do not have too many of them on a page, let them span parts of your page like e.g. header, footer and left menu.

Usage

```
{cache-block [ keys=keys                    ]
             [ expiry=expiry                ]
             [ ignore_content_expiry        ]
             [ subtree_expiry=subtree_expiry  ]}


...


{/cache-block}
```

Parameters

Name	Type	Description	Required
keys	string or array	Cache key(s) - either as a string or an array of strings.	No.
expiry	integer	The number of seconds that the cache should be allowed to live.	No.
ignore_content_expiry	-	Disables cache expiry when new content is published.	No.
subtree_expiry	string or integer	A subtree that expires the cache block, either node url alias or node id. no.	No.

Description

This solution makes it possible to reduce the processing time of the main template ("pagelayout.tpl"), which often contains a lot of dynamic elements. It can be used to instruct the system to store and reuse cached blocks of template code based on different conditions.

A typical example of where the "cache-block" mechanism should be used is the main menu of a site. The menu is often dynamically generated by fetching and displaying information about some nodes. It is usually the same for almost every page, therefore it should not be generated from scratch every time eZ Publish is instructed to render a page. This is where the "cache-block" solution comes in. In this particular scenario, it can be used to cache the contents of the main menu and thus reduce the processing time for each page load.

Note 1: Template variables defined or set within the cache block will not be available further down the page, as only the html resulting from the execution of the cache-block is stored. For non-security-critical information, you might use javascript to that effect.

Note 2: The cache-block stores it's content as text fragments, and does not execute any of it content again before the cache has been expired.

Cache keys

The "keys" parameter can be used to define the uniqueness of a cache block. It must be either a string or an array of strings. By default, eZ Publish uses the name of the template and the position of the cache block as keys. This means that if the cache block is common for all cases that use the given template (normally "pagelayout.tpl"), there is no need to define any keys. However, the "keys" parameter is quite handy when it comes to relating a cache block to something specific (for example URLs, users, etc.). Please refer to the examples below for a demonstration of how this parameter can be used.

Time-based expiration

The "expiry" parameter makes it possible to manually specify how long a cache block should live (number of seconds). The default expiration time is two hours (this is hard-coded in the system and can not be configured). If an object is published, all blocks will automatically be expired. A value of zero will produce a cache block that will never expire.

Content expiration

By default, all cache blocks will be expired whenever an object is published. If the "ignore_content_expiry" parameter is used, the cache block will not be expired when an object is published. However, it will still expire after two hours unless an alternative time is specified using the "expiry" parameter.

Subtree expiration

The "subtree_expiry" parameter can be used to bind the expiration of a cache block to a certain part of the content node tree. When this is done, the block will expire if an object is published below the given subtree instead of the entire tree. In addition, it will also expire after two hours unless an alternative time is specified using the "expiry" parameter.

Value can be either node url alias, a node id or system url in the following format: "content/view/full/<node_id>".

Note: Using node id is recommended as it avoids node id look-up by url, avoids potential issues if url is changed and multi-lingual url if you hard code url alias and it's used on a siteaccess with different locale.

Tips and tricks

Since cache blocks themselves also produce some overhead, too many blocks may lead to longer response times than expected. Because of this, only a few cache blocks should be used; and their keys should be as unique as possible. It is often very efficient to have two large cache blocks. One which caches all header information (title, path, etc.) and one which will take care of the bottom/footer of the page. This solution combined with a nested cache block used for the main menu (or several menus, etc.) often leads to good results. Please note that although the cache block mechanism was designed to minimize the processing of the main template, it may also be used in view templates. For example, it is possible to cache a part of a view - this is typically useful when the viewcache is frequently deleted. Another scenario is when the view cache is turned off and there is a need to create a cache on a per-user basis.

Examples

Example 1

```
...
{include uri='design:page_toppath.tpl'}

{cache-block}
    {include uri='design:menu.tpl'}
{/cache-block}

{$module_result.content}

{include uri='design:page_bottom.tpl'}
...
```

This example demonstrates how the cache block solution can be used to cache the contents of a menu (which will be the same for all pages) in the page layout.

Example 2

```
{cache-block expiry=130}
...
{/cache-block}
```

This example demonstrates how to create a cache block that will expire after 130 seconds.

Example 3

```
{cache-block keys=$uri_string}
...
{/cache-block}
```

This example demonstrates how to create a cache block that will be unique for every URL.

Example 4

```
{cache-block keys=array( $uri_string, $current_user.contentobject_id )}
...
{/cache-block}
```

This example demonstrates how to create a cache block that is unique for each URL and each user.

Example 5

```
{cache-block ignore_content_expiry}
...
{/cache-block}
```

This example demonstrates how to create a cache block that will not expire when new content is published. However, it will expire every second hour unless an alternative "time to live" value is specified using the "expiry" parameter.

Example 6

```
{cache-block subtree_expiry=44}
...
{/cache-block}
```

This example demonstrates how to create a cache block that will expire only if something is modified within the subtree of node id '44', for example if sub content is modified or new content is published beneath this node.

Example 7

```
{cache-block subtree_expiry=$my_node.node_id}
...
{/cache-block}
```

This example demonstrates how to create a cache block that will expire only if something is modified within the subtree of $my_node. Example demonstrates that node id can come from an dynamic variable.

5

fetch_alias

Summary

Executes a fetch based on configuration settings.

Usage

```
fetch_alias( alias_name, hash( [ parameter1, value1, ]
                               [ parameter2, value2  ] ) )
```

Parameters

Name	Type	Description	Required
alias_name	string	The name of the fetch alias that should be used.	Yes.
parameter1	string	The name of parameter 1.	No.
value1	string	The value of parameter 1.	No.
parameter2	string	The name of parameter2.	No.
value2	string	The value of parameter 2.	No.

Description

This function can be thought of as a configuration-file based version of the "fetch" (page 1174) operator. It makes it possible to move data fetching blocks from template code to a configuration file and thus gather all fetches at one place. The advantage of such a scenario is that it allows quick modifications without the need of having to locate and modify different templates. The fetch aliases must be defined in a configuration override for "fetchalias.ini". Each fetch has to be defined within its own block with a unique name. The following code shows the basic syntax/ structure of a fetch block.

```
[fetch_alias_name]
Module=module_name
FunctionName=function_name
Parameter[parameter1]=fetch_alias_name1
Parameter[parameter2]=fetch_alias_name2
...
Constant[parameter3]=<any value>
Constant[parameter4]=<any value>
...
```

Directive	Description
Module	The name of the target module (for example

5

	"content").
FunctionName	The name of the target fetch function (for example "list").
Parameter	The "Parameters" array may be used to specify variables that will be set in the template(s). The "parameter_name" maps to the parameter name used in normal fetch functions. The "fetch_alias_name" will be the parameter name used in the template(s).
Constant	Parameters that are defined as constants within the regular fetch function(s).

Examples

Example 1

Configuration block:

```
[object]
Module=content
FunctionName=object
Parameter[object_id]=id
```

Template code:

```
{def $object=fetch_alias( 'object', hash( 'id', 1 ) )}
```

This example demonstrates how to fetch an object.

Example 2

Configuration block:

```
[comments]
Module=content
FunctionName=list
Constant[sort_by]=published;0
Parameter[parent_node_id]=parent_node_id
Constant[class_filter_type]=include
Constant[class_filter_array]=comment
```

Template code:

```
{def $comments=fetch_alias( 'comments', hash( 'parent_node_id', 42 ) )}
```

This example demonstrates how to fetch comments.

Example 3

Configuration block:

```
[news_list]
Module=content
FunctionName=tree
Constant[sort_by]=published;0
Constant[class_id]=2
Constant[parent_node_id]=2
Constant[class_filter_type]=include
Constant[limit]=10
Constant[class_filter_array]=2
```

Template code:

```
{foreach fetch_alias( 'news_list' ) as $article}
    {node_view_gui node=$article}
{/foreach}
```

This example demonstrates how to fetch and display the 10 latest news articles using full view.

5

include

Summary

Includes a file.

Usage

```
{include uri='path_to_file' [ name='namespace' ] [ parameter(s)='value(s)'  ]}
```

Parameters

Name	Type	Description	Required
uri	string	Path + name of the file that should be included.	Yes.
name	string	Alternative namespace for the included template.	No.
other parameters	any	Parameters that will be passed to the included template.	No.

Description

This function includes a file in the template from where the function was called. The "uri" param-
eter must be used to specify the target file. In most cases, the value of this parameter starts with
a "design:", which tells the system to look for the desired template within the current (and fall-
back) design resources. The "name" parameter can be used to specify an alternative namespace
for the included template, this is useful for avoiding variable name clashes when including other
templates. All other parameters will be passed to the included template as template variables.
This function makes it possible to share template code among different parts of the solution.

Examples

Example 1

```
{include uri='design:example/menu.tpl' something='Hello world'}
```

This example demonstrates how to include a template called "menu.tpl" (which is located within
the "example" subdirectory of the "templates" directory. If eZ Publish is unable to find the tem-
plate in the current design, it will automatically attempt to locate it in one of the fallback designs
or the standard design. The value of the "something" parameter will be available through a
variable called $something within the template that is included.

5

ldelim

Summary

Outputs a left curly bracket, "{".

Usage

```
{ldelim}
```

Description

This function displays a left curly bracket, "{". It can for example be used to add JavaScript functions in a template.

Examples

Example 1

```
<script language="JavaScript" type="text/javascript">
<!--
function foo()
{ldelim}
    alert ('Call me!' );
{rdelim}
//-->
</script>
```

This example demonstrates how to use the "ldelim" and "rdelim" template functions to generate curly brackets.

5

literal

Summary

Instructs the parser to ignore a block of template code.

Usage

```
{literal}
...
{/literal}
```

Description

This function can be used to encapsulate foreign code (for example JavaScript) that makes use of characters that may confuse the template parser. Everything that is inside a literal block will be completely ignored by the parser.

Examples

Example 1

```
{literal}
<script language="JavaScript" type="text/javascript">
<!--
function foo()
{
    alert ( "Call me" );
}
//-->
</script>
{/literal}
```

This example demonstrates how to include a JavaScript snippet in a template using the "{literal}" and "{/literal}" notation.

5

rdelim

Summary

Outputs a right curly bracket, "}".

Usage

```
{rdelim}
```

Description

This function displays a right curly bracket, "}". It can for example be used to add JavaScript functions in a template.

Examples

Example 1

```
<script language="JavaScript" type="text/javascript">
<!--
function foo()
{ldelim}
    alert ('Call me!' );
{rdelim}
//-->
</script>
```

This example demonstrates how to use the "ldelim" and "rdelim" template functions to generate curly brackets.

5

run-once

Summary

Assures that a block of template code is run only once within a page view.

Usage

```
{run-once}
    ...
{/run-once}
```

Description

This function makes sure that a block of template code is processed only once within a page view. It is typically useful when it comes to displaying elements that should appear once or to do timeconsuming calculations that only has to be processed once (and the result is included in multiple templates).

Examples

Example 1

```
{def $elements=array( 'A', 'B', 'C' )}

{foreach $elements as $element}

    {run-once}
        Hello world <br/>
    {/run-once}

    {$element} <br />

{/foreach}
```

The following output will be produced:

```
Hello World
A
B
C
```

5.9.3   Variables

append-block   (page 1396)
> Redirects the output from multiple blocks of template code to an array.

def   (page 1398)
> Declares (and defines) a variable. Warns if the variable already exists.

default   (page 1400)
> Deprecated.

let   (page 1401)
> Deprecated.

sequence   (page 1402)
> Creates a sequence that can be iterated.

set   (page 1403)
> Sets the value of a variable.

set-block   (page 1405)
> Redirects the output from a block of template code to a string.

undef   (page 1407)
> Destroys previously defined variable(s).

5

append-block

Summary

Redirects the output from multiple blocks of template code to an array.

Usage

```
{append-block variable=$variable [ name=name ] [ scope=scope ]}
...
{/append-block}
```

Parameters

Name	Type	Description	Required
name	string	Name of the namespace.	No.
scope	string	The scope ("global", "root" or "relative").	No.
variable	string	The name of variable that will be returned.	Yes.

Description

This mechanism will silently process all template code which is encapsulated by "{append-block ...}" and "{/append-block}". It will not produce any actual output. Instead, the generated output will be assigned to a variable specified by the "variable" parameter (as an array). If the variable does not exist, it will be automatically created. If the same target variable is used in several blocks, the function will simply add new elements to the array and thus the previous contents will be preserved.

Examples

Example 1

```
{append-block variable=$alien}
    It seems to have a life,
{/append-block}


...


{append-block variable=$alien}
    organic life...
{/append-block}
```

```
...

{foreach $alien as $element}
    {$element} <br />
{/foreach}
```

This example demonstrates how to create an array called $alien using the "append-block" mechanism. The output from the code that is encapsulated by "{append-block ...}" and "{/append-block}" will be assigned as elements to the target variable. When the $alien array is inspected, the following output will be produced:

It seems to have a life,
organic life...

5

def

Summary

Declares (and defines) a variable. Warns if the variable already exists.

Usage

```
{def $var1=value1 [ $var2=value2 [...] ]}
```

Parameters

Name	Type	Description	Required
$var1	string	Name of variable number one (with a dollar sign in front of it).	Yes.
value1	any	The value that should be assigned to variable one.	Yes.
$var2	string	Name of variable number two (with a dollar sign in front of it).	No.
value2	any	The value that should be assigned to variable two.	No.

Description

This function allows the declaration and definition of a single variable or a group of variables. The "undef" (page 1407) function can be used to flush/destroy variables that were created using the "def" function. Please note that this function does not support the "name" and the "scope" parameters (like the old {let} did).

Replacement for "default"

The following technique can be used as a replacement for the old "default" function:

```
{if is_set( $a )|not}
  {def $a=5}
```

Examples

Example 1

```
{def $oranges=13}
```

5

This example demonstrates how the "def" function can be used to declare a variable called "oranges". The variable will be declared as an integer with a value of 13.

Example 2

```
{def $oranges=13 $apples='There are no apples.'}
```

or

```
{def $oranges=13
     $apples='There are no apples.'}
```

These code snippets demonstrates how the "def" function can be used to declare multiple variables. A variable called "oranges" will be declared as an integer with a value of 13. A variable called "apples" will be declared as a string containing the following characters: "There are no apples.".

5

default

Summary

Deprecated.

Description

This function is deprecated and should not be used. If you need more information, please refer to the documentation of the old template syntax (it is still present in some of the default templates included in the distributions).

5

let

Summary

Deprecated.

Description

This function is deprecated and should not be used. If you need more information, please refer to the documentation of the old template syntax (it is still present in some of the default templates included in the distributions).

5

sequence

Summary

Creates a sequence that can be iterated.

Usage

```
{sequence name=name loop=loop}
```

Parameters

Name	Type	Description	Required
name	string	The name of the target namespace.	Yes.
loop	array	The iteration elements (as an array).	Yes.

Description

This function allows the creation of a sequence which can be iterated. If the number of iterations exceed the length of the sequence, the contents of the sequence will be wrapped and thus repeated. It is typically useful when it comes to the creation of lists / tables with alternating colors. Both the name and the elements (which will be iterated) must be defined.

5

set

Summary

Sets the value of a variable.

Usage

```
{set $var1=value1 [ var2=value2 [...] ] [ name=name ] [ scope=scope ]}
```

Parameters

Name	Type	Description	Required
var1	string	Name of variable number one (with a dollar sign in front of it).	Yes.
value1	any	The value that should be assigned to variable 1.	Yes.
var2	string	Name of variable number two (with a dollar sign in front of it).	No.
value2	any	The value that should be assigned to variable 2.	No.
name	string	The name of the target namespace.	No.
scope	string	The scope ("global", "root" or "relative").	No.

Description

This function makes it possible to assign new values to variables that previously have been declared using either the "def" (page 1398) function. The "name" and "scope" parameters are optional and can be used to set the desired namespace and scope.

Examples

Example 1

```
{def $apples=4}
Before: {$apples} <br/>
...
{set $apples=8}
After: {$apples} <br/>
```

The following output will be produced:

5

```
Before: 4
After: 8
```

Example 1

```
{def name=ns1 $var1='ns1 org value'}
{def name=ns2 $var1='ns2 org value'}

Original values: <br/>
$ns1:var1 : {$ns1:var1} <br/>
$ns1:ns2:var1 : {$ns1:ns2:var1} <br/>
...
{set name=ns1 scope=root var1='new value'}
{set var1='new value'}
...
New values: <br />
$ns1:var1 : {$ns1:var1}<br/>
$ns1:ns2:var1 : {$ns1:ns2:var1}<br/>
```

The following output will be produced:

```
Original values:
$ns1:var1 : ns1 org value
$ns1:ns2:var1 : ns2 org value

New values:
$ns1:var1 : new value
$ns1:ns2:var1 : new value
```

5

set-block

Summary

Redirects the output from a block of template code to a string.

Usage

```
{set-block variable=$variable [ name=name ] [ scope=scope ]}
...
{/set-block}
```

Parameters

Name	Type	Description	Required
variable	string	The name of the variable (with dollar sign).	Yes.
name	string	The name of the target namespace.	No.
scope	string	The scope ("global", "root" or "relative").	No.

Description

This mechanism will silently process all template code which is encapsulated by "{set-block ...}" and "{/set-block}". It will not produce any actual output. Instead, the generated output will be assigned to a variable specified by the "variable" parameter (as a string). If the variable does not exist, it will be automatically created. If the same target variable is used in several blocks, its contents will be overwritten every time a new block is processed.

Examples

Example 1

```
{set-block variable=$example}
    {def $test=array( 'x', 'y', 'z' )}
    Hello world - {$test[1]}
{/set-block}
...
{$example}
```

The code which is encapsulated by "set-block" will not produce any output. Instead, the output will be put into a string called $example. When this variable is accessed directly, the following output will be produced: "Hello world - y".

5

Example 2

```
{set-block scope=global variable=cache_ttl}0{/set-block}
```

This will put zero into the "cache_ttl" global variable and thus disable view caching for this page.

5

undef

Summary

Destroys previously defined variable(s).

Usage

```
{undef [ $var1 [ ... ] ]}
```

Parameters

Name	Type	Description	Required
var1	string	The name of the variable that should be destroyed ($-notation).	No.

Description

This function destroys variables that have previously been created using the "def" (page 1398) function. The names of the variables that should be destroyed can be provided as parameters. If no parameters are specified, the function will automatically get rid of all variables that were previously defined within the current/same namespace.

Examples

Example 1

```
{def $a=1 $b=2 $c=3}
...
{undef}
```

This example demonstrates how the "undef" function can be used to clean up / get rid of a previously the variables that were previously created using the "def" function .

Example 2

```
{def $a=1 $b=2 $c=3}
...
{undef $b}
```

This example demonstrates how the "undef" function can be used to clean up / get rid of a previously created variable. The variables $a and $c will not be destroyed.

5.9.4   Visualization

attribute_edit_gui   (page 1410)
> Outputs the edit template for a content object attribute.

attribute_pdf_gui   (page 1411)
> Outputs the PDF template for a content object attribute (DEPRECATED).

attribute_result_gui   (page 1412)
> Outputs the result template for a content object attribute.

attribute_view_gui   (page 1413)
> Outputs the view template for a content object attribute.

class_attribute_edit_gui   (page 1414)
> Outputs the edit template for a content class attribute.

class_attribute_view_gui   (page 1415)
> Outputs the view template for a content class attribute.

collaboration_icon   (page 1416)
> Outputs the icon for a collaboration item.

collaboration_participation_view   (page 1417)
> Outputs information about a collaboration participant.

collaboration_simple_message_view   (page 1418)
> Outputs the view template for a collaboration message.

collaboration_view_gui   (page 1419)
> Outputs the template for a collaboration item.

content_pdf_gui   (page 1420)
> Outputs the PDF template for a content object (DEPRECATED).

content_version_view_gui   (page 1421)
> Outputs a view template for a content object version.

content_view_gui   (page 1422)
> Outputs a view template for a content object.

event_edit_gui   (page 1423)
> Outputs the edit template for a workflow event.

node_view_gui   (page 1424)
> Outputs the view template for a node.

related_view_gui   (page 1425)
> Not documented yet.

shop_account_view_gui   (page 1426)
> Outputs the view template for a specified order.

5

tool_bar   (page 1427)
    Outputs the template for a toolbar.

5

attribute_edit_gui

Summary

Outputs the edit template for a content object attribute.

Usage

```
{attribute_edit_gui attribute=attribute [ parameter=value [...] ]}
```

Parameters

Name	Type	Description	Required
attribute	object	The target content object attribute.	Yes.
parameter	any	Parameter(s) that will be passed to the included template.	No.

Description

This function shows the edit interface for a content object attribute. The attribute must be specified (as a "ezcontentobjectattribute" (page 1042) object) using the "attribute" parameter. All other parameters (of any type) will be passed on and thus become available as template variables in the included template.

5

attribute_pdf_gui

Summary

Outputs the PDF template for a content object attribute (DEPRECATED).

Usage

```
{attribute_pdf_gui attribute=attribute [ parameter=value [...] ]}
```

Parameters

Name	Type	Description	Required
attribute	object	The target content object attribute.	Yes.
parameter	any	Parameter(s) that will be passed to the included template.	No.

Description

This function shows the PDF view interface for a content object attribute. The target attribute must be specified (as a "ezcontentobjectattribute" (page 1042) object) using the "attribute" parameter. All other parameters (of any type) will be passed on and thus become available as template variables in the included template

5

attribute_result_gui

Summary

Outputs the result template for a content object attribute.

Usage

```
{attribute_result_gui attribute=attribute [ parameter=value [...] ]}
```

Parameters

Name	Type	Description	Required
attribute	object	The target content object attribute.	Yes.
parameter	any	Parameter(s) that will be passed to the included template.	No.

Description

This function shows the information collection result interface for a content object attribute. The attribute must be specified (as a "ezcontentobjectattribute" (page 1042) object) using the "attribute" parameter. All other parameters (of any type) will be passed on and thus become available as template variables in the included template.

5

attribute_view_gui

Summary

Outputs the view template for a content object attribute.

Usage

```
{attribute_view_gui attribute=attribute [ parameter=value [...] ]}
```

Parameters

Name	Type	Description	Required
attribute	object	The target content object attribute.	Yes.
parameters	any	Parameter(s) that will be passed to the included template.	No.

Description

This function shows the view interface for a content object attribute. The attribute must be specified (as a "ezcontentobjectattribute" (page 1042) object) using the "attribute" parameter. All other parameters (of any type) will be passed on and thus become available as template variables in the included template.

5

class_attribute_edit_gui

Summary

Outputs the edit template for a content class attribute.

Usage

```
{class_attribute_edit_gui class_attribute=attribute [ parameter=value [...] ]}
```

Parameters

Name	Type	Description	Required
class_attribute	object	The target content class attribute.	Yes.
other parameters	any	Parameter(s) that will be passed to the included template.	No.

Description

This function shows the edit interface for a content class attribute. The attribute must be specified (as a "ezcontentclassattribute" (page 1026) object) using the "attribute" parameter. All other parameters (of any type) will be passed on and thus become available as template variables in the included template.

5

class_attribute_view_gui

Summary

Outputs the view template for a content class attribute.

Usage

```
{class_attribute_view_gui class_attribute=attribute [ parameter=value [...] ]}
```

Parameters

Name	Type	Description	Required
class_attribute	object	The target content class attribute.	Yes.
parameter	any	Parameter(s) that will be passed to the included template.	No.

Description

This function shows the view interface for a content class attribute. The attribute must be specified (as a "ezcontentclassattribute" (page 1026) object) using the "attribute" parameter. All other parameters (of any type) will be passed on and thus become available as template variables in the included template.

5

collaboration_icon

Summary

Outputs the icon for a collaboration item.

Usage

```
{collaboration_icon collaboration_item=item [ view=view  [ parameter=value
[...] ] ]}
```

Parameters

Name	Type	Description	Required
collaboration_item	object	Collaboration item object.	Yes.
view	string	The view mode to use.	No.
parameter	any	Parameters passed to the GUI template.	No.

Description

This function outputs the icon for a collaboration item. The "collaboration_item" parameter must be used to specify the target collaboration item. The "view" parameter is optional, it can be used to specify a view mode (for example "small"). All other parameters (of any type) will be passed on and thus become available as template variables in the included template.

5

collaboration_participation_view

Summary

Outputs information about a collaboration participant.

Usage

```
{collaboration_participation_view collaboration_participant=link [ view=view [
parameter=value [...] ] ]}
```

Parameters

Name	Type	Description	Required
collaboration_ participant	object	The target collaboration participant object.	Yes.
view	string	The view mode to use.	No.
parameters	mixed	Parameters passed to the GUI template.	No.

Description

This function shows a collaboration participant. The "collaboration_participant" parameter must be used to specify the target collaboration participant (as an "ezcollaborationItemparticipantlink" object). The optional "view" parameter can be used to specify the view mode (for example "text_linked"). All other parameters (of any type) will be passed on and thus become available as template variables in the included template.

5

collaboration_simple_message_view

Summary

Outputs the view template for a collaboration message.

Usage

```
{collaboration_simple_message_view
  sequence=sequence
  is_read=status
  item_link=message_link
  collaboration_message=simple_message
[ view=mode ]
[ parameter=value [...] ]}
```

Parameters

Name	Type	Description	Required
collaboration_message	object	An eZCollaborationSimpleMessage object.	Yes.
sequence	string	Display sequence value	Yes.
is_read	boolean	TRUE if the message has been read, FALSE if not.	Yes.
item_link	object	eZCollaborationItemMessageLink object.	Yes.
view	string	The view mode that should be used.	No.
other parameters	mixed	Parameter(s) that will be passed to the included template.	No.

Description

This function shows the interface for a collaboration message.

5

collaboration_view_gui

Summary

Outputs the template for a collaboration item.

Usage

```
{collaboration_view_gui item_class=class
                        collaboration_item=item
                    [ view=mode ]
                    [ parameter=value [...] ]}
```

Parameters

Name	Type	Description	Required
item_class	string	The item class.	Yes.
collaboration_item	object	The collaboration item.	Yes.
view	string	The view mode that should be used.	No.
parameter	any	Parameter(s) that will be passed to the included template.	No.

Description

This function shows the view interface for a collaboration item. The class and the target object must be specified using the "item_class" and the "collaboration_item" parameters. The "view" parameter is optional, it can be used to specify a desired view. All other parameters (of any type) will be passed on and thus become available as template variables in the included template.

5

content_pdf_gui

Summary

Outputs the PDF template for a content object (DEPRECATED).

Usage

```
{content_pdf_gui content_object=object [ view=reserved ] [ parameter=value [
... ] ]}
```

Parameters

Name	Type	Description	Required
content_object	object	The target content object.	Yes.
view	string	Reserved for future use.	No.
parameter	any	Parameter(s) that will be passed to the included template.	No.

Description

This function shows the PDF interface for a content object. The target object must be specified (as an "ezcontentobject" (page 1032) object) using the "content_object" parameter. The "view" parameter is reserved for future use. All other parameters (of any type) will be passed on and thus become available as template variables in the included template.

5

content_version_view_gui

Summary

Outputs a view template for a content object version.

Usage

```
{content_version_view_gui content_version=version [ view=mode ]  [
parameter=value [...] ]}
```

Parameters

Name	Type	Description	Required
content_version	object	The target version.	Yes.
view	string	The view mode that should be used.	No.
parameter	any	Parameter(s) that will be passed to the included template.	No.

Description

This function shows the view interface for a content object version. The target version must be specified (as a "ezcontentobjectversion" (page 1056) object) using the "content_version" parameter. The "view" parameter is optional, it can be used to specify a desired view (for example "full", "plain", "text", etc.). The function will attempt to use the following template from within either the current design or one of the fallback designs: "templates/content/version/view/[name_of_view_mode].tpl". All other parameters (of any type) will be passed on and thus become available as template variables in the included template.

5

content_view_gui

Summary

Outputs a view template for a content object.

Usage

```
{content_view_gui content_object=object [ view=view ] [ parameter=value [...]
]}
```

Parameters

Name	Type	Description	Required
content_object	object	The target content object.	Yes.
view	string	The view mode that should be used.	No.
other parameters	any	Parameter(s) that will be passed to the included template.	No.

Description

This function shows the view interface for a content object. The target object must be specified (as an "ezcontentobject" (page 1032) object) using the "content_object" parameter. The "view" parameter is optional, it can be used to specify a desired view (for example "text", "text_linked", "embed", etc.). The function will attempt to use the following template from either the current design or one of the fallback designs: "templates/content/view/[name_of_view_mode].tpl". All other parameters (of any type) will be passed on and thus become available as template variables in the included template.

5

event_edit_gui

Summary

Outputs the edit template for a workflow event.

Usage

```
{event_edit_gui event=event [ parameter=value [...] ] }
```

Parameters

Name	Type	Description	Required
event	object	The target workflow event object.	Yes.
parameter	any	Parameter(s) that will be passed to the included template.	No.

Description

This function shows the edit interface for a content object attribute. The target workflow event must be specified using the "event" parameter. All other parameters (of any type) will be passed on and thus become available as template variables in the included template.

5

node_view_gui

Summary

Outputs the view template for a node.

Usage

```
{node_view_gui content_node=node [ view=view_mode  [ parameter=value [ ... ] ]
]}
```

Parameters

Name	Type	Description	Required
content_node	object	The target node (as an ezcontentob-jecttreenode object).	Yes.
view	string	The view mode that should be used.	Yes.
parameter	any	Parameter(s) that will be passed to the included template.	No.

Description

This function makes it possible to display a node using its view (or override) template. The target node must be specified as an "ezcontentobjecttreenode" (page 1051) object using the "content_node" parameter. The "view" parameter specifies which view mode that should be used. All other parameters (of any type) will be passed on and thus become available as template variables in the view template.

Examples

Example 1

```
{def $my_node=fetch( 'content', 'node', hash( 'node_id', 96 ) )}
{node_view_gui view='example' content_node=$my_node}
```

In this example, node number 96 is fetched and stored in $my_node. The "node_view_gui" function is used to display the target node using the "example" view mode. If there are no override rules for the specified view mode, the system will search for "example.tpl" in the "templates/node/view/" directory of the current design. If the requested template file is not found, eZ Publish will continue searching for it in the fallback designs and the standard design.

5

related_view_gui

Summary

Not documented yet.

Description

Not documented, the templates seem to be missing from the distribution(s).

5

shop_account_view_gui

Summary

Outputs the view template for a specified order.

Usage

```
{shop_account_view_gui order=order [ view=view [ parameter=value [ ... ] ] ]}
```

Parameters

Name	Type	Description	Required
order	object	The target order object.	Yes.
view	string	The view mode that should be used.	No.
other parameters	any	Parameter(s) that will be passed to the included template.	No.

Description

This function shows the view interface for an order. The order must be specified (as a "ezorder" (page 1097) object) using the "order" parameter. The "view" parameter is optional, it can be used to select a desired view mode: either "html" or "ascii". All other parameters (of any type) will be passed on and thus become available as template variables in the included template.

5

tool_bar

Summary

Outputs the template for a toolbar.

Usage

```
{tool_bar name=name view=view [ parameter=value [...] ]}
```

Parameters

Name	Type	Description	Required
name	string	The name of the toolbar.	Yes.
view	string	The view mode that should be used.	Yes.
parameter	any	Parameter(s) that will be passed to the included template.	No.

Description

This function can be used to display a toolbar template. The name parameter must be used to specify the name of the toolbar that should be show. The default/standard names are "top", "right" and "bottom" - custom names are also allowed. The "view" parameter can be used to specify a desired view mode ("line", "full", etc.). All other parameters (of any type) will be passed on and thus become available as template variables in the included template.

Examples

Example 1

```
{tool_bar name='top' view='line'}
```

Shows a toolbar called "top" using the "line" view mode.

5

5.10 Template control structures

The template control structures are documented in the following sections:

- Conditional control (page)
- Looping (page)
- Deprecated (page)

5

5.10.1 Conditional control

if (page 1430)

  Allows conditional control by the way of an IF-THEN-ELSE mechanism.

switch (page 1432)

  Allows conditional control of code execution.

5

if

Summary

Allows conditional control by the way of an IF-THEN-ELSE mechanism.

Usage

```
{if <condition>}
...
[ {elseif <condition>} ]
...
[ {else} ]
...
{/if}
```

Description

This 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.

Examples

Example 1

```
{if eq( $var, 128 )}
    Hello world <br />
{/if}
```

If $var equals 128, the following output will be produced: "Hello world". If it does not equal 128, no output will be produced.

Example 2

```
{if eq( $var, 128 )}
    Hello world <br />
{else}
    No world here, move along. <br />
{/if}
```

If $var equals 128, the following output will be produced: "Hello world". If it does not equal 128, the output will be "No world here, move along.".

Example 3

```
{if eq( $fruit, 'apples' )}
    Apple tree <br />
{elseif eq( $fruit, 'oranges' )}
    Orange juice <br />
{else}
    Banana split <br />
{/if}
```

If $fruit equals "apples", the output will be "Apple tree", if it equals "oranges" then the output will be "Orange juice" - otherwise the output will be "Banana split".

5

switch

Summary

Allows conditional control of code execution.

Usage

```
{switch match=<variable>}

    {case match=<value>}
    {case in=<array>}
    ...
    {/case}

    {case}
    ...
    {/case}

{/switch}
```

Description

This 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. A default case should always be provided.

Please note that it is also possible to match inside arrays. This can be done by making use of the "in" argument, it is demonstrated in the last (third) example.

Examples

Example 1

```
{def $fruits='oranges'}

{switch match=$fruits}

    {case match='apples'}
        Apples <br />
    {/case}

    {case match='oranges'}
        Oranges <br />
```

5

```
    {/case}

    {case}
        Unidentified fruit! <br />
    {/case}

{/switch}
```

The following output will be produced: "Oranges".

Example 2

```
{def $fruits='Hello world'}

{switch match=$fruits}

    {case match='apples'}
        Apples <br />
    {/case}

    {case match='oranges'}
        Oranges <br />
    {/case}

    {case}
        Unidentified fruit! <br />
    {/case}

{/switch}
```

The following output will be produced: "Unidentified fruit!" - which is the outcome of the default case (none of the other cases matched).

Example 3

```
{def $digit=1}

{switch match=$digit}

    {case in=array( 1, 2 )}
        This one matches.
    {/case}
```

```
    {case in=array( 2, 3 )}
        This one does not match.
    {/case}

    {case}
        Not this one either.
    {/case}

{/switch}
```

The following output will be produced: "This one matches.".

5.10.2   Looping

do   (page 1436)
>   Creates a do...while loop.

for   (page 1437)
>   Creates a generic for loop.

foreach   (page 1438)
>   Iterates over arrays in different ways.

while   (page 1441)
>   Creates a while loop.

5

do

Summary

Creates a do...while loop.

Usage

```
{do}
    [ {delimiter}...{/delimiter} ]
    [ {break}    ]
    [ {continue} ]
    [ {skip}     ]
{/do while <condition> [ sequence <array> as $seqVar ]}
```

Description

This mechanism is very similar to the "while" (page 1438) construct, 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). It supports breaking, continual and skipping.

Examples

Example 1

```
{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)
```

5

for

Summary

Creates a generic for loop.

Usage

```
{for <number> to <number> as $itemVar [ sequence <array> as $seqVar ]}
    [ {delimiter}...{/delimiter}]
    [ {break}    ]
    [ {continue} ]
    [ {skip}     ]
{/for}
```

Description

This mechanism makes it possible to do generic looping. It supports looping over numerical ranges in both directions. In addition it also supports breaking, continual and skipping.

Examples

Example 1

```
{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
```

5

foreach

Summary

Iterates over arrays in different ways.

Usage

```
{foreach <array> as [ $keyVar => ] $itemVar
        [ sequence <array> as $sequenceVar ]
        [ offset <offset>                  ]
        [ max <max>                        ]
        [ reverse                          ]}

    [ {delimiter}...{/delimiter} ]
    [ {break}    ]
    [ {continue} ]
    [ {skip}     ]

{/foreach}
```

Description

This construct makes it possible to iterate over arrays in different ways. The loop can be tweaked using parameters:

Parameter	Description	Value
sequence	Defines a sequence which is iterated together with the normal loop parameter (see example below)	array
offset	Determines the start of the loop array for the iterations	integer
max	Determines the maximum number of iterations	integer
reverse	The loop is performed in the reverse order if this is set	

There are also a set of inline template loop control structures available, these need to be a direct child of the foreach structure:

	Description	Value
delimiter	Determines a block of template elements which should be placed in between two iterations	mixed
skip	Jumps to next iteration, does	

	NOT output delimiter	
breaks	Breaks out of the foreach	
continue	Jumps to next iteration, but outputs delimiter	

Examples

Example 1

```
{foreach $objects as $object}

    {$object.name} <br />

{/foreach}
```

This example 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
```

Example 2

```
{foreach $objects as $index => $object}

    {$index} : {$object.name} <br />

{/foreach}
```

This example demonstrates how to create an iteration counter.

```
0: Emmett Brown
1: Marty McFly
2: Lorraine Baines
3: Biff Tannen
```

Example 3

5

```
{foreach $objects as $object sequence array( 'dark', 'light' ) as $style}

    <div class="{$style}">{$object.name}</div>

{/foreach}
```

This example demonstrates how to create a loop where the iterations are displayed using alternating styles (in this case dark, light, dark, light and so on).

5

while

Summary

Creates a while loop.

Usage

```
{while <condition> [ sequence <array> as $seqVar ]}
    [ {delimiter}...{/delimiter} ]
    [ {break}    ]
    [ {continue} ]
    [ {skip}     ]
{/while}
```

Description

This 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.

Examples

Example 1

```
{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)
```

5

5.10.3   Deprecated

section   (page 1443)

5

section

Summary

Deprecated looping, branching, etc.

Description

Refer to the documentation of the old template syntax for more information about this control structure.

Please note that this control structure has been deprecated and should not be used. It is included here only because some of the default templates in the distributions are still using it.

5

5.11   Template override conditions

This section contains an overview of the override conditions that can be used to override the system templates.

Override example

```
[magic_pockets]                 (1)
Source=node/view/full.tpl       (2)
MatchFile=magic_pocket.tpl      (3)
Subdir=templates                (4)
Match[class_identifier]=pocket  (5)
Match[section]=34               (6)
...                             ...
```

1. The name of the override.

2. The template that should be overridden.

3. The alternate template that should be used.

4. The location of the override templates.

5. Match condition #1

6. Match condition #2

The following sections reveal the override rules for the different template files:

\begin{itemize}   \item   class/edit.tpl\refer{label899821738581b8dd033bc446afc55ffe} \item   class/groupedit.tpl\refer{label04524bac993153f9fe970fe7676715e9}   \item class/view.tpl\refer{labelfbc362e16464c6747e9f40357b48b6e4}   \item   content/ advancedsearch.tpl\refer{label10f6aea1b664f5b8da63ea98648b78ca}   \item   content/browse.tpl\refer{label25dcd2315e4d88d4b95938e3300f198b}   \item   content/ collectedinfo/*.tpl\refer{labelbd7db2116d3594da6e9fe344fb7086e7}   \item   content/collectedinfo/*.tpl\refer{label56eb4696eac8b2f6b7d4a28eeed02442}   \item content/collectedinfomail/*.tpl\refer{labeld85fcb18ebb5282ca93e8dbe25569006} \item   content/datatype/edit/*.tpl\refer{labelec7636b5d141dda72e1d72859bedfcce} \item   content/datatype/view/*.tpl\refer{label43fb8235bab72e311c0c75fd98b79259} \item   content/edit.tpl\refer{label875108e2e9bb4414116a9b30e4f29e3a}   \item content/history.tpl\refer{label13f9ad1ba488cd7518904b187f289bca}   \item   content/queued.tpl\refer{label6239fb5b2130aca3ded040b3b3cd5abb}   \item   content/search.tpl\refer{labelb1290846b194717d32af5147dbd297d1}   \item   content/tipafriend.tpl\refer{labeladd5d2a048ee86eb76c34e355d505709}   \item content/tipafriendmail.tpl\refer{labele8cb72c81575ba7f1585fe0c52ed9e99} \item   content/versions.tpl\refer{labelf019809bbc7590a56c9ed62738cb79b6}

\item   content/versionview.tpl\refer{label967831a0d93cdb5c3f5f26ed64e4523c}
\item   content/view/*.tpl\refer{labelea381300a7a8fe49d602d3d04b13d7ae}
\item   layout/set.tpl\refer{labelce985df557f2d8c044535d1468b8ce4a}   \item
node/view/*.tpl\refer{label75bdfc772eb05c3a358af71a9e665bda}   \item
node/view/pdf.tpl\refer{label87bb760d6e50278cd3b04e42a68f4ada}   \item
pagelayout.tpl\refer{label7ab549b4632831f5a72b556de8399ea9}   \item   work-
flow/edit.tpl\refer{label6f64d13aaba84a8797b59a27cdc63f66}   \item   workflow/
groupedit.tpl\refer{label056ad454965254b15acd3a24b4d7a16b}   \item   workflow/
view.tpl\refer{label346ada60d6e43530190adf4befab1c49} \end{itemize}

5.11.1 node/view/*.tpl

Module/view: view

Condition	Description
object	Matches the ID number of the object.
node	Matches the ID number of the node.
parent_node	Matches the ID number of the parent node.
class	Matches the ID number of the class which the object is an instance of.
class_identifier	Matches the identifier of the class which the object is an instance of.
view_offset	Matches the offset view parameter.
depth	Matches the depth of the node. The depth of a top level node is 1.
section	Matches the number of the section which the object belongs to.
section_identifier	Matches the identifier of the section which the object belongs to. Available since eZ Publish 4.4.
navigation_part_identifier	Matches the identifier of the navigation part.
viewmode	Matches the name of the view mode (full, line, etc.)
url_alias	Matches the virtual URL of the node.
class_group	Matches the ID number of the group that the class which the object is an instance of belongs to.
parent_class	Matches the ID number of the parent class.
parent_class_identifier	Matches the identifier of the parent class.

5

5.11.2 content/search.tpl

Module/view: search

Condition	Description
section	Matches the number of the section in which the search was conducted.
section_identifier	Matches the identifier of the section which the object belongs to. Available since eZ Publish 4.4

5

5.11.3   content/edit.tpl

Module/view: edit

Condition	Description
object	Matches the ID number of the object.
class	Matches the ID number of the class which the object is an instance of.
class_identifier	Matches the identifier of the class which the object is an instance of.
section	Matches the number of the section which the object belongs to.
section_identifier	Matches the identifier of the section which the object belongs to. Available since eZ Publish 4.4.
class_group	Matches the ID number of the group that the class which the object is an instance of belongs to.

5

5.11.4   content/advancedsearch.tpl

Module/view: advancedsearch

Condition	Description
section	Matches the number of the section in which the search was conducted.
section_identifier	Matches the identifier of the section which the object belongs to. Available since eZ Publish 4.4.

5

5.11.5 content/versions.tpl

Module/view: versions

Condition	Description
object	Matches the ID number of the object.
class	Matches the ID number of the class which the object is an instance of.
class identifier	Matches the identifier of the class which the object is an instance of.
section id	Matches the number of the section which the object belongs to.

5

5.11.6 content/browse.tpl

Module/view: browse

Condition	Description
object	Matches the ID number of the object.
node	Matches the ID number of the node.
parent_node	Matches the ID number of the parent node.
class	Matches the ID number of the class which the object is an instance of.
class_identifier	Matches the identifier of the class which the object is an instance of.
view_offset	Matches the offset view parameter.
depth	Matches the depth of the node. The depth of a top level node is 1.
navigation_part_identifier	Matches the identifier of the navigation part.
url_alias	Matches the virtual URL of the node.

5

5.11.7 content/versionview.tpl

Module/view: versionview

Condition	Description
navigation_part_identifier	Matches the identifier of the navigation part.

5

5.11.8 node/view/pdf.tpl

Module/view: pdf

Condition	Description
object	Matches the ID number of the object.
node	Matches the ID number of the node.
parent_node	Matches the ID number of the parent node.
class	Matches the ID number of the class which the object is an instance of.
class_identifier	Matches the identifier of the class which the object is an instance of.
depth	Matches the depth of the node. The depth of a top level node is 1.
url_alias	Matches the virtual URL of the node.
class_group	Matches the ID number of the group that the class which the object is an instance of belongs to.

5

5.11.9   content/collectedinfo/*.tpl

Module/view: collectinformation

Condition	Description
object	Matches the ID number of the object.
node	Matches the ID number of the node.
parent_node	Matches the ID number of the parent node.
class	Matches the ID number of the class which the object is an instance of.
class_identifier	Matches the identifier of the class which the object is an instance of.
depth	Matches the depth of the node. The depth of a top level node is 1.
navigation_part_identifier	Matches the identifier of the navigation part.
url_alias	Matches the virtual URL of the node.

5

5.11.10   content/collectedinfomail/*.tpl

Module/view: collectinformation

Condition	Description
object	Matches the ID number of the object.
node	Matches the ID number of the node.
parent_node	Matches the ID number of the parent node.
class	Matches the ID number of the class which the object is an instance of.
class_identifier	Matches the identifier of the class which the object is an instance of.
depth	Matches the depth of the node. The depth of a top level node is 1.
navigation_part_identifier	Matches the identifier of the navigation part.
url_alias	Matches the virtual URL of the node.

5

5.11.11   content/collectedinfo/*.tpl

Module/view: collectedinfo

Condition	Description
object	Matches the ID number of the object.
node	Matches the ID number of the node.
parent_node	Matches the ID number of the parent node.
class	Matches the ID number of the class which the object is an instance of.
depth	Matches the depth of the node. The depth of a top level node is 1.
navigation_part_identifier	Matches the identifier of the navigation part.
url_alias	Matches the virtual URL of the node.

5

5.11.12 class/edit.tpl

Module/view: edit

Condition	Description
class	Matches the ID number of the class.

5.11.13   class/groupedit.tpl

Module/view: groupedit

Condition	Description
object	Matches the ID number of class group.

5

5.11.14   class/view.tpl

Module/view: view

Condition	Description
class	Matches the ID number of the class.
class_identifier	Matches the identifier of the class (for example "folder").

5

5.11.15 workflow/edit.tpl

Module/view: edit

Condition	Description
workflow	Matches the ID number of the workflow.

5

5.11.16 workflow/groupedit.tpl

Module/view: groupedit

Condition	Description
workflowgroup	Matches the ID number of the workflow group.

5

5.11.17 workflow/view.tpl

Module/view: view

Condition	Description
workflow	Matches the ID number of the workflow.

5

5.11.18   layout/set.tpl

Module/view: set

Condition	Description
layout	Matches the name of the layout (for example "print", "fullpage", etc.).

5

5.11.19 pagelayout.tpl

Module/view: none.

Condition	Description
persistent_variable	Matches the value of the persistent variable if the "view" view of the "content" module was executed. (The persistent variable should be set in the "node/view/<view_mode>.tpl" template that was used by this view.)
[any]	The pagelayout may be overridden using the same keys that are available for the view which was used within the request. For example, if the "view" view of the "content" module was accessed, then the override conditions of that view will be matched. This makes it possible to create a pagelayout override that will be triggered when, for example, nodes referencing objects that belong to a certain section are requested.

5

5.11.20 content/datatype/edit/*.tpl

Module/view: none.

Condition	Description
class_identifier	Matches the identifier of the class.
attribute_identifier	Matches the identifier of the attribute.

5

5.11.21   content/datatype/view/*.tpl

Module/view: none.

Condition	Description
class_identifier	Matches the identifier of the class.
attribute_identifier	Matches the identifier of the attribute.

5

5.11.22   content/view/*.tpl

Module/view: view

Condition	Description
object	Matches the ID number of the object.
class	Matches the ID number of the class which the object is an instance of.
class_identifier	Matches the identifier of the class which the object is an instance of.
section	Matches the number of the section which the object belongs to.
section_identifier	Matches the identifier of the section which the object belongs to. Available since eZ Publish 4.4.
class_group	Matches the ID number of the group that the class which the object is an instance of belongs to.

5

5.11.23   content/tipafriend.tpl

Module/view: tipafriend

Condition	Description
object	Matches the ID number of the object.
class	Matches the ID number of the class which the object is an instance of.
class_identifier	Matches the identifier of the class which the object is an instance of.
class_group	Matches the ID number of the group that the class which the object is an instance of belongs to.
section	Matches the number of the section which the object belongs to.
section_identifier	Matches the identifier of the section which the object belongs to. Available since eZ Publish 4.4.
node	Matches the ID number of the node.
parent_node	Matches the ID number of the parent node.
depth	Matches the depth of the node. The depth of a top level node is 1.
url_alias	Matches the virtual URL of the node.

5

5.11.24   content/tipafriendmail.tpl

Module/view: none.

Condition	Description
object	Matches the ID number of the object.
class	Matches the ID number of the class which the object is an instance of.
class_identifier	Matches the identifier of the class which the object is an instance of.
class_group	Matches the ID number of the group that the class which the object is an instance of belongs to.
section	Matches the number of the section which the object belongs to.
section_identifier	Matches the identifier of the section which the object belongs to. Available since eZ Publish 4.4.
node	Matches the ID number of the node.
parent_node	Matches the ID number of the parent node.
depth	Matches the depth of the node. The depth of a top level node is 1.
url_alias	Matches the virtual URL of the node.

5

5.11.25 content/history.tpl

Module/view: history

Condition	Description
object	Matches the ID number of the object.
class	Matches the ID number of the class which the object is an instance of.
class_identifier	Matches the identifier of the class which the object is an instance of.
section	Matches the number of the section which the object belongs to.
section_identifier	Matches the identifier of the section which the object belongs to. Available since eZ Publish 4.4.

5

5.11.26 content/queued.tpl

Module/view: none.

Condition	Description
object	Matches the content ID number of the object.
remote_id	Matches the remote ID number of the object.
class	Matches the ID number of the content class which the object is an instance of.
class_identifier	Matches the identifier of the content class which the object is an instance of.
class_group	Matches the identifier of the class group which the content class is an instance of.
state	Matches the content object state id.
state_identifier	Matches the content object state identifier.
section	Matches the section ID.
section_identifier	Matches the section identifier

5

5.12 Template fetch functions

The template fetch functions are documented in the following sections:

\begin{itemize} \item{} class\refer{labelc146a473e33839db3a5fc701ae1e3cff}
\item{} collaboration\refer{labelcdbffe230491e005f12fe18ad35b290c}
\item{} content\refer{labelbcc7790f765a2f37194a891e41311b12}
\item{} error\refer{label972fbecf33e1466485241fffaa0c244f} \item{}
ezinfo\refer{label1ac35587886356901005765e184af596} \item{}
form\refer{labela16f94e5a7b4c527f3052dda925eac1a} \item{}
infocollector\refer{label8ac1efacf19076e81688deca52f1f5b9} \item{}
layout\refer{label591b2f550b6268bddcef86f4c2213768} \item{}
notification\refer{labelcbe8851bddcbcc298ea473b72ab483fd} \item{}
package\refer{label2a67ee5fef16f267188ff312b9d36538} \item{}
pdf\refer{label6c2763cc00da3c12b89073fdbf39a56e} \item{} reference\refer{labelb47ab678ae94b1db69}
\item{} role\refer{label94182e871cec747f7b9ec971d783412a} \item{}
rss\refer{label29f5d5ac43621cb5e38ee57a0dc4f00f} \item{} search\refer{label49430a14de263750a46fb0
\item{} section\refer{label2a3a1534fea5512800f257e40994b329}
\item{} setup\refer{labelb05971b446ee1cd4ba485c20d12902e1}
\item{} shop\refer{labela15950e81574941c6276c8cdcd8c80f3}
\item{} trigger\refer{labelf19407e4a0462b6c4dcd036e8e185731}
\item{} url\refer{label173fb25b4957feb67b5dc718c710878a} \item{}
user\refer{labele9f8960664d8112cc37534696dea1c84} \item{}
workflow\refer{labelc4dfa05ca43cea88f846537dbafa6f9c} \end{itemize}

5

5.13 Template PDF functions

anchor (page 1475)
> Creates an internal link anchor (DEPRECATED).

create_index (page 1476)
> Creates an index based on the specified keywords (DEPRECATED).

filled_circle (page 1477)
> Creates a filled circle (DEPRECATED).

filled_rectangle (page 1479)
> Creates a filled rectangle (DEPRECATED).

footer (page 1481)
> Sets the footer (DEPRECATED).

footer_block (page 1483)
> Inserts complex footers (DEPRECATED).

frame_header (page 1484)
> Sets the header text (DEPRECATED).

frontpage (page 1486)
> Adds content to the frontpage (DEPRECATED).

header (page 1487)
> Inserts a header / title (DEPRECATED).

header_block (page 1489)
> Inserts a complex header (DEPRECATED).

image (page 1490)
> Inserts an image to the PDF document (DEPRECATED).

keyword (page 1492)
> Adds a keyword to the keyword index (DEPRECATED).

line (page 1493)
> Draws a line (DEPRECATED).

link (page 1495)
> Inserts an external link (DEPRECATED).

new_line (page 1496)
> Inserts a new line (DEPRECATED).

new_page (page 1497)
> Inserts a new page (DEPRECATED).

page_number (page 1498)
> Starts the page number counter (DEPRECATED).

5

set_font   (page 1499)
>   Changes the default font (DEPRECATED).

set_margin   (page 1501)
>   Sets the page margins or line spacing (DEPRECATED).

strike   (page 1502)
>   Inserts striked text (DEPRECATED).

table   (page 1503)
>   Inserts a table (DEPRECATED).

text   (page 1505)
>   This function inserts formatted text into the PDF document (DEPRECATED).

text_box   (page 1508)
>   Inserts text at a specified location (DEPRECATED).

text_frame   (page 1509)
>   Inserts a text with a frame (DEPRECATED).

toc   (page 1511)
>   Inserts a generated table of contents (DEPRECATED).

ul   (page 1512)
>   Inserts a bullet list (DEPRECATED).

5

5.13.1 anchor

Summary

Creates an internal link anchor (DEPRECATED).

Usage

```
{pdf( anchor, label )}
```

Parameters

Name	Type	Description	Required
label	string	The name of the anchor.	Yes.

Description

This function does not work. It is supposed to create an anchor for making internal links in PDF documents.

Examples

Example 1

```
{pdf( 'anchor', 'image_1' )}
```

5

5.13.2   create_index

Summary

Creates an index based on the specified keywords (DEPRECATED).

Usage

```
pdf( create_index )
```

Description

This function creates an index page from a collection of keywords. The keywords must be specified using the "keyword" (page 1492) PDF template function. The index will start on the next page, and consists of the specified keywords with their corresponding page number.

Examples

Example 1

```
{pdf( 'text', 'Index page demo.' | wash( 'pdf' ) )}
{pdf( 'keyword', 'apples' )}
{pdf( 'keyword', 'bananas' )}
{pdf( 'create_index' )}
```

This example creates an index page using the following keywords "apples" and "bananas".

5

5.13.3   filled_circle

Summary

Creates a filled circle (DEPRECATED).

Usage

```
pdf( filled_circle, hash( radius, circle_radius,
                          x,      x_position,
                          y,      y_position,
                        [ rgb,    rgb_color, ]
                        [ cmyk,   cmyk_color ] ) )
```

Parameters

Name	Type	Description	Required
radius	integer	The radius of the circle.	Yes.
x	integer	The x position of the circle placement.	Yes.
y	integer	The y position of the circle placement.	Yes.
rgb	array	The RGB color of the circle.	Only if the "cmyk" parameter is omitted.
cmyk	array	The CMYK color of the circle.	Only if the "rgb" parameter is omitted.

Description

This function creates a filled circle with a given radius and color. The position of the circle is specified with the "x" and "y" parameters. The given coordinate must be within the page margins. The lower-left corner is at (0,0).

Examples

Example 1

```
{pdf( 'new_page' )}

{pdf( 'filled_circle', hash( 'radius', 100,
                            'x',      300,
                            'y',      400,
                            'rgb',    array( 0, 0, 0 ) ) )}

{pdf( 'filled_circle', hash( 'radius', 98,
                            'x',      300,
                            'y',      400,
                            'rgb',    array( 255, 0, 0 ) ) )}
```

This example creates a black circle with red filling on a new page.

5

5.13.4   filled_rectangle

Summary

Creates a filled rectangle (DEPRECATED).

Usage

```
pdf( filled_rectangle, hash( x,          x_position,
                             y,          y_position,
                             width,      width,
                             height,     height,
                           [ rgb,        rgb_color,         ]
                           [ rgbTop,     rgb_top_color,
                             rgbBottom,  rgb_bottom_color,  ]
                           [ cmyk,       cmyk_color,        ]
                           [ cmykTop,    cmyk_top_color,
                             cmykBottom, cmyk_bottom_color, ] ) )
```

Parameters

Name	Type	Description	Required
x	integer/ float	X coordinate.	Yes.
y	integer/ float	Y coordinate.	Yes.
width	integer/ float	Rectangle width.	Yes.
height	integer/ float	Rectangle height.	Yes.
rgb	array	Rectangle RGB color.	No.
rgbTop	array	Rectangle top RGB color.	No.
rgbBottom	array	Rectangle bottom RGB color.	No.
cmyk	array	Rectangle CMYK color.	No.
cmykTop	array	Rectangle top CMYK color.	No.
cmykBottom	array	Rectangle bottom CMYK color.	No.

Description

This function creates a filled rectangle. The "x" and "y" parameters specify the lower-left corner of the rectangle. The "width" and "height" parameters determine the size of the rectangle.

The color of the rectangle can be specified using either the "rgb" or the "cmyk" parameter. The "rgb" parameter must be an array consisting of three numbers between 0 and 255. The "cmyk" parameter must be an array of four floats between 0 and 1. The "rgb" or "cmyk" parameter

fills the entire rectangle with the same color; no color interpolation. This option has the same result as setting the "rgbTop", "rgbBottom" or "cmykTop", "cmykBottom" with the same color. A different top and bottom color results in a gradient fill. The color will be linearly interpolated from the top color to the bottom color of the rectangle.

This function uses features that appeared in the PDF1.3 specification. Because of this, rectangles will not visible in PDF-viewers that are not PDF 1.3 compliant (for example "Ghostview", etc.).

Examples

Example 1

```
{pdf( 'filled_rectangle', hash( 'x',          -10,
                                'y',          30.5,
                                'width',      282.9,
                                'height,      821.9,
                                'cmykTop,     array( 0.96, 0.30, 0, 0 ),
                                'cmykBottom', array( 0.15, 0.04, 0, 0 ) ) )}

{pdf( 'filled_rectangle', hash( 'x',          272.9,
                                'y',          30.5,
                                'width',      333,
                                'height',     821.9,
                                'cmykTop',    array( 0.34, 0.05, 0, 0 ),
                                'cmykBottom', array( 0.10, 0.02, 0, 0 ) ) ) }

{pdf( 'filled_rectangle', hash( 'x',     -10,
                                'y',     -10,
                                'width', 606,
                                'height', 40.5,
                                'cmyk',   array( 0.89, 0.43, 0.01, 0 ) ) )}
```

This example fills the page with three different rectangles.

5

5.13.5   footer

Summary

Sets the footer (DEPRECATED).

Usage

```
pdf(footer, hash( [ text,       text,                                 ]
                  [ align,      text_alignment,                       ]
                  [ font,       font_name,                            ]
                  [ size,       font_size,                            ]
                  [ page,       page_interval,                        ]
                  [ pageOffset, page_offset,                          ]

                  [ margin,     hash( [ bottom, bottom_margin, ]
                                      [ left,   left_margin,   ]
                                      [ right,  right_margin,  ] ),       ]

                  [ line,       hash( [ leftMargin,  left_margin,    ]
                                      [ rightMargin, right_margin,   ]
                                      [ thickness,   line_thickness, ] ), ]

                  [ newline, boolean                                 ] ) )
```

5

Parameters

Name	Type	Description	Required
text	string	Text used in the footer.	No.
align	string	Alignment of the text.	No.
page	string	Page occurence is all, even or odd.	No.
pageOffset	integer	Page number where the footer starts.	No.
size	integer	Size of the font.	No.
font	string	Type of the font.	No.
margin	hash	The definition of the margins.	No.
left	integer/ float	Left footer margin.	No.
right	integer/ float	Right footer margin.	No.
bottom	integer/ float	Bottom footer margin.	No.
line	hash	The definition of a line.	No.
leftMargin	integer/ float	Left line margin.	No.
rightMargin	integer/ float	Right line margin.	No.
thickness	integer/ float	The thickness of the line.	No.
newline	boolean	Adds a new line to the footer. Force footer entry to new line	No.

Description

This function makes it possible to create a footer for a collection of pages. Please note that it does not work correctly. The page number from where the footer starts can be specified using the "pageOffset" parameter. The "page" parameters can be used to specify whether the footer should occur on "all", "even" or "odd" pages.

If the "text" parameter is used, the parameters "align", "type", and "size" can be used to change the default alignment, font type and size. Please refer to the "text" (page 1505) function for more information about these parameters.

There are two keywords reserved in the text string (parameter):

- #page: will be replaced by the current page number.

- #total: will be replaces by the total number of pages.

The footer should only be included once in the PDF document.

5.13.6 footer_block

Summary

Inserts complex footers (DEPRECATED).

Usage

```
{pdf( footer_block, hash( block_code, $variable ) )}
```

Parameters

Name	Type	Description	Required
block_code	mixed	Variable of template code set using "set_block".	Yes.

Description

This function makes it possible to create complex footers. There are some known problems with this function.

5

5.13.7   frame_header

Summary

Sets the header text (DEPRECATED).

Usage

```
{pdf(frame_header, hash( [ text, text, ]
              [ align, text_alignment, ]
              [ page, page_interval, ]
              [ pageOffset, page_offset, ]
              [ size, font_size, ]
              [ font, font_name, ]
              [ margin, hash( [ bottom, bottom_margin, ]
                      [ left, left_margin, ]
                      [ right, right_margin, ] ), ]
              [ line, hash( [ leftMargin, left_margin, ]
                      [ rightMargin, right_margin, ]
                      [ thickness, line_thickness, ] ), ] ) ) }
```

Parameters

Name	Type	Description	Required
text	string	The header text.	No.
align	string	The text alignment.	No.
page	string	Page occurence ("all", "even" or "odd").	No.
pageOffset	integer	Page number to start footer at.	No.
size	integer	The font size.	No.
font	string	The font name.	No.
margin	hash	Margin definition.	No.
left	integer/ float	Left header margin.	No.
right	integer/ float	Right header margin.	No.
bottom	integer/ float	Bottom footer margin.	No.
line	hash	Line definition.	No.
leftMargin	integer/ float	Left line margin.	No.
rightMargin	integer/ float	Right line margin.	No.
thinkness	integer/ float	Line thickness.	No.
newline	boolean	Force header entry to new line.	No.

Description

This function is deprecated and should not be used.

5.13.8 frontpage

Summary

Adds content to the frontpage (DEPRECATED).

Usage

```
pdf( frontpage, hash( text,        page_text,
                    [ align,      text_alignment,  ]
                    [ size,       text_size,       ]
                    [ top_margin, top_text_margin  ] ) )
```

Parameters

Name	Type	Description	Required
text	string	Text that will be added to the front page.	Yes.
align	string	The text alignment.	No.
size	int	The font size.	No.
top_margin	int	The top margin.	No.

Description

This function makes it possible to add text to the front page. If the frontpage is not present, a new front page will be created when the function is used for the first time. This function should be called after all other content has been added to the PDF document. The "align" and "size" parameters control the alignment and the size of the text. The "top_margin" parameter can be used to specify the top margin (vertical positioning) for the text.

Examples

Example 1

```
{pdf( 'frontpage', hash( 'text', 'eZ Publish
'|wash( 'pdf' ), 'align', 'center', 'size', 32, 'top_margin', 350) )}

{pdf( 'frontpage', hash( 'text', 'The road ahead'| wash( 'pdf' ), 'align',
'center', 'size', 22, 'top_margin', 400 ) )}
```

This example adds a main and a sub title to the front page. The sub title is smaller and positioned below the main title. Both titles are centered. Notice that the main title ends with a new line, this assures that the sub title is correctly centered.

5

5.13.9 header

Summary

Inserts a header / title (DEPRECATED).

Usage

```
pdf(header, hash( text,  header_text,
                  level, header_level,
                  size,  font_size,
                [ align, text_alignment, ]
                [ font,  font_type      ] ) )
```

Parameters

Name	Type	Description	Required
text	string	The header text.	Yes.
level	integer	The header level.	Yes.
size	integer	The font size.	Yes.
align	string	The text alignment.	No.
font	string	The font that should be used.	No.

Description

This function inserts a header or title. The main difference between the "text" and the header function is the "level" parameter. This parameter specifies the type of the header. The level number goes from 1 and up:

1. Chapters

2. Section

3. Subsection

4. Subsubsection

5. etc.

The level has nothing to do with the header text, it controls how the text appears in the table of contents (page). The font size, font type, and text alignment can be specified using the "size", "font", and "align" parameters. Please refer to the documentation of the "text" function for an explanation of these parameters.

5

Examples

Example 1

```
{pdf( 'header', hash( 'level', 1, 'text', 'The first chapter'|wash( 'pdf' ),
'size', 20 ) )}
{pdf( 'header', hash( 'level', 2, 'text', 'The first section'|wash( 'pdf' ),
'size', 16 ) )}
{pdf( 'header', hash( 'level', 2, 'text', 'The second section'|wash( 'pdf' ),
'size', 16 ) )}
{pdf( 'header', hash( 'level', 3, 'text', 'The first subsection'|wash( 'pdf'
), 'size', 14 ) )}
{pdf( 'header', hash( 'level', 1, 'text', 'The second chapter'|wash( 'pdf' ),
'size', 20 ) )}
```

This example shows how to create a chapters, section and subsections.

5

5.13.10 header_block

Summary

Inserts a complex header (DEPRECATED).

Usage

```
{pdf( header_block, hash( block_code, $variable ) )}
```

Parameters

Name	Type	Description	Required
block_code	mixed	Variable of template code set using the "set_block" function.	Yes.

Description

This function is not documented because there are some problems with it.

5

5.13.11   image

Summary

Inserts an image to the PDF document (DEPRECATED).

Usage

```
pdf( image, hash( src,     image_path,
                 [ width,  image_width,    ]
                 [ height, image_height,   ]
                 [ align,  justification,  ]
                 [ x,      x_position,     ]
                 [ y,      y_position,     ]
                 [ dpi,    resolution,     ]
                 [ static, flow_properties ] ) ) )
```

Parameters

Name	Type	Description	Required
src	string	The path to the image.	Yes.
width	integer	The width of the image.	No.
height	integer	The height of the image.	No.
align	string	The alignment of the image.	No.
x	integer	The absolute horizontal offset.	No.
y	integer	The absolute y offset.	No.
dpi	integer	The image resolution.	No.
static	boolean	Float properties of the image.	No.

Description

This function makes it possible to insert an image. The image location or image source points to the root directory of the eZ publish installation. The image location is relative to this directory. Currently, the supported image types are "jpg" and "png" without alpha channel.

The width and height of the image can be specified using the "width" and "height" parameters. If these parameters are omitted, the image will be scaled to 100 by 100 dots. The "align" parameter can be used to specify the horizontal position of the image. Possible alignments are "left", "right", and "center" - the default is "left". The "x" and "y" parameter makes it possible to place the image at an exact location. These parameters will override the settings from the image alignment ("align") and float properties ("static").

If the image is too sharp, the dots-per-inch can be decreased using the "dpi" parameter. (It is not possible to make the image sharper than the original.)

The "static" parameter can be used to specify whether the content around the image will float

5

around the image or not. When the "static" parameter is set to TRUE, additional content will not float around and may actually overlap the image.

Examples

Example 1

```
{pdf( 'text', 'The logo is on the right hand side of this text.'|wash( 'pdf' )
)}
{pdf( 'image', hash( 'src', 'design/mysite/images/company_logo.png',
                     'width', 200,
                     'height', 200,
                     'align', 'right' ) )}
```

This example displays an image with some text on the left hand side.

Example 2

```
{pdf( 'image', hash( 'src', $image.full_path, 'width', $image.width, 'height',
$image.height ) )}
```

This example adds the image assigned to the $image variable.

5

5.13.12 keyword

Summary

Adds a keyword to the keyword index (DEPRECATED).

Usage

```
pdf( keyword, word )
```

Parameters

Name	Type	Description	Required
word	string	The keyword that should be added.	Yes.

Description

This function adds a word to the keywords. When generating the keyword index (using the "create_index" (page 1476) function), an internal link to the position where the keyword was added will be created.

Examples

Example 1

```
{pdf( 'keyword', 'alien' )}
```

This example will add the "alien" word to the keywords.

5

5.13.13   line

Summary

Draws a line (DEPRECATED).

Usage

```
pdf( line, hash( x1,        x_start,
                 y1,        y_start,
                 x2,        x_stop,
                 y2,        y_stop,
               [ page,      occurence,    ]
               [ thickness, line_thickness ] ) )
```

Parameters

Name	Type	Description	Required
x1	float	Start coordinate of the X-ax.	Yes.
y1	float	Start coordinate of the Y-ax.	Yes.
x2	float	Stop coordinate of the X-ax.	Yes.
y2	float	Stop coordinate of the Y-ax.	Yes.
pages	string	The pages on which the line should appear.	No.
thickness	float	The thickness of the line.	No.

Description

This function draws a line. The line will be drawn from the (x1, y1) coordinate to the (x2, y2) coordinate. If the "pages" parameter is omitted or set to "current", the line will only be present on the current page. If "pages" is set to all, it will appear on all the pages. The thickness of the line can be set with the "thickness" parameter; the default thickness is 1.

When the line must be visible on all the pages, the line definition must be set after the content is written to the PDF-document (at the end of the template) and the line should be defined only once. A common technique is to use the "include" (page 1390) operator.

Examples

Example 1

```
{pdf( 'line', hash( 'x1', 100, 'y1', 100, 'x2', 100, 'y2', 300 ) )}
```

This example draws a vertical line, from (100, 100) to (100, 300) on the current page.

Example 2

```
{pdf( 'line', hash( 'x1', 20, 'y1', 30, 'x2', 20, 'y2', 40, pages, 'all' ) )}
{pdf( 'line', hash( 'x1', 20, 'y1', 40, 'x2', 30, 'y2', 40, pages, 'all' ) )}
{pdf( 'line', hash( 'x1', 30, 'y1', 40, 'x2', 30, 'y2', 35, pages, 'all' ) )}
{pdf( 'line', hash( 'x1', 25, 'y1', 35, 'x2', 33, 'y2', 35, pages, 'all' ) )}
{pdf( 'line', hash( 'x1', 33, 'y1', 35, 'x2', 33, 'y2', 27, pages, 'all' ) )}
{pdf( 'line', hash( 'x1', 25, 'y1', 35, 'x2', 25, 'y2', 27, pages, 'all' ) )}
{pdf( 'line', hash( 'x1', 25, 'y1', 27, 'x2', 33, 'y2', 27, pages, 'all' ) )}
{pdf( 'line', hash( 'x1', 20, 'y1', 30, 'x2', 25, 'y2', 30, pages, 'all' ) )}
```

This example draws the eZ logo in the lower left corner of every page.

5

5.13.14   link

Summary

Inserts an external link (DEPRECATED).

Usage

```
pdf( link, hash( url,  url,
                 text, link_text ) )
```

Parameters

Name	Type	Description	Required
url	string	The URL that should be called.	Yes.
text	string	The text of the link.	Yes.

Description

This function insert a link to an external document.

Examples

Example 1

```
{pdf( 'link', hash( 'url',  'http://www.ez.no',
                    'text', 'eZ Systems website' ) )}
```

This example creates an link to the eZ Systems website.

5

5.13.15 new_line

Summary

Inserts a new line (DEPRECATED).

Usage

```
pdf( new_line )
```

Description

This function inserts a new line.

Examples

Example 1

```
{pdf( 'text', 'Some text.'|wash( 'pdf' ) )}
{pdf( 'text', 'This sentence is written at the same line.'|wash( 'pdf' ) )}
{pdf( 'new_line' )}
{pdf( 'text', 'The start of a new line.'|wash( 'pdf' ) )}
{pdf( 'new_line' )}
{pdf( 'new_line' )}
{pdf( 'text', 'There is a blank line above this one.'|wash( 'pdf' ) )}
```

This example demonstrates everything that can be done with the "new_line" function.

5

5.13.16   new_page

Summary

Inserts a new page (DEPRECATED).

Usage

```
pdf( new_page )
```

Description

This function inserts a new page, following content will be placed on a new page. Please note that the system automatically takes care of adding new pages when needed. This function simply makes it possible to manually insert a new page.

Examples

Example 1

```
{pdf( 'text', 'Text that will be on page #1.' )}
{pdf( 'new_page' )}
{pdf( 'header', hash( 'text', 'Text that will be on page #2.', 'level', 1,
'size', 18 ) )}
```

This example demonstrates the manual insertion of a new page.

5

5.13.17 page_number

Summary

Starts the page number counter (DEPRECATED).

Usage

```
{pdf( page_number )}
```

Description

This function starts the page number counter.

5

5.13.18   set_font

Summary

Changes the default font (DEPRECATED).

Usage

```
pdf(set_font, hash( [ name,          font_name, ]
                    [ size,          font_size, ]
                    [ colorCMYK,     cmyk_color,]
                    [ colorRGB,      rgb_color, ]
                    [ justification, text_justification ] ) )
```

Parameters

Name	Type	Description	Required
name	string	The font type (name).	No.
size	integer	The size of the font.	No.
colorCMYK	array	The CMYK color.	No.
colorRGB	array	The RGB color.	No.
justification	string	The text alignment.	No.

Description

This function makes it possible to change the default/current font and the text color, style, etc. The following list shows the fonts that can be used.

- Courier-Bold

- Courier-BoldOblique

- Courier-Oblique

- Courier

- Helvetica-Bold

- Helvetica-BoldOblique

- Helvetica-Oblique

- Helvetic

- Symbol

- Times-Bold

5

- Times-BoldItalic

- Times-Italic

- Times-Roman

Either the "colorRGB" or the "colorCMYK" parameter can be used to specify the font color. The "colorRGB" parameter must be an array consisting of three integers between 0 and 255. The "colorCMYK" parameter must be an array of four decimal values between 0 and 1.

The justification parameter controls the text alignment, it can be:

- left

- right

- center

- full

Examples

Example 1

```
{pdf( 'set_font', hash( 'name', 'Times-Roman', 'size', 12 ) )}

{pdf( 'text', 'Hello world'|wash( 'pdf' ) )}

{pdf( 'new_line' )}
{pdf( 'new_line' )}
{pdf( 'set_font', hash( 'name', 'Times-Italic',
                        'size', 10,
                        'justification', 'full',
                        'colorRGB', array( 255, 255, 128 ) ) )}

{pdf( 'text', 'Example'|wash( 'pdf' ) )}
```

This example demonstrates how to change the default font type, size, style, and color.

5

5.13.19   set_margin

Summary

Sets the page margins or line spacing (DEPRECATED).

Usage

```
pdf( set_margin, hash( [ left,       left_margin,  ]
                       [ right,      right_margin, ]
                       [ top,        top_margin,   ]
                       [ bottom,     bottom_margin,]
                       [ x,          x_offset,     ]
                       [ y,          y_offset,     ]
                       [ line_space, line_space    ] ) )
```

Parameters

Name	Type	Description	Required
left	float	Left page margin.	No.
right	float	Right page margin.	No.
top	float	Top page margin.	No.
bottom	float	Bottom page margin.	No.
x	float	Page x offset.	No.
y	float	Page y offset.	No.
line_space	float	Line space size.	No.

Description

This function sets the page margins of the current and following pages. The "left", "right", "top", and "bottom" parameters specify the distances from the edges. The parameters "x" and "y" specify a point on the current page where the new margin starts. For the next pages, the parameters "x" and "y" are ignored and thus the margins will affect the entire area of the upcoming pages. The "line_space" parameter specifies the amount of white space between each line. The default margins are configured in the "pdf.ini" file.

Examples

Example 1

```
{pdf( 'set_margin', hash( 'left', 370, 'right', 100 ) )}
{pdf( 'text', 'The text written on this page is presented in a small column.
The space between each new line is also adjusted.'|wash( 'pdf' ) )}
```

This example creates a small column with some text.

5

5.13.20 strike

Summary

Inserts striked text (DEPRECATED).

Usage

```
pdf( strike, text )
```

Parameters

Name	Type	Description	Required
text	string	Text to strike through.	Yes.

Description

This function draws a line through the specified text. The "set_font" (page 1499) function is the only way to change the font type, font size, font color, and text alignment.

Examples

Example 1

```
{pdf( 'strike', 'A strike is a deliberate absence from work.'|wash( 'pdf' ) )}
```

This example demonstrates a striked sentence.

5

5.13.21 table

Summary

Inserts a table (DEPRECATED).

Usage

```
pdf(table, table_rows, hash( [ showLines,         number,                ]
                             [ firstRowTitle,     boolean,               ]
                             [ titleCellCMYK,     cmyk_color,            ]
                             [ titleTextCMYK,     cmyk_color,            ]
                             [ titleFontSize,     font_size,             ]
                             [ cellCMYK,          cmyk_color,            ]
                             [ textCMYK,          cmyk_color,            ]
                             [ rowGap,            row_gap,               ]
                             [ colGap,            col_gap,               ]
                             [ cellPadding,       cell_padding,          ]
                             [ width,             table_width,           ]
                             [ repeatTableHeader, repeat_table_header ] ) )
```

Parameters

Name	Type	Description	Required
showLines	integer	Defines the drawing style of the table.	No.
firstRowTitle	boolean	Defines whether the first row should be displayed as a title header (or not).	No.
titleCellCMYK	array	CMYK color of the title cell.	No.
titleTextCMYK	array	CMYK color of the title text.	No.
titleFontSize	integer	Font size of the title.	No.
cellCMYK	array	CMYK color of the cells.	No.
textCMYK	array	CMYK color of the text.	No.
rowGap	integer/ float	Space between the table rows.	No.
colGap	integer/ float	Space between the table columns.	No.
cellPadding	integer/ float	Padding around the text in each cell.	No.
width	integer/ float	Width of the table.	No.
repeatTableHeader	boolean	Defines whether the table header should be repeated on a new page.	No.

5

Description

This function inserts a table. The table data must defined using the "table_rows" parameter. Use the "set-block" (page 1405) template function to specify the rows and the cells. The rows and cells are defined the same way as rows and cells are defined in HTML. Note that nested tables are not supported.

The "showLines" parameter specifies the drawing style of the table, it must be one of the following numbers:

- 0 - Don't draw lines at all.

- 1 - Draw only the inner lines.

- 2 - Draw all the lines.

The "firstRowTitle" expects an boolean value. This parameter defines whether the first row in the table, should be used for column descriptions. With the "titleCellCMYK" and "titleTextCMYK" parameters the color of the cell-background and the text of the column titles can be controlled. The title font size can be changed with the "titleFontSize" parameter.

For the rest of the cells can the background and text color be changed with the "cellCMYK" and "textCMYK".

The size of the table and the space between the columns and rows is defined with the "rowGap", "colGap", "cellPadding", and "width" parameters.

The parameter "repeatTableHeader" defines whether the title should be repeated on each new page, if the table is stretched over multiple pages.

5

5.13.22   text

Summary

This function inserts formatted text into the PDF document (DEPRECATED).

Usage

```
pdf( text, page_text, hash( [ font,  font_name,     ]
                            [ size,  font_size,     ]
                            [ rgb,   rgb_color,     ]
                            [ cmyk,  cmyk_color,    ]
                            [ align, text_alignment ] ) )
```

Parameters

Name	Type	Description	Required
text	string	The text that should be inserted.	Yes.
font	string	The font type.	No.
size	integer	The size of the font.	No.
rgb	array	The RGB text color.	No.
cmyk	array	The CMYK text color.	No.
align	text	The text alignment.	No.

Description

This function inserts regular text into the PDF document. The font type, font size, font color, and text alignment can be specified. If not specified, this function uses the current settings. The available font types are:

- Courier-Bold

- Courier-BoldOblique

- Courier-Oblique

- Courier

- Helvetica-Bold

- Helvetica-BoldOblique

- Helvetica-Obliqu

- Helvetic

- Symbol

5

- Times-Bold

- Times-BoldItalic

- Times-Italic

- Times-Roman

The "rgb" and "cmyk" parameters can be used to specify the font color. Only one of these two parameters can be used at the same time. The "rgb" parameter must be an array of three integers between 0 and 255. The "cmyk" parameter must be an array of four decimal values between 0.0 and 1.0.

The alignment parameter can be used to set the text alignment, possible values are:

- left

- right

- center

- full

The last options will spread/justify the text from the left to the right margin if it covers at least 80% of the margin-width.

Examples

Example 1

```
{pdf( 'text', 'eZ publish and PDF'|wash( 'pdf' ),
      hash( 'font', 'Times-Bold', 'size', 22, 'align', 'center' ) )}
```

This example generates a centered text using "Times-Bold" at 22 points.

Example 2

```
{pdf( 'text', 'red'|wash( 'pdf' ), hash( 'rgb', array( 255, 0, 0 ) ) )}
{pdf( 'text', 'green'|wash( 'pdf' ), hash( 'rgb', array( 0, 255, 0 ) ) )}
{pdf( 'text', 'blue'|wash( 'pdf' ), hash( 'rgb', array( 0, 0, 255 ) ) )}
```

This example demonstrates text coloring.

Example 3

```
{pdf( 'text', 'A primary color is a color that can not be created by mixing
other colors in the gamut of a given color space. Primary colors may
themselves be mixed to produce most of the colors in a given color
space.'|wash( 'pdf' ), hash( 'align', 'full' ) )}
```

This example demonstrates how the text can be justified.

5

5.13.23   text_box

Summary

Inserts text at a specified location (DEPRECATED).

Usage

```
pdf( text_box, hash( text, text
                     x,      x_offset,
                     y,      y_offset,
                     width, total width,
                   [ align, text_alignment, ]
                   [ size,  text_size       ] ) )
```

Parameters

Name	Type	Description	Required
text	string	The text that should be inserted into the box.	Yes.
x	float	X coordinate of the text box.	Yes.
y	float	Y coordinate of the text box.	Yes.
width	float	The width of the text box.	Yes.
align	string	The alignment of the text.	No.
size	float	The font size.	No.

Description

This function creates a text box at the specified place on the current page. The font properties (type, size, and color) are not affected by the "set_font" (page 1499) function.

If the text box is positioned before any other content, the content will flow nicely over and under the text box. If other content is present before the text box is added, the text box will overlap it.

Examples

Example 1

```
{pdf( 'text_box',  hash( 'text', 'This text appears in a text box.'|wash(
'pdf' ),
                        'x', 250,
                        'y', 400,
                        'width', 30) )}
```

This example will write a sentence at coordinate (250, 400).

5

5.13.24   text_frame

Summary

Inserts a text with a frame (DEPRECATED).

Usage

```
pdf( text_frame, text, hash( [ frameRGB,      rgb_frame_color,    ]
                              [ frameCMYK,     cmyk_frame_color,   ]
                              [ textRGB,       rgb_text_color,     ]
                              [ textCMYK,      cmyk_text_color,    ]
                              [ fontSize,      font_size,          ]
                              [ fontName,      font_name,          ]
                              [ padding,       text_padding,       ]
                              [ leftPadding,   left_text_padding,  ]
                              [ rightPadding,  right_text_padding, ]
                              [ topPadding,    top_text_padding,   ]
                              [ bottomPadding, bottom_text_padding, ]
                              [ roundEnds,     round_ends          ] ) )
```

Parameters

Name	Type	Description	Required
frameRGB	array	Frame RGB color.	No.
frameCMYK	array	Frame CMYK color.	No.
textRGB	array	Text RGB color.	No.
textCMYK	array	Text CMYK color.	No.
fontSize	integer	Font size.	No.
fontName	string	Font name.	No.
padding	integer	Padding around the text.	No.
leftPadding	integer	Padding left of the text.	No.
rightPadding	integer	Padding right of the text.	No.
topPadding	integer	Padding above the text.	No.
bottomPadding	integer	Padding below the text.	No.
roundEnds	boolean	Squared or rounded frame edges.	No.
text	string	The text that should be displayed inside the frame.	Yes.

Description

This function creates a text with a colored frame around it. The frame has the size of the text plus the amount of padding. The padding can be specified for each side of the text with the parameters "leftPadding", "rightPadding", "topPadding", and "bottomPadding". If the padding is equal for each side, the "padding" parameter can be used instead.

The "leftPadding" and "rightPadding" parameters may be set to "-1", which will result in padding that reaches the page margin.

By default the frame has squared corners. If the "roundEnds" parameter is set to TRUE, the frame will have rounded corners.

The colors of the frame and the text can be specified using the "frameRGB", "frameCMYK", "textRGB", and "textCMYK" parameters. The RGB parameters must be arrays consisting of three integers between 0 and 255. The CMYK parameters must be arrays consisting of four decimal values between 0.0 and 1.0.

The font can be changed using the "fontSize" and "fontName" parameters.

Examples

Example 1

```
{pdf(text_frame, "Test frame", hash( roundEnds, true(),
    textCMYK, array(0.89, 0.43, 0.01, 0),
    frameRGB, array(255,255,128),
    padding, 8,
    fontSize, 14))}
```

This example creates a frame (with round edges) around the sentence "Test frame".

5.13.25   toc

Summary

Inserts a generated table of contents (DEPRECATED).

Usage

```
pdf(toc, hash( contentText, toc_header_text,
               [ size,       size_array,  ]
               [ dots,       boolean,     ]
               [ indent,     indent_array ] ) )
```

Parameters

Name	Type	Description	Required
contentText	string	Table of contents header.	Yes.
size	array	An array of numbers, indicating the font sizes for each header level.	No.
indent	array	An array of booleans, indicating whether the title should be indented for each header level.	No.
dots	boolean	Display dots between the titles and page numbers.	No.

Description

This function generates and inserts the table of contents after the front page. If the front page is not available, the table of contents is inserted at the beginning of the PDF document.

Examples

Example 1

```
{pdf( 'toc', hash( 'size', array( 18, 16, 14, 12, 10 ),
      'dots', true(),
      'contentText', 'Content'|wash( 'pdf' ),
      'indent', array( 0, 4, 6, 8, 10 ) ) )}
```

This example creates a table of contents. The size of the level 1 headers will be 18 and will not indented, level 2 headers will have size 16 and will be indented 4 dots, etc. There will be dots between the headers and the page numbers.

5.13.26   ul

Summary

Inserts a bullet list (DEPRECATED).

Usage

```
pdf( ul, text, hash( [ rgb,        rgb_color,    ]
                      [ cmyk,       cmyk_color,   ]
                      [ radius,     dot_radius,   ]
                      [ indent,     text_indent,  ]
                      [ pre_indent, bullet_indent ] ) ) )
```

Parameters

Name	Type	Description	Required
text	string	Bullet text.	Yes.
rgb	array	Array of RGB colors.	No.
cmyk	array	Array of CMYK colors.	No.
radius	float	The radius of the dot.	No.
indent	float	Text indentation after the dot.	No.
pre_indent	float	Indentation before the bullet.	No.

Description

This function inserts a bullet list into the PDF document.

The colors of the bullet can be specified using the "rgb" or the "cmyk" parameter. The "rgb" parameter must be an array consisting of three integers between 0 and 255. The "cmyk" parameter must be array of four decimal numbers between 0.0 and 1.0.

The size of the bullet can be controlled using the "radius" parameter. The indentation (in dots) before and after the dot can be specified using the "pre_indent" and "indent" parameters.

Examples

Example 1

```
{pdf( 'text', "Most popular internet browsers in 2004:" |wash( 'pdf' )}

{pdf( 'ul', 'Internet Explorer (88,9%)'|wash( 'pdf' ) )}
{pdf( 'ul', 'Version 6 (80.95%)'|wash( 'pdf' ), hash( 'pre_indent', 15 ) )}
{pdf( 'ul', 'Version 5.5 (4.18%)'|wash( 'pdf' ), hash( 'pre_indent', 15 ) )}
{pdf( 'ul', 'Version 5.0 (3.66%)'|wash( 'pdf' ), hash( 'pre_indent', 15 ) )}
```

5

```
{pdf( 'ul', 'Mozilla-based browsers (7.35%)'|wash( 'pdf' ) )}
{pdf( 'ul', 'Rest (3.75%)'|wash( 'pdf' ) )}
```

This example generates a bullet list where some of the bullets are indented.

5

5.14   Configuration files

\begin{description}   \item[audit.ini]\refer{label5b50f2ead8ac760a5920a1b585c76426}\   \\
Settings related to audit trailing of various user operations.

\item[binaryfile.ini]\refer{label5304155f972efd296d8c8010075ee87d}\ \\ This configuration
file allows you to set handlers for new datatypes.

\item[browse.ini]\refer{label5010e81f7ff997b7728c0a4210d2fee8}\ \\ This configuration file
contains settings related to the browse page.

\item[collaboration.ini]\refer{label2ee32fce07c7c2de84e24d944042ad92}\   \\   Not docu-
mented yet.

\item[collect.ini]\refer{label7fcc517f5e64603a9e7af64d420b71b5}\ \\ This configuration file
contains settings regarding information collection.

\item[config.php]\refer{labele967a503b3bb6de8291b74431331a067}\   \\   Configurations
points available from file config.php

\item[content.ini]\refer{label46f09d546c7ed2720f0477334f5f3049}\ \\ Settings related to
content.

\item[contentstructuremenu.ini]\refer{label1522c44a00f2e8ecf04c5b51912fd962}\   \\   Set-
tings related to the "Content structure" tree menu in the admin interface.

\item[country.ini]\refer{labela69d6645bfdadd82c0861de34c6f745e}\ \\ List of countries for
the "ezcountry" datatype.

\item[cronjob.ini]\refer{labelb8eaeebead3c0754f0a4ccbe5b1e4828}\   \\   Settings related to
cronjobs.

\item[datatype.ini]\refer{label767acbfe777e661536290c63e93d9a02}\ \\ This configuration
file contains settings regarding datatypes.

\item[datetime.ini]\refer{labelf803eba41af3ac72c30cd9dac390cc72}\ \\ Not documented yet.

\item[dbschema.ini]\refer{label35969679bed795ba4df4ae9b8b6bae9f}\ \\ settings related to
schemas

\item[debug.ini]\refer{labeldb7620427ff99d9a229c2cdac58402a0}\ \\ These files help you de-
fine settings related to debugging.

\item[design.ini]\refer{label2c03b32c8bdd3355ee617876703630ae}\ \\ Settings related to de-
signs and design related files like css and javascripts.

\item[error.ini]\refer{label7de1eddf8dd54481f7213de077ee647b}\ \\ Not documented yet.

\item[extendedattributefilter.ini]\refer{label58a354c5932a40c8612288a40077616d}\   \\   Not
documented yet.

\item[ezxml.ini]\refer{label9c9cf2668bf11a8473754f1e5df99419}\ \\ Settings related to XML
handling (the XML Block datatype).

\item[fetchalias.ini]\refer{label020c11733843f25dab676bff2872d123}\   \\   Not   documented
yet.

5

\item[file.ini]\refer{labelc6a655cd40467a222fdc61938e82fa46}\ \\ Settings related to file handling

\item[i18n.ini]\refer{labelba40b8d55297f2c94c8b11c3464b304d}\ \\ Settings related to internationalization.

\item[icon.ini]\refer{label6298555e61e24a8fde80a815962c9e3c}\ \\ Contains settings regarding icons and flag icons

\item[image.ini]\refer{label64349d56883c761f9e0fb8c35ed919ab}\ \\ contains settings related to images

\item[layout.ini]\refer{label1e2e14025117d5f7da040578d668298d}\ \\ Not documented yet.

\item[ldap.ini]\refer{label32dcba0c82188d826904f9f8c382ea17}\ \\ Settings regarding connection and authentication of users via LDAP

\item[logfile.ini]\refer{label8a40c0152316ae8d78af319aa83392ef}\ \\ Settings related to log files.

\item[menu.ini]\refer{labele309d5c687e61c8f5e3988a8db4740ad}\ \\ Settings regarding the Menu such as the location of menus and a list of the classes that are allowed to show in the top menu's

\item[module.ini]\refer{labelb454b9751e66cabfd6bbf9aadcfead5d}\ \\ contains settings related to modules

\item[notification.ini]\refer{label04fffa9fdfa5ba79f1d84d9eb05a57b8}\ \\ contains configuration settings regarding notification e-mails

\item[override.ini]\refer{label2dfa8dbc03e484fb67646f1db8f9619f}\ \\ Not documented yet, but valuable links provided here

\item[package.ini]\refer{label3663eabebdd2aacbbae8c6c53e2b6427}\ \\ Settings for the packages on the eZ Publish Setup Wizard

\item[paymentgateways.ini]\refer{label244095b0b4227fcbb212d591d95ee951}\ \\ Setting related to the payment gateways

\item[setup.ini]\refer{label9ec8858c3e79c380b75dcb15fccde1fd}\ \\ Setup.ini contains the configuration settings for all test that are run by the setup, as well as settings regarding the setup itself.

\item[shopaccount.ini]\refer{label9044390a226969b2fd8c5dbe2ffea59c}\ \\ Settings related to shop account handlers

\item[site.ini]\refer{label1546d4101d8f6bc5b5e016c710d0826c}\ \\ Controls the overall/ main behavior of the system.

\item[soap.ini]\refer{labelefab12b69cab277821c0ab690f0227ef}\ \\ Settings related to SOAP functions and its extensions

\item[staticcache.ini]\refer{label0b7f556f40c08c738710029c934b4936}\ \\ Settings related to the static cache.

\item[template.ini]\refer{labelb05832f3ddd45ebc4b7464b5cae751ed}\ \\ This file controls the

5

settings for the template engine

\item[textfile.ini]\refer{label11e0a0c25541407da30c6a09bdf74ab0}\ \\ eZ Publish configuration file for the textfile based login handler

\item[texttoimage.ini]\refer{labele64a4f770a69f05359f5a693ecd26df3}\ \\ This setting relates to converting text to images

\item[toolbar.ini]\refer{label709808bf563ca41d1ecf48f205354cf4}\ \\ Not documented yet.

\item[transform.ini]\refer{labeld54a68da4dc7d875d1fb9e590f81c939}\ \\ This file contains settings regarding character transformation.

\item[units.ini]\refer{label1c5b78243f3df5e91f03aba7a2c10473}\ \\ This file defines the units to use.

\item[upload.ini]\refer{labelae50a2588a73aec335689896ee89d7ca}\ \\ This setting sets up the 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.

\item[viewcache.ini]\refer{label498be0414457d003fca2b05bf68a5718}\ \\ Settings related to the view cache system.

\item[webdav.ini]\refer{labeldea078d09d6acdc9b1ae5c856ed4bf08}\ \\ Settings related to WebDAV

\item[wordtoimage.ini]\refer{label31b936053d9d24281938245eb0d8d2fc}\ \\ This setting relates to converting words to images and images to words

\item[workflow.ini]\refer{label063e55802bf993d670d97e9b1cde0418}\ \\ Settings related to workflows.

\end{description}

5.14.1   audit.ini

The configuration blocks are documented in the following sections:

\begin{itemize}   \item{}   [AuditSettings]\refer{label0a994d824653bb9cd13884e8146f80f7}
\end{itemize}

5

[AuditSettings]

\begin{description} \item[Audit]\refer{label807276dba477e8a0f95a2a47b4915b28}\ \\ Enables or disables the generation of audit logs.

\item[AuditFileNames]\refer{label723185570176e60a603e4a07189eeaf3}\ \\ Sets which types of activities that should be logged and to which files.

\item[LogDir]\refer{labela3c39e39c2844efb883e25980f7cc3d0}\ \\ Sets the directory where eZ Publish stores the audit log files.

\end{description}

5

AuditFileNames

Summary

Sets which types of activities that should be logged and to which files.

Usage

```
AuditFileNames[audit_function1]=logfile1
AuditFileNames[audit_function2]=logfile2
...
```

Description

This directive makes it possible to control which types of activities that should be logged and to which files they should be logged. By default, this configuration array lists all the built-in audit functions along with a default set of corresponding log file names (see below). The target directory of the audit logs is defined by the "LogDir (page 1522)" directive. Note that the master "Audit (page 1521)" configuration directive must be enabled for auditing to work.

The default configuration enables all the built-in audit function and suggests a default set of files:

```
AuditFileNames[]
AuditFileNames[user-login]=login.log
AuditFileNames[user-failed-login]=failed_login.log
AuditFileNames[content-delete]=content_delete.log
AuditFileNames[content-move]=content_move.log
AuditFileNames[role-change]=role_change.log
AuditFileNames[role-assign]=role_assign.log
AuditFileNames[section-assign]=section_assign.log
AuditFileNames[order-delete]=order_delete.log
```

Refer to the €Audit trailing (page 232)€ section of the "Features" chapter for more information.

Examples

Example 1

```
Audit=enabled
LogDir=var/log/audit
AuditFileNames[]
AuditFileNames[user-login]=login.log
```

With this configuration, the system will only record information about successful login attempts (all other audit functions will be skipped). The information will be put to the "login.log" file

located in the "var/log/audit" directory. Each record in the file will contain information about the user's name and ID number, IP address, plus the date and time when the user logged into the system.

5

Audit

Summary

Enables or disables the generation of audit logs.

Usage

```
Audit= enabled|disabled
```

Description

This directive controls whether eZ Publish should generate audit logs based on user activity. By default, this setting is disabled, which means that the system will not generate audit logs. If enabled, the system will store information about various user operations in audit log files. You can use the "AuditFileNames (page 1519)" setting to specify which operations that should be logged using which log files. The "LogDir (page 1522)" setting can be used to specify where the audit log files should be stored.

Refer to the €Audit trailing (page 232)€ section of the "Features" chapter for more information.

5

LogDir

Summary

Sets the directory where eZ Publish stores the audit log files.

Usage

```
LogDir= directory_name
```

Description

This directive makes it possible to control where the audit logs should be stored. As of eZ Publish 4.7, the directory is relative to the VarDir (page 2384) defined in site.ini. The default value is "log/audit". The "AuditFileNames (page 1519)" setting can be used to control which activities that should be logged and to which files they should be logged. Note that the master "Audit (page 1521)" configuration directive must be enabled for auditing to work.

Refer to the €Audit trailing (page 232)€ section of the "Features" chapter for more information.

5

5.14.2   binaryfile.ini

The configuration blocks are documented in the following sections:

\begin{itemize} \item{} [HandlerSettings]\refer{labelfb2fb96ca4027081672708d1a3caeca0} \item{}   [PDFHandlerSettings]\refer{label3b83dd1c5548c735e94ff53c1aeccef7}   \item{} [WordHandlerSettings]\refer{label66bdaea6abefcc61c4754f703a66dbf5} \end{itemize}

5

[WordHandlerSettings]

This block defines handler settings regarding Word files.

TextExtractionTool   (page 1525)
    Sets the text extraction tool used to fetch information from Word files.

5

TextExtractionTool

Summary

Sets the text extraction tool used to fetch information from Word files.

Description

This setting sets the text extraction tool used to fetch information from Word files. This text extraction tool is an external program which returns the content of the MS word file as plain text.
This feature requires the Word View (wv) program to work properly.
WV or Word View is a library and set of command-line programs that understands Microsoft Word. It is able to convert MS Word documents into HTML (wvHtml), Abiword (wvAbw), plain text (wvText), RTF, PostScript, PDF, and other formats.

By default wvText is used by eZ Publish, the default settings for using this program are:

```
[HandlerSettings]

MetaDataExtractor[application/msword]=ezword

[WordHandlerSettings]

TextExtractionTool=wvWare -x /usr/local/wv/wvText.xml
```

This setting is case sensitive.

Examples

```
TextExtractionTool=wvWare -x /usr/local/wv/wvText.xml
```

This is the default setting.

5

[PDFHandlerSettings]

TextExtractionTool   (page)
   Sets the text extraction tool to use to fetch information from PDF files.

5

TextExtractionTool

Summary

Sets the text extraction tool to use to fetch information from PDF files.

Usage

```
TextExtractionTool=<name text extraction tool>
```

Description

This setting sets the text extraction tool to use to fetch information from PDF files. This text extraction tool is an external program which returns the content of the PDF file as plain text. For eZ Publish pstotext and pdftotext programs have been tested. Other programs should also work. By default pstotext is used, the default settings for using this program are:

```
[HandlerSettings]

MetaDataExtractor[application/pdf]=ezpdf

[PDFHandlerSettings]

TextExtractionTool=pstotext
```

As an alternative, The pdftotext program can be used which is a PDF reader that also provides numerous PDF and PS utilities.
This setting is case sensitive.

Examples

```
TextExtractionTool=pstotext
```

This is the default setting.

5

[HandlerSettings]

This block contains settings related to binaryfile handler settings.

ExtensionRepositories   (page 1529)
    Lists extensions with metadata extractor handlers.

MetaDataExtractor   (page 1531)
    Sets the metadata extractor handler to use.

Repositories   (page 1530)
    Sets directories with metadata extractor handlers.

5

ExtensionRepositories

Summary

Lists extensions with metadata extractor handlers.

Usage

```
ExtensionRepositories[]=<extension name>
```

Description

Sets a list of extensions which have metadata extractor handlers. It's recommended to create a settings/binaryfile.ini.append.php file in your extension and add the extension name, this will allow your installation to automatically get handlers from the extension when it is turned on.
It is necessary to place handlers in the extension subdir "plugins". For example:

```
.../extension/ezfind/search/plugins
```

This setting is case sensitive.

Examples

```
ExtensionRepositories[]=ezfind
```

This is the default setting.

5

Repositories

Summary

Sets directories with metadata extractor handlers.

Usage

```
Repositories[]=<path to directory>
```

Description

Sets a list of directories where your eZ Publish installation must search for metadata extractor handlers.
The path to the directories is used as value.
This setting is case sensitive.

Examples

```
Repositories[]=kernel/classes/datatypes/ezbinaryfile
```

This is the default setting.

5

MetaDataExtractor

Summary

Sets the metadata extractor handler to use.

Usage

```
MetaDataExtractor[format]=<name metadata extractor handler>
```

Description

This setting defines the metadata extractor handler to use, to extract metadata from a range of formats, for example PDF and MS Word.
In eZ Publish versions prior to 3.8.0, the following default settings for binary file indexing were specified in the "binaryfile.ini" configuration file:

```
MetaDataExtractor[text/plain]=plaintext
 MetaDataExtractor[application/pdf]=pdf
 MetaDataExtractor[application/msword]=word
```

Since eZ Publish 3.8.0, the default settings for binary file indexing have changed, and now look like this:

```
MetaDataExtractor[text/plain]=ezplaintext
 MetaDataExtractor[application/pdf]=ezpdf
 MetaDataExtractor[application/msword]=ezword
```

This setting is case sensitive.

Examples

```
MetaDataExtractor[text/plain]=ezplaintext
MetaDataExtractor[application/pdf]=ezpdf
MetaDataExtractor[application/msword]=ezword
```

This is the default setting.

5

5.14.3   browse.ini

The configuration blocks are documented in the following sections:

- [AddBookmark] (page 1639)

- [AddCustomer] (page 1684)

- [AddNodeAssignment] (page 1594)

- [AddPrimaryNodeAssignment] (page 1599)

- [AddRelatedImageToDataType] (page 1635)

- [AddRelatedObjectsListToDataType] (page 1631)

- [AddRelatedObjectsToDataType] (page 1623)

- [AddRelatedObjectsToDataType] (page 1627)

- [AddRelatedObjects] (page 1619)

- [AddSubtreeSubscribingNode] (page 1643)

- [AssignRole] (page 1570)

- [AssignSection] (page 1544)

- [BrowseSettings] (page 1534)

- [CopyNode] (page 1663)

- [CopySubtree] (page 1667)

- [ExampleAction] (page 1679)

- [ExportSourceBrowse] (page 1548)

- [FindLimitationNode] (page 1580)

- [FindLimitationSubtree] (page 1576)

- [FindProduct] (page 1584)

- [MoveNodeAssignment] (page 1604)

- [MoveNode] (page 1609)

- [NewObjectAddNodeAssignment] (page 1588)

- [RSSExportImageBrowse] (page 1553)

- [RSSObjectBrowse] (page 1559)

- [RSSUserBrowse] (page 1564)

5

- [SelectMultipleUsers] (page 1675)

- [SelectObjectRelationListNode] (page 1647)

- [SelectObjectRelationNode] (page 1651)

- [SelectSingleUser] (page 1671)

- [SelectToolbarNodePath] (page 1659)

- [SelectToolbarNode] (page 1655)

- [SwapNode] (page 1614)

- [users] (page 1541)

5

[BrowseSettings]

This block contains settings related to content browsing.

AliasList   (page 1538)
>   This setting maps node aliases to node IDs.

DefaultActionName   (page 1537)
>   Sets the default name of the action.

DefaultReturnType   (page 1536)
>   This setting defines the type of the return values.

DefaultSelectionType   (page 1535)
>   Sets the default browse selection type.

DefaultTopLevelNodes   (page 1540)
>   Lists the default top level nodes.

MaximumRecentItems   (page 1539)
>   Sets the default maximum value of recent items which can be shown from the browse page.

5

DefaultSelectionType

Summary

Sets the default browse selection type.

Usage

```
DefaultSelectionType= single | multiple
```

Description

Sets the default browse selection type.

Possible values are:

- single: Allows for just one item to be selected

- multiple: Allows for multiple items to be selected

This setting is case sensitive.

Examples

```
DefaultSelectionType=single
```

The default value is "single".

5

DefaultReturnType

Summary

This setting defines the type of the return values.

Usage

```
DefaultReturnType= NodeID | ObjectID
```

Description

This setting defines the type of the return values.
Possible values are:

- NodeID: Returns the node ids of the selected items
- ObjectID: Returns the object ids of the selected items

This setting is case sensitive.

Examples

```
DefaultReturnType=NodeID
```

The default value is NodeID.

5

DefaultActionName

Summary

Sets the default name of the action.

Usage

```
DefaultActionName=<action name>
```

Description

Sets the default name of the action.

This setting is case sensitive.

Examples

```
DefaultActionName=content
```

This is the default value.

5

AliasList

Summary

This setting maps node aliases to node IDs.

Usage

```
AliasList[<node alias>]=<node ID>
```

Description

This setting maps node aliases to node IDs.

For start nodes and top level nodes it is possible to use text aliases in browse.ini. The following aliases are possible by default:

- content: Represents the content root node defined "[NodeSettings] RootNode=" in content.ini, usually 2

- users: Represents the users root node defined "[NodeSettings] UserRootNode=" in content.ini, usually 5

Other, custom, aliases can also be used, but these must be defined in the here.

This setting is case sensitive.

Examples

```
AliasList[images]=2
```

This is the default setting.

5

MaximumRecentItems

Summary

Sets the default maximum value of recent items which can be shown from the browse page.

Usage

```
MaximumRecentItems=<integer>
```

Description

Sets the default maximum value of recent items which can be shown from the browse page.

Examples

```
MaximumRecentItems=10
```

"10" is the default value.

5

DefaultTopLevelNodes

Summary

Lists the default top level nodes.

Usage

```
DefaultTopLevelNodes[]=<node>
```

Description

Lists the top level nodes that are to be shown by default in browse page.

Nodes are used as value.

This setting is case sensitive.

Examples

```
DefaultTopLevelNodes[]=content
DefaultTopLevelNodes[]=users
DefaultTopLevelNodes[]=media
DefaultTopLevelNodes[]=setup
```

This will show the Content, Users, Media and Setup nodes. This is the default value.

5

[users]

This block contains browse settings for users.

StartNode (page 1543)
    This setting sets the node from where the browsing should start for the action "Users".

TopLevelNodes (page 1542)
    Lists the top level nodes to show in browse page for the action "Users".

5

TopLevelNodes

Summary

Lists the top level nodes to show in browse page for the action "Users".

Usage

```
TopLevelNodes[]=<node>
```

Description

Lists the top level nodes to show in browse page for the action "Users".

Examples

```
TopLevelNodes[]=users
```

"users" is the default value for this setting.

5

StartNode

Summary

This setting sets the node from where the browsing should start for the action "Users".

Usage

StartNode=<node>

Description

This setting sets the node from where the browsing should start for the action "Users".

Nodes can be used as value.

This setting is case sensitive.

Examples

StartNode=users

"Users" is the default value here.

5

[AssignSection]

This block is used when assigning a section in the node tree.

ReturnType   (page 1545)
> This setting defines the type of return values.

SelectionType   (page 1546)
> Sets the type of selection when assigning a section.

StartNode   (page 1547)
> This setting sets the node from where the browsing should start when assigning a section to the node tree.

5

ReturnType

Summary

This setting defines the type of return values.

Usage

```
ReturnType=NodeID | ObjectID
```

Description

This setting defines the type of return values.

Possible values are:

- NodeID: Returns the node ids of the selected items
- ObjectID: Returns the object ids of the selected items

This setting is case sensitive.

Examples

```
ReturnType=NodeID
```

"NodeID" is the default value for this setting.

5

SelectionType

Summary

Sets the type of selection when assigning a section.

Usage

```
SelectionType=single | multiple
```

Description

Sets the type of selection when assigning a section.
Possible values are:

- single: Allows for just one item to be selected
- multiple: Allows for multiple items to be selected

This setting is case sensitive.

Examples

```
SelectionType=multiple
```

"multiple" is the default value for this setting.

5

StartNode

Summary

This setting sets the node from where the browsing should start when assigning a section to the node tree.

Usage

```
StartNode=<node>
```

Description

This setting set the node from where the browsing should start when assigning a section to the node tree.

Nodes can be used as value, so possible values are content, users, media or setup.

This setting is case sensitive.

Examples

```
StartNode=content
```

"content" is the default value for this setting.

5

[ExportSourceBrowse]

This block is used to select a source when exporting part on content tree.

ReturnType   (page 1549)
> This setting defines the type of the return values.

SelectionType   (page 1550)
> Sets the type of selection for this action.

StartNode   (page 1551)
> This setting sets the node from where the browsing should start when the action "Export-SourceBrowse" is used.

TopLevelNodes   (page 1552)
> Lists the top level nodes to show on browse page for this action.

5

ReturnType

Summary

This setting defines the type of the return values.

Usage

```
ReturnType=NodeID | ObjectID
```

Description

This setting defines the type of the return values.

Possible values are:

- NodeID: Returns the node ids of the selected items
- ObjectID: Returns the object ids of the selected items

This setting is case sensitive.

Examples

```
ReturnType=NodeID
```

"NodeID" is the default value for this setting.

5

SelectionType

Summary

Sets the type of selection for this action.

Usage

```
SelectionType=single | multiple
```

Description

Sets the type of selection for this action.

Possible values are:

- single: Allows for just one item to be selected
- multiple: Allows for multiple items to be selected

This setting is case sensitive.

Examples

```
SelectionType=single
```

"Single" is the default value for this setting.

5

StartNode

Summary

This setting sets the node from where the browsing should start when the action "ExportSource-Browse" is used.

Usage

```
StartNode=<node>
```

Description

This setting sets the node from where the browsing should start when the action "ExportSource-Browse" is used.

Node can be used as value, so possible values are "content", "users", "media" and "setup".

This setting is case sensitive.

Examples

```
StartNode<b>=</b>content
```

"content" is the default value for this setting.

5

TopLevelNodes

Summary

Lists the top level nodes to show on browse page for this action.

Usage

```
TopLevelNodes[]=<node>
```

Description

Lists the top level nodes to show in browse page for this action. Possible values are content, users, media and setup.

This setting is case sensitive

Examples

```
TopLevelNodes[]=content
```

"content" is the default value for this setting.

5

[RSSExportImageBrowse]

This action is used to select RSS export image when browsing.

Class   (page 1555)
> This setting allows you to specifying a class identifier.

ReturnType   (page 1556)
> This setting defines the type of the return values.

SelectionType   (page 1557)
> Sets the type of selection for this action.

StartNode   (page 1558)
> This setting sets the node from where the browsing should start when the action "RSSObjectBrowse" is used.

TopLevelNodes   (page 1554)
> Lists the top level nodes to show in browse page.

5

TopLevelNodes

Summary

Lists the top level nodes to show in browse page.

Usage

```
TopLevelNodes[]=<node>
```

Description

Lists the top level nodes to show in browse page for this action. Possible values are content, users, media and setup.

This setting is case sensitive.

Examples

```
TopLevelNodes[]=content
TopLevelNodes[]=media
```

5

Class

Summary

This setting allows you to specifying a class identifier.

Usage

```
Class[]=<class>
```

Description

This setting allows you to specifying a class identifier, therefore allowing only objects of a specified class to be selected.

This setting is case sensitive.

Examples

```
Class[]=image
```

"Image" is the default value for this setting.

5

ReturnType

Summary

This setting defines the type of the return values.

Usage

```
ReturnType=NodeID | ObjectID
```

Description

This setting defines the type of the return values.
Possible values are:

- NodeID: Returns the node ids of the selected items
- ObjectID: Returns the object ids of the selected items

This setting is case sensitive.

Examples

```
ReturnType=NodeID
```

"NodeID" is the default value for this setting.

5

SelectionType

Summary

Sets the type of selection for this action.

Usage

```
SelectionType=single | multiple
```

Description

Sets the type of selection for this action.

Possible values are:

- single: Allows for just one item to be selected
- multiple: Allows for multiple items to be selected

This setting is case sensitive.

Examples

```
SelectionType=single
```

"single" is the default value for this setting.

5

StartNode

Summary

This setting sets the node from where the browsing should start when the action "RSSObject-Browse" is used.

Usage

```
StartNode=<node>
```

Description

This setting sets the node from where the browsing should start when the action "RSSObject-Browse" is used.

Nodes can be used as value, possibilities being content, users, media and setup.

This setting is case sensitive.

Examples

```
StartNode=content
```

"content" is the default value for this setting.

5

[RSSObjectBrowse]

This action is used to select destination and source in the RSS module.

ReturnType   (page 1561)
> This setting defines the type of the return values.

SelectionType   (page 1562)
> Sets the type of selection for this action.

StartNode   (page 1563)
> This setting sets the node from where the browsing should start when the action "RSSOb-jectBrowse" is used.

TopLevelNodes   (page 1560)
> Lists the top level nodes to show in browse page for this action.

5

TopLevelNodes

Summary

Lists the top level nodes to show in browse page for this action.

Usage

```
TopLevelNodes[]=<node>
```

Description

Lists the top level nodes to show in browse page for this action. Possible values are Content, Users, Media and Setup.

This setting is case sensitive.

Examples

```
TopLevelNodes[]=content
TopLevelNodes[]=media
```

5

ReturnType

Summary

This setting defines the type of the return values.

Usage

```
ReturnType=NodeID | ObjectID
```

Description

This setting defines the type of the return values.

Possible values are:

- NodeID: Returns the node ids of the selected items
- ObjectID: Returns the object ids of the selected items

This setting is case sensitive.

Examples

```
ReturnType=NodeID
```

"NodeID" is the default value for this setting.

5

SelectionType

Summary

Sets the type of selection for this action.

Usage

```
SelectionType=single | multiple
```

Description

Sets the type of selection for this action.

Possible values are:

- single: Allows for just one item to be selected
- multiple: Allows for multiple items to be selected

This setting is case sensitive.

Examples

```
SelectionType=single
```

"single" is the default value for this setting.

5

StartNode

Summary

This setting sets the node from where the browsing should start when the action "RSSObject-Browse" is used.

Usage

```
StartNode=<node>
```

Description

This setting sets the node from where the browsing should start when the action "RSSObject-Browse" is used.

Nodes can be used as value, possibilities being content, users, media and setup.

This setting is case sensitive.

Examples

```
StartNode=content
```

"content" is the default value for this setting.

5

[RSSUserBrowse]

This action is used to select owner of imported objects in the RSS module.

Class   (page 1567)
    This setting allows you to specifying a class identifier.

ReturnType   (page 1569)
    This setting defines the type of the return values.

SelectionType   (page 1568)
    Sets the type of selection for this action.

StartNode   (page 1565)
    This setting sets the node from where the browsing should start when the action "RSSUser-Browse" is used.

TopLevelNodes   (page 1566)
    Lists the top level nodes to show in browse page.

5

StartNode

Summary

This setting sets the node from where the browsing should start when the action "RSSUser-Browse" is used.

Usage

```
StartNode=<node>
```

Description

This setting sets the node from where the browsing should start when the action "RSSUser-Browse" is used.
Nodes such as content, users, media and setup are used as value.

This setting is case sensitive.

Examples

```
StartNode=users
```

"users" is the default value for this setting.

5

TopLevelNodes

Summary

Lists the top level nodes to show in browse page.

Usage

```
TopLevelNodes[]=<node>
```

Description

Lists the top level nodes to show in browse page for this action. Possible values are content, users, media and setup.

This setting is case sensitive.

Examples

```
TopLevelNodes[]=users
```

5

Class

Summary

This setting allows you to specifying a class identifier.

Usage

```
Class[]=<class>
```

Description

This setting allows you to specifying a class identifier, therefore allowing only objects of a specified class to be selected.

This setting is case sensitive.

Examples

```
Class[]=user
Class[]=user_group
```

This is the default setting.

5

SelectionType

Summary

Sets the type of selection for this action.

Usage

```
SelectionType=single | multiple
```

Description

Sets the type of selection for this action.

Possible values are:

- single: Allows for just one item to be selected

- multiple: Allows for multiple items to be selected

This setting is case sensitive.

Examples

```
SelectionType=multiple
```

"multiple" is the default value for this setting.

5

ReturnType

Summary

This setting defines the type of the return values.

Usage

```
ReturnType=NodeID | ObjectID
```

Description

This setting defines the type of the return values.

Possible values are:

- NodeID: Returns the node ids of the selected items
- ObjectID: Returns the object ids of the selected items

This setting is case sensitive.

Examples

```
ReturnType=ObjectID
```

"ObjectID" is the default value for this setting.

5

[AssignRole]

This action is used when assigning a role to a user or user group.

Class   (page 1572)
    This setting allows you to specifying a class identifier.

ReturnType   (page 1573)
    This setting defines the type of the return values.

SelectionType   (page 1575)
    sets how many values can be selected

StartNode   (page 1574)
    This setting sets the node from where the browsing should start when the action "Assign-Role" is used.

TopLevelNodes   (page 1571)
    Lists the top level nodes to show in browse page.

5

TopLevelNodes

Summary

Lists the top level nodes to show in browse page.

Usage

```
TopLevelNodes[]=<node>
```

Description

Lists the top level nodes to show in browse page for this action. Possible values are Content, Users, Media and Setup.

This setting is case sensitive.

Examples

```
TopLevelNodes[]=users
```

5

Class

Summary

This setting allows you to specifying a class identifier.

Usage

```
Class[]=<class>
```

Description

This setting allows you to specifying a class identifier, therefore allowing only objects of a specified class to be selected.

This setting is case sensitive.

Examples

```
Class[]=user
Class[]=user_group
```

This is the default setting.

5

ReturnType

Summary

This setting defines the type of the return values.

Usage

```
ReturnType=NodeID | ObjectID
```

Description

This setting defines the type of the return values.

Possible values are:

- NodeID: Returns the node ids of the selected items
- ObjectID: Returns the object ids of the selected items

This setting is case sensitive.

Examples

```
ReturnType=ObjectID
```

"ObjectID" is the default value for this setting.

5

StartNode

Summary

This setting sets the node from where the browsing should start when the action "AssignRole" is used.

Usage

```
StartNode=<node>
```

Description

This setting sets the node from where the browsing should start when the action "AssignRole" is used.

Nodes such as content, users, media and setup are used as value.

This setting is case sensitive

Examples

```
StartNode=users
```

"users" is the default value for this setting.

5

SelectionType

Summary

sets how many values can be selected

Usage

```
SelectionType=single | multiple
```

Description

Sets the type of selection for this action.

Possible values are:

- single: Allows for just one item to be selected
- multiple: Allows for multiple items to be selected

This setting is case sensitive.

Examples

```
SelectionType=multiple
```

"multiple" is the default value for this setting.

5

[FindLimitationSubtree]

This action is used when the start of a subtree should be selected in policy editing.

ReturnType   (page 1579)
　　This setting defines the type of the return values.

SelectionType   (page 1578)
　　Sets the type of selection for this action.

StartNode   (page 1577)
　　This setting sets the node from where the browsing should start when the action "FindLim-
　　itationNode" is used.

5

StartNode

Summary

This setting sets the node from where the browsing should start when the action "FindLimitationNode" is used.

Usage

```
StartNode=<node>
```

Description

This setting sets the node from where the browsing should start when the action "FindLimitationNode" is used.

Nodes are used as value, therefore possible values include content, users, media, setup.

This setting is case sensitive.

Examples

```
StartNode=content
```

"content" is the default value for this setting.

5

SelectionType

Summary

Sets the type of selection for this action.

Usage

```
SelectionType=single | multiple
```

Description

Sets the type of selection for this action.

Possible values are:

- single: Allows for just one item to be selected
- multiple: Allows for multiple items to be selected

This setting is case sensitive.

Examples

```
SelectionType=multiple
```

"multiple" is the default value for this setting.

5

ReturnType

Summary

This setting defines the type of the return values.

Usage

```
ReturnType=NodeID | ObjectID
```

Description

This setting defines the type of the return values.
Possible values are:

- NodeID: Returns the node ids of the selected items
- ObjectID: Returns the object ids of the selected items

This setting is case sensitive.

Examples

```
ReturnType=NodeID
```

"NodeID" is the default value for this setting

5

[FindLimitationNode]

This action is used when a node should be selected in policy editing.

ReturnType   (page 1583)
    This setting defines the type of the return values.

SelectionType   (page 1582)
    Sets the type of selection for this action.

StartNode   (page 1581)
    This setting sets the node from where the browsing should start when the action "FindLim-
    itationNode" is used.

5

StartNode

Summary

This setting sets the node from where the browsing should start when the action "FindLimitationNode" is used.

Usage

```
StartNode=<node>
```

Description

This setting sets the node from where the browsing should start when the action "FindLimitationNode" is used.

Nodes are used as value, therefore possible values include content, users, media, setup.

This setting is case sensitive.

Examples

```
StartNode=content
```

"content" is the default value for this setting.

5

SelectionType

Summary

Sets the type of selection for this action.

Usage

```
SelectionType=single | multiple
```

Description

Sets the type of selection for this action.

Possible values are:

- single: Allows for just one item to be selected

- multiple: Allows for multiple items to be selected

This setting is case sensitive.

Examples

```
SelectionType=multiple
```

This is the default setting.

5

ReturnType

Summary

This setting defines the type of the return values.

Usage

```
ReturnType=NodeID | ObjectID
```

Description

This setting defines the type of the return values.

Possible values are:

- NodeID: Returns the node ids of the selected items
- ObjectID: Returns the object ids of the selected items

This setting is case sensitive.

Examples

```
ReturnType=NodeID
```

"NodeID" is the default value for this setting.

5

[FindProduct]

This action is used when a node should be selected in discount rule edit.

ReturnType   (page 1587)
> This setting defines the type of the return values.

SelectionType   (page 1586)
> Sets the type of selection for this action.

StartNode   (page 1585)
> This setting sets the node from where the browsing should start when the action "Find-Product" is used.

5

StartNode

Summary

This setting sets the node from where the browsing should start when the action "FindProduct" is used.

Usage

```
StartNode=<node>
```

Description

This setting sets the node from where the browsing should start when the action "FindProduct" is used.

Nodes such content, users, media and setup are used as value.

This setting is case sensitive.

Examples

```
StartNode=content
```

"content" is the default value for this setting.

5

SelectionType

Summary

Sets the type of selection for this action.

Usage

```
SelectionType=single | multiple
```

Description

Sets the type of selection for this action.

Possible values are:

- single: Allows for just one item to be selected
- multiple: Allows for multiple items to be selected

This setting is case sensitive.

Examples

```
SelectionType=multiple
```

"multiple" is the default value for this setting.

5

ReturnType

Summary

This setting defines the type of the return values.

Usage

```
ReturnType=NodeID | ObjectID
```

Description

This setting defines the type of the return values.

Possible values are:

- NodeID: Returns the node ids of the selected items
- ObjectID: Returns the object ids of the selected items

This setting is case sensitive.

Examples

```
ReturnType=ObjectID
```

"ObjectID" is the default value for this setting.

5

[NewObjectAddNodeAssignment]

This action is used when newly created objects should select a node placement, because for the new object no placement is known yet (for instance from the left menu).

ActionName   (page 1590)
    Sets the name of the action that needs to be used.

ReturnType   (page 1591)
    This setting defines the type of the return values.

SelectionType   (page 1592)
    Sets the type of selection for this action.

StartNode   (page 1593)
    This setting sets the node from where the browsing should start when the action "NewObjectAddNodeAssignment" is used.

StartNode_{overridekeyname}   (page 1589)
    This setting overrides the start node.

5

StartNode_{overridekeyname}

Summary

This setting overrides the start node.

Usage

```
StartNode_{override key name}[key value]=<node>
```

Description

This setting overrides the start node.

The syntax begins with StartNode, an underscore "_", the override key name to match and the key value placed between the brackerts "[]".

This setting is case sensitive.

Examples

```
StartNode_classgroup[2]=users
```

5

ActionName

Summary

Sets the name of the action that needs to be used.

Usage

```
ActionName=<action name>
```

Description

Sets the name of the action that needs to be used.

This setting is case sensitive.

Examples

```
ActionName=SelectParentNode
```

"SelectParentNode" is the default value for this setting.

5

ReturnType

Summary

This setting defines the type of the return values.

Usage

```
ReturnType=NodeID | ObjectID
```

Description

This setting defines the type of the return values.

Possible values are:

- NodeID: Returns the node ids of the selected items
- ObjectID: Returns the object ids of the selected items

This setting is case sensitive.

Examples

```
ReturnType=NodeID
```

"NodeID" is the default value for this setting.

5

SelectionType

Summary

Sets the type of selection for this action.

Usage

```
SelectionType=single | multiple
```

Description

Sets the type of selection for this action.

Possible values are:

- single: Allows for just one item to be selected

- multiple: Allows for multiple items to be selected

This setting is case sensitive.

Examples

```
SelectionType=single
```

"single" is the default value for this setting.

5

StartNode

Summary

This setting sets the node from where the browsing should start when the action "NewObjectAddNodeAssignment" is used.

Usage

```
StartNode=<node>
```

Description

This setting sets the node from where the browsing should start when the action "NewObjectAddNodeAssignment" is used.

Nodes can be used as value, such as content, users, media, setup.

This setting is case sensitive.

Examples

```
StartNode=content
```

"content" is the default value for this setting.

5

[AddNodeAssignment]

This action is used for selecting a node placement for an object.

ReturnType   (page 1596)
　　This setting defines the type of the return values.

SelectionType   (page 1597)
　　Sets the type of selection for this action.

StartNode   (page 1598)
　　This setting sets the node from where the browsing should start when the action "AddNode-
　　Assignment" is used.

StartNode_{overridekeyname}   (page 1595)
　　This setting overrides the start node.

5

StartNode_{overridekeyname}

Summary

This setting overrides the start node.

Usage

```
StartNode_{override key name}[key value]=<node>
```

Description

This setting overrides the start node.

The syntax begins with StartNode, an underscore "_", the override key name to match and the key value placed between the brackerts "[]".

This setting is case sensitive.

Examples

```
StartNode_classgroup[2]=users
```

5

ReturnType

Summary

This setting defines the type of the return values.

Usage

```
ReturnType=NodeID | ObjectID
```

Description

This setting defines the type of the return values.

Possible values are:

- NodeID: Returns the node ids of the selected items
- ObjectID: Returns the object ids of the selected items

This setting is case sensitive.

Examples

```
ReturnType=NodeID
```

"NodeID" is the default value for this setting.

5

SelectionType

Summary

Sets the type of selection for this action.

Usage

```
SelectionType=single | multiple
```

Description

Sets the type of selection for this action.

Possible values are:

- single: Allows for just one item to be selected
- multiple: Allows for multiple items to be selected

This setting is case sensitive.

Examples

```
SelectionType=multiple
```

"Multiple" is the default value for this setting.

5

StartNode

Summary

This setting sets the node from where the browsing should start when the action "AddNodeAssignment" is used.

Usage

```
StartNode=<node>
```

Description

This setting sets the node from where the browsing should start when the action "AddNodeAssignment" is used.

Nodes are used as value, such as content, users, media and setup.

This setting is case sensitive.

Examples

```
StartNode=content
```

"content" is the default value for this setting.

5

[AddPrimaryNodeAssignment]

This action is used for selecting a node placement for an object.

ReturnType   (page 1601)
  This setting defines the type of the return values.

SelectionType   (page 1602)
  Sets the type of selection for this action.

StartNode   (page 1603)
  This setting sets the node from where the browsing should start when the action "AddPrimaryNodeAssignment" is used.

StartNode_{overridekeyname}   (page 1600)
  This setting overrides the start node for this action.

5

StartNode_{overridekeyname}

Summary

This setting overrides the start node for this action.

Usage

```
StartNode_{override key name}[key value]=<node>
```

Description

This setting overrides the start node.

The syntax begins with StartNode, an underscore "_", the override key name to match and the key value placed between the brackerts "[]".

Nodes are used as value, such as content, users, media and setup.

This setting is case sensitive.

Examples

```
StartNode_classgroup[2]=users
```

5

ReturnType

Summary

This setting defines the type of the return values.

Usage

```
ReturnType=NodeID | ObjectID
```

Description

This setting defines the type of the return values.

Possible values are:

- NodeID: Returns the node ids of the selected items
- ObjectID: Returns the object ids of the selected items

This setting is case sensitive.

Examples

```
ReturnType=NodeID
```

"NodeID" is the default value for this setting.

5

SelectionType

Summary

Sets the type of selection for this action.

Usage

```
SelectionType=single | multiple
```

Description

Sets the type of selection for this action.

Possible values are:

- single: Allows for just one item to be selected
- multiple: Allows for multiple items to be selected

This setting is case sensitive.

Examples

```
SelectionType=multiple
```

"multiple" is the default value for this setting.

5

StartNode

Summary

This setting sets the node from where the browsing should start when the action "AddPrimaryN-odeAssignment" is used.

Usage

```
StartNode=<node>
```

Description

This setting sets the node from where the browsing should start when the action "AddPrimaryN-odeAssignment" is used.

Nodes are used as value, such as content, users, media and setup.
This setting is case sensitive.

Examples

```
StartNode=content
```

"content" is the default value for this setting.

5

[MoveNodeAssignment]

This action is used for moving a node placement.

ReturnType   (page 1606)
   This setting defines the type of the return values.

SelectionType   (page 1607)
   Sets the type of selection for this action.

StartNode   (page 1608)
   This setting sets the node from where the browsing should start when the action "MoveN-
   odeAssignment" is used.

StartNode_{overridekeyname}   (page 1605)
   This setting overrides the start node.

5

StartNode_{overridekeyname}

Summary

This setting overrides the start node.

Usage

```
StartNode_{override key name}[key value]=<node>
```

Description

This setting overrides the start node.

The syntax begins with StartNode, an underscore "_", the override key name to match and the key value placed between the brackerts "[]".

Nodes are used as value, such as content, users, media and setup.

This setting is case sensitive.

Examples

```
StartNode_classgroup[2]=users
```

5

ReturnType

Summary

This setting defines the type of the return values.

Usage

```
ReturnType=NodeID | ObjectID
```

Description

This setting defines the type of the return values.

Possible values are:

- NodeID: Returns the node ids of the selected items
- ObjectID: Returns the object ids of the selected items

This setting is case sensitive.

Examples

```
ReturnType=NodeID
```

"NodeID" is the default value for this setting.

5

SelectionType

Summary

Sets the type of selection for this action.

Usage

```
SelectionType=single | multiple
```

Description

Sets the type of selection for this action.

Possible values are:

- single: Allows for just one item to be selected
- multiple: Allows for multiple items to be selected

This setting is case sensitive.

Examples

```
SelectionType=single
```

"single" is the default value for this setting.

5

StartNode

Summary

This setting sets the node from where the browsing should start when the action "MoveNodeAssignment" is used.

Usage

```
StartNode=<node>
```

Description

This setting sets the node from where the browsing should start when the action "MoveNodeAssignment" is used.

Nodes are used as value, such as content, users, media and setup.

This setting is case sensitive.

Examples

```
StartNode=content
```

"content" is the default value for this setting.

5

[MoveNode]

This action is used for moving a node.

ReturnType   (page 1611)
   This setting defines the type of the return values.

SelectionType   (page 1612)
   Sets the type of selection for this action.

StartNode   (page 1613)
   This setting sets the node from where the browsing should start when the action "MoveN-
   ode" is used.

StartNode_{overridekeyname}   (page 1610)
   This setting overrides the start node.

5

StartNode_{overridekeyname}

Summary

This setting overrides the start node.

Usage

```
StartNode_{override key name}[key value]=<node>
```

Description

This setting overrides the start node.

The syntax begins with StartNode, an underscore "_", the override key name to match and the key value placed between the brackerts "[]".

This setting is case sensitive.

Examples

```
StartNode_classgroup[2]=users
```

5

ReturnType

Summary

This setting defines the type of the return values.

Usage

```
ReturnType=NodeID | ObjectID
```

Description

This setting defines the type of the return values.

Possible values are:

- NodeID: Returns the node ids of the selected items
- ObjectID: Returns the object ids of the selected items

This setting is case sensitive.

Examples

```
ReturnType=NodeID
```

"NodeID" is the default value for this setting.

5

SelectionType

Summary

Sets the type of selection for this action.

Usage

```
SelectionType=single | multiple
```

Description

Sets the type of selection for this action.
Possible values are:

- single: Allows for just one item to be selected
- multiple: Allows for multiple items to be selected

This setting is case sensitive.

Examples

```
SelectionType=single
```

"single" is the default value for this setting.

5

StartNode

Summary

This setting sets the node from where the browsing should start when the action "MoveNode" is used.

Usage

```
StartNode=<node>
```

Description

This setting sets the node from where the browsing should start when the action "MoveNode" is used.

Nodes are used as value, such as content, users, media and setup.

This setting is case sensitive.

Examples

```
StartNode=content
```

"content" is the default value for this setting.

5

[SwapNode]

This action is used for swaping a node.

ReturnType   (page 1616)
    This setting defines the type of the return values.

SelectionType   (page 1617)
    Sets the type of selection for this action.

StartNode   (page 1618)
    This setting sets the node from where the browsing should start when the action "SwapNode" is used.

StartNode_{overridekeyname}   (page 1615)
    This setting overrides the start node.

5

StartNode_{overridekeyname}

Summary

This setting overrides the start node.

Usage

```
StartNode_{override key name}[key value]=<node>
```

Description

This setting overrides the start node.

The syntax begins with StartNode, an underscore "_", the override key name to match and the key value placed between the brackerts "[]".

Nodes are used as value, such as content, users, media and setup.

This setting is case sensitive.

Examples

```
StartNode_classgroup[2]=users
```

5

ReturnType

Summary

This setting defines the type of the return values.

Usage

```
ReturnType=NodeID | ObjectID
```

Description

This setting defines the type of the return values.

Possible values are:

- NodeID: Returns the node ids of the selected items
- ObjectID: Returns the object ids of the selected items

This setting is case sensitive.

Examples

```
ReturnType=NodeID
```

"NodeID" is the default value for this setting.

5

SelectionType

Summary

Sets the type of selection for this action.

Usage

```
SelectionType=single | multiple
```

Description

Sets the type of selection for this action.

Possible values are:

- single: Allows for just one item to be selected
- multiple: Allows for multiple items to be selected

This setting is case sensitive.

Examples

```
SelectionType=single
```

"single" is the default value for this setting.

5

StartNode

Summary

This setting sets the node from where the browsing should start when the action "SwapNode" is used.

Usage

```
StartNode=<node>
```

Description

This setting sets the node from where the browsing should start when the action "SwapNode" is used.

Nodes are used as value, such as content, users, media and setup.

This setting is case sensitive.

Examples

```
StartNode=content
```

"content" is the default value for this setting.

5

[AddRelatedObjects]

This action is used for finding related objects.

ReturnType   (page 1620)
    This setting defines the type of the return values.

SelectionType   (page 1621)
    Sets the type of selection for this action.

StartNode   (page 1622)
    This setting sets the node from where the browsing should start when the action "AddRelatedObject" is used.

5

ReturnType

Summary

This setting defines the type of the return values.

Usage

```
ReturnType=NodeID | ObjectID
```

Description

This setting defines the type of the return values.
Possible values are:

- NodeID: Returns the node ids of the selected items
- ObjectID: Returns the object ids of the selected items

This setting is case sensitive.

Examples

```
ReturnType=ObjectID
```

"ObjectID" is the default value for this setting.

5

SelectionType

Summary

Sets the type of selection for this action.

Usage

```
SelectionType=single | multiple
```

Description

Sets the type of selection for this action.

Possible values are:

- single: Allows for just one item to be selected

- multiple: Allows for multiple items to be selected

This setting is case sensitive.

Examples

```
SelectionType=multiple
```

"multiple" is the default value for this setting.

5

StartNode

Summary

This setting sets the node from where the browsing should start when the action "AddRelatedObject" is used.

Usage

```
StartNode=<node>
```

Description

This setting sets the node from where the browsing should start when the action "AddRelatedObject" is used.

Nodes are used as value, such as content, users, media and setup.

This setting is case sensitive.

Examples

```
StartNode=content
```

"content" is the default value for this setting.

5

[AddRelatedObjectsToDataType]

This action for finding objects to related object datatype.

ReturnType   (page 1624)
    This setting defines the type of the return values.

SelectionType   (page 1625)
    Sets the type of selection for this action.

StartNode   (page 1626)
    This setting sets the node from where the browsing should start when the action "Ad-dRelatedObjectToDataType" is used.

5

ReturnType

Summary

This setting defines the type of the return values.

Usage

```
ReturnType=NodeID | ObjectID
```

Description

This setting defines the type of the return values.

Possible values are:

- NodeID: Returns the node ids of the selected items
- ObjectID: Returns the object ids of the selected items

This setting is case sensitive.

Examples

```
ReturnType=ObjectID
```

"ObjectID" is the default value for this setting.

5

SelectionType

Summary

Sets the type of selection for this action.

Usage

```
SelectionType=single | multiple
```

Description

Sets the type of selection for this action.

Possible values are:

- single: Allows for just one item to be selected
- multiple: Allows for multiple items to be selected

This setting is case sensitive.

Examples

```
SelectionType=single
```

"single" is the default value for this setting.

5

StartNode

Summary

This setting sets the node from where the browsing should start when the action "AddRelatedObjectToDataType" is used.

Usage

```
StartNode=<node>
```

Description

This setting sets the node from where the browsing should start when the action "AddRelatedObjectToDataType" is used.

Nodes are used as value, such as content, users, media and setup.

This setting is case sensitive.

Examples

```
StartNode=content
```

"content" is the default value for this setting.

5

[AddRelatedObjectsToDataType]

This action for finding objects to related object datatype.

ReturnType   (page 1628)
>   This setting defines the type of the return values.

SelectionType   (page 1629)
>   Sets the type of selection for this action.

StartNode   (page 1630)
>   This setting sets the node from where the browsing should start when the action "Ad-
>   dRelatedObjectToDataType" is used.

5

ReturnType

Summary

This setting defines the type of the return values.

Usage

```
ReturnType=NodeID | ObjectID
```

Description

This setting defines the type of the return values.

Possible values are:

- NodeID: Returns the node ids of the selected items
- ObjectID: Returns the object ids of the selected items

This setting is case sensitive

Examples

```
ReturnType=ObjectID
```

"ObjectID" is the default value for this setting.

5

SelectionType

Summary

Sets the type of selection for this action.

Usage

```
SelectionType=single | multiple
```

Description

Sets the type of selection for this action.

Possible values are:

- single: Allows for just one item to be selected

- multiple: Allows for multiple items to be selected

This setting is case sensitive.

Examples

```
SelectionType=single
```

"single" is the default value for this setting.

5

StartNode

Summary

This setting sets the node from where the browsing should start when the action "AddRelatedObjectToDataType" is used.

Usage

```
StartNode=<node>
```

Description

This setting sets the node from where the browsing should start when the action "AddRelatedObjectToDataType" is used.

Nodes are used as value, such as content, users, media and setup.

This setting is case sensitive.

Examples

```
StartNode=content
```

"content" is the default value for this setting

5

[AddRelatedObjectsListToDataType]

This action is used for finding objects to related object list datatype.

ReturnType   (page 1632)
> This setting defines the type of the return values.

SelectionType   (page 1633)
> Sets the type of selection for this action.

StartNode   (page 1634)
> This setting sets the node from where the browsing should start when the action "AddRelatedObjectListToDataType" is used.

ReturnType

Summary

This setting defines the type of the return values.

Usage

```
ReturnType=NodeID | ObjectID
```

Description

This setting defines the type of the return values.

Possible values are:

- NodeID: Returns the node ids of the selected items
- ObjectID: Returns the object ids of the selected items

This setting is case sensitive.

Examples

```
ReturnType=ObjectID
```

"ObjectID" is the default value for this setting.

5

SelectionType

Summary

Sets the type of selection for this action.

Usage

```
SelectionType=single | multiple
```

Description

Sets the type of selection for this action.

Possible values are:

- single: Allows for just one item to be selected
- multiple: Allows for multiple items to be selected

This setting is case sensitive.

Examples

```
SelectionType=multiple
```

"multiple" is the default value for this setting.

5

StartNode

Summary

This setting sets the node from where the browsing should start when the action "AddRelatedObjectListToDataType" is used.

Usage

```
StartNode=<node>
```

Description

This setting sets the node from where the browsing should start when the action "AddRelatedObjectListToDataType" is used.
Nodes are used as value, such as content, users, media and setup.
This setting is case sensitive.

Examples

```
StartNode=content
```

This is the default setting.

5

[AddRelatedImageToDataType]

This action is used for finding images to related object datatype.

ReturnType   (page 1636)
> This setting defines the type of the return values.

SelectionType   (page 1637)
> Sets the type of selection for this action.

StartNode   (page 1638)
> This setting sets the node from where the browsing should start when the action "AddRelatedImageToDataType" is used.

5

ReturnType

Summary

This setting defines the type of the return values.

Usage

```
ReturnType=NodeID | ObjectID
```

Description

This setting defines the type of the return values.
Possible values are:

- NodeID: Returns the node ids of the selected items
- ObjectID: Returns the object ids of the selected items

This setting is case sensitive.

Examples

```
ReturnType=ObjectID
```

"ObjectID" is the default value for this setting.

5

SelectionType

Summary

Sets the type of selection for this action.

Usage

```
SelectionType=single | multiple
```

Description

Sets the type of selection for this action.
Possible values are:

- single: Allows for just one item to be selected
- multiple: Allows for multiple items to be selected

This setting is case sensitive.

Examples

```
SelectionType=single
```

"single" is the default value for this setting.

5

StartNode

Summary

This setting sets the node from where the browsing should start when the action "AddRelatedImageToDataType" is used.

Usage

```
StartNode=<node>
```

Description

This setting sets the node from where the browsing should start when the action "AddRelatedImageToDataType" is used.

Nodes are used as value, such as content, users, media and setup.

This setting is case sensitive.

Examples

```
StartNode=media
```

"media" is the default value for this setting.

5

[AddBookmark]

This action is used for finding objects to related object datatype.

ReturnType   (page 1640)
    This setting defines the type of the return values.

SelectionType   (page 1641)
    Sets the type of selection for this action.

StartNode   (page 1642)
    This setting sets the node from where the browsing should start when the action "AddBook-
    mark" is used.

5

ReturnType

Summary

This setting defines the type of the return values.

Usage

```
ReturnType=NodeID | ObjectID
```

Description

This setting defines the type of the return values.

Possible values are:

- NodeID: Returns the node ids of the selected items
- ObjectID: Returns the object ids of the selected items

This setting is case sensitive.

Examples

```
ReturnType=NodeID
```

"NodeID" is the default value for this setting.

SelectionType

Summary

Sets the type of selection for this action.

Usage

```
SelectionType=single | multiple
```

Description

Sets the type of selection for this action.

Possible values are:

- single: Allows for just one item to be selected
- multiple: Allows for multiple items to be selected

This setting is case sensitive.

Examples

```
SelectionType=multiple
```

"multiple" is the default value for this setting.

5

StartNode

Summary

This setting sets the node from where the browsing should start when the action "AddBookmark" is used.

Usage

```
StartNode=<node>
```

Description

This setting sets the node from where the browsing should start when the action "AddBookmark" is used.
Nodes are used as value, such as content, users, media and setup.
This setting is case sensitive.

Examples

```
StartNode=content
```

"content" is the default value for this setting.

5

[AddSubtreeSubscribingNode]

This action is used for finding nodes when adding bookmarks.

ReturnType   (page 1644)
> This setting defines the type of the return values.

SelectionType   (page 1645)
> Sets the type of selection for this action.

StartNode   (page 1646)
> This setting sets the node from where the browsing should start when the action "AddSub-treeSubscribingNode" is used.

5

ReturnType

Summary

This setting defines the type of the return values.

Usage

```
ReturnType=NodeID | ObjectID
```

Description

This setting defines the type of the return values.

Possible values are:

- NodeID: Returns the node ids of the selected items
- ObjectID: Returns the object ids of the selected items

This setting is case sensitive.

Examples

```
ReturnType=NodeID
```

"NodeID" is the default value for this setting.

5

SelectionType

Summary

Sets the type of selection for this action.

Usage

```
SelectionType=single | multiple
```

Description

Sets the type of selection for this action.

Possible values are:

- single: Allows for just one item to be selected
- multiple: Allows for multiple items to be selected

This setting is case sensitive.

Examples

```
SelectionType=multiple
```

"multiple" is the default value for this setting.

5

StartNode

Summary

This setting sets the node from where the browsing should start when the action "AddSubtreeSub-scribingNode" is used.

Usage

```
StartNode=<node>
```

Description

This setting sets the node from where the browsing should start when the action "AddSubtreeSub-scribingNode" is used.
Nodes are used as value, such as content, users, media and setup.
This setting is case sensitive.

Examples

```
StartNode=content
```

"content" is the default value for this setting.

5

[SelectObjectRelationListNode]

This action is used for finding placement for object relation list.

ReturnType   (page 1648)
    This setting defines the type of the return values.

SelectionType   (page 1649)
    Sets the type of selection for this action.

StartNode   (page 1650)
    This setting sets the node from where the browsing should start when the action "SelectO-
    bjectRelationListNode" is used.

5

ReturnType

Summary

This setting defines the type of the return values.

Usage

```
ReturnType=NodeID | ObjectID
```

Description

This setting defines the type of the return values.
Possible values are:

- NodeID: Returns the node ids of the selected items

- ObjectID: Returns the object ids of the selected items

This setting is case sensitive.

Examples

```
ReturnType=NodeID
```

"NodeID" is the default value for this setting.

5

SelectionType

Summary

Sets the type of selection for this action.

Usage

```
SelectionType=single | multiple
```

Description

Sets the type of selection for this action.
Possible values are:

- single: Allows for just one item to be selected

- multiple: Allows for multiple items to be selected

This setting is case sensitive.

Examples

```
SelectionType=single
```

"single" is the default value for this setting.

5

StartNode

Summary

This setting sets the node from where the browsing should start when the action "SelectObjectRelationListNode" is used.

Usage

```
StartNode=content | users | media | setup
```

Description

This setting sets the node from where the browsing should start when the action "SelectObjectRelationListNode" is used.
Nodes are used as value, such as content, users, media and setup.
This setting is case sensitive.

Examples

```
StartNode=content
```

"content" is the default value for this setting.

5

[SelectObjectRelationNode]

This action is used for finding placement for objectrelation.

ReturnType   (page 1652)
This setting defines the type of the return values.

SelectionType   (page 1653)
Sets the type of selection for this action.

StartNode   (page 1654)
This setting sets the node from where the browsing should start when the action "SelectO-bjectgRelationNode" is used.

5

ReturnType

Summary

This setting defines the type of the return values.

Usage

```
ReturnType=NodeID | ObjectID
```

Description

This setting defines the type of the return values.
Possible values are:

- NodeID: Returns the node ids of the selected items
- ObjectID: Returns the object ids of the selected items

This setting is case sensitive.

Examples

```
ReturnType=NodeID
```

"NodeID" is the default value for this setting.

5

SelectionType

Summary

Sets the type of selection for this action.

Usage

```
SelectionType=single | multiple
```

Description

Sets the type of selection for this action.
Possible values are:

- single: Allows for just one item to be selected

- multiple: Allows for multiple items to be selected

This setting is case sensitive.

Examples

```
SelectionType=single
```

"single" is the default value for this setting

5

StartNode

Summary

This setting sets the node from where the browsing should start when the action "SelectObject-gRelationNode" is used.

Usage

```
StartNode=<node>
```

Description

This setting sets the node from where the browsing should start when the action "SelectObjec-tRelationNode" is used.

Nodes are used as value, such as content, users, media and setup.

This setting is case sensitive.

Examples

```
StartNode=content
```

"content" is the default value for this setting.

5

[SelectToolbarNode]

This action is used for finding placement for toolbar node.

ReturnType   (page 1656)
  This setting defines the type of the return values.

SelectionType   (page 1657)
  Sets the type of selection for this action.

StartNode   (page 1658)
  This setting sets the node from where the browsing should start when the action "Select-ToolbarNode" is used.

5

ReturnType

Summary

This setting defines the type of the return values.

Usage

```
ReturnType=NodeID | ObjectID
```

Description

This setting defines the type of the return values.
Possible values are:

- NodeID: Returns the node ids of the selected items

- ObjectID: Returns the object ids of the selected items

This setting is case sensitive.

Examples

```
ReturnType=NodeID
```

"NodeID" is the default value for this setting.

5

SelectionType

Summary

Sets the type of selection for this action.

Usage

```
SelectionType=single | multiple
```

Description

Sets the type of selection for this action.
Possible values are:

- single: Allows for just one item to be selected

- multiple: Allows for multiple items to be selected

This setting is case sensitive.

Examples

```
SelectionType=single
```

"single" is the default value for this setting.

5

StartNode

Summary

This setting sets the node from where the browsing should start when the action "SelectToolbarNode" is used.

Usage

```
StartNode=<node>
```

Description

This setting sets the node from where the browsing should start when the action "SelectToolbarNode" is used.
Nodes such as content, users, media and setup are used as value.
This setting is case sensitive.

Examples

```
StartNode=content
```

"content" is the default value for this setting.

5

[SelectToolbarNodePath]

This action for finding placement for toolbar node.

ReturnType   (page 1660)
    This setting defines the type of the return values.

SelectionType   (page 1661)
    Sets the type of selection for this action.

StartNode   (page 1662)
    This setting sets the node from where the browsing should start when the action "Select-ToolbarNodePath" is used.

5

ReturnType

Summary

This setting defines the type of the return values.

Usage

```
ReturnType=NodeID | ObjectID
```

Description

This setting defines the type of the return values.
Possible values are:

- NodeID: Returns the node ids of the selected items

- ObjectID: Returns the object ids of the selected items

This setting is case sensitive.

Examples

```
ReturnType=NodeID
```

"NodeID" is the default value for this setting.

5

SelectionType

Summary

Sets the type of selection for this action.

Usage

```
SelectionType=single | multiple
```

Description

Sets the type of selection for this action.
Possible values are:

- single: Allows for just one item to be selected

- multiple: Allows for multiple items to be selected

This setting is case sensitive.

Examples

```
SelectionType=single
```

"single" is the default value for this setting.

5

StartNode

Summary

This setting sets the node from where the browsing should start when the action "SelectTool-barNodePath" is used.

Usage

```
StartNode=<node>
```

Description

This setting sets the node from where the browsing should start when the action "SelectTool-barNodePath" is used.
Nodes such as content, users, media and setup are used as value.
This setting is case sensitive.

Examples

```
StartNode=content
```

"content" is the default value for this setting.

5

[CopyNode]

This action is used for copying a node.

ReturnType   (page 1664)
      This setting defines the type of the return values.

SelectionType   (page 1665)
      Sets the type of selection for this action.

StartNode   (page 1666)
      This setting sets the node from where the browsing should start when the action "CopyN-
      ode" is used.

5

ReturnType

Summary

This setting defines the type of the return values.

Usage

```
ReturnType=NodeID | ObjectID
```

Description

This setting defines the type of the return values.
Possible values are:

- NodeID: Returns the node ids of the selected items

- ObjectID: Returns the object ids of the selected items

This setting is case sensitive.

Examples

```
ReturnType=NodeID
```

"NodeID" is the default value for this setting.

5

SelectionType

Summary

Sets the type of selection for this action.

Usage

```
SelectionType=single | multiple
```

Description

Sets the type of selection for this action.
Possible values are:

- single: Allows for just one item to be selected

- multiple: Allows for multiple items to be selected

This setting is case sensitive.

Examples

```
SelectionType=single
```

"single" is the default value for this setting.

5

StartNode

Summary

This setting sets the node from where the browsing should start when the action "CopyNode" is used.

Usage

```
StartNode=<node>
```

Description

This setting sets the node from where the browsing should start when the action "CopyNode" is used.
Nodes such as content, users, media and setup are used as value.
This setting is case sensitive.

Examples

```
StartNode=content
```

"content" in the default value for this setting.

5

[CopySubtree]

This action is used for copying subtree.

ReturnType   (page 1668)
> This setting defines the type of the return values.

SelectionType   (page 1669)
> Sets the type of selection for this action.

StartNode   (page 1670)
> This setting sets the node from where the browsing should start when the action "Copy-Subtree" is used.

5

ReturnType

Summary

This setting defines the type of the return values.

Usage

```
ReturnType=NodeID | ObjectID
```

Description

This setting defines the type of the return values.
Possible values are:

- NodeID: Returns the node ids of the selected items

- ObjectID: Returns the object ids of the selected items

This setting is case sensitive.

Examples

```
ReturnType=NodeID
```

"NodeID" is the default value for this setting.

5

SelectionType

Summary

Sets the type of selection for this action.

Usage

```
SelectionType=single | multiple
```

Description

Sets the type of selection for this action.
Possible values are:

- single: Allows for just one item to be selected

- multiple: Allows for multiple items to be selected

This setting is case sensitive.

Examples

```
SelectionType=single
```

"single" is the default value for this setting.

5

StartNode

Summary

This setting sets the node from where the browsing should start when the action "CopySubtree" is used.

Usage

```
StartNode=<node>
```

Description

This setting sets the node from where the browsing should start when the action "CopySubtree" is used.

Nodes such as content, users, media and setup are used as value.

This setting is case sensitive.

Examples

```
StartNode=content
```

"content" is the default value for this setting.

5

[SelectSingleUser]

This action is used for adding single user limitation.

ReturnType   (page 1672)
    This setting defines the type of the return values.

SelectionType   (page 1673)
    Sets the type of selection for this action.

StartNode   (page 1674)
    This setting sets the node from where the browsing should start when the action "SelectS-ingleUser" is used.

5

ReturnType

Summary

This setting defines the type of the return values.

Usage

```
ReturnType=NodeID | ObjectID
```

Description

This setting defines the type of the return values.
Possible values are:

- NodeID: Returns the node ids of the selected items

- ObjectID: Returns the object ids of the selected items

This setting is case sensitive.

Examples

```
ReturnType=ObjectID
```

"objectID" is the default value for this setting.

5

SelectionType

Summary

Sets the type of selection for this action.

Usage

```
SelectionType=single | multiple
```

Description

Sets the type of selection for this action.
Possible values are:

- single: Allows for just one item to be selected

- multiple: Allows for multiple items to be selected

This setting is case sensitive.

Examples

```
SelectionType=single
```

"single" is the default value for this setting.

5

StartNode

Summary

This setting sets the node from where the browsing should start when the action "SelectSingleUser" is used.

Usage

```
StartNode=<node>
```

Description

This setting sets the node from where the browsing should start when the action "SelectSingleUser" is used.

Nodes such as content, users, media and setup are used as value.

This setting is case sensitive.

Examples

```
StartNode=users
```

"users" is the default value for this setting.

5

[SelectMultipleUsers]

This action is used for adding multiple users and groups limitation.

ReturnType   (page 1676)
> This setting defines the type of the return values.

SelectionType   (page 1677)
> Sets the type of selection for this action.

StartNode   (page 1678)
> This setting sets the node from where the browsing should start when the action "Select-MultipleUsers" is used.

5

ReturnType

Summary

This setting defines the type of the return values.

Usage

```
ReturnType=NodeID | ObjectID
```

Description

This setting defines the type of the return values.
Possible values are:

- NodeID: Returns the node ids of the selected items

- ObjectID: Returns the object ids of the selected items

This setting is case sensitive.

Examples

```
ReturnType=ObjectID
```

"ObjectID" is the default value for this setting.

5

SelectionType

Summary

Sets the type of selection for this action.

Usage

```
SelectionType=single | multiple
```

Description

Sets the type of selection for this action.
Possible values are:

- single: Allows for just one item to be selected

- multiple: Allows for multiple items to be selected

This setting is case sensitive.

Examples

```
SelectionType=multiple
```

"Multiple" is the default value for this setting.

5

StartNode

Summary

This setting sets the node from where the browsing should start when the action "SelectMultipleUsers" is used.

Usage

```
StartNode=<node>
```

Description

This setting sets the node from where the browsing should start when the action "SelectMultipleUsers" is used.

Nodes such as content, users, media and setup are used as value.

This setting is case sensitive.

Examples

```
StartNode=users
```

"users" is the default value for this setting.

5

[ExampleAction]

This is an example of an Action block! The block needs to be named the same as the action name.

ReturnType (page 1681)
    This setting defines the type of the return values.

SelectionType (page 1682)
    Sets the type of selection for this action.

StartNode (page 1683)
    This setting sets the node from where the browsing should start.

StartNode_{overridekeyname} (page 1680)
    This setting allows you to override the start node.

5

StartNode_{overridekeyname}

Summary

This setting allows you to override the start node.

Usage

```
StartNode_{override key name}[key value]=<node>
```

Description

This setting allows you to override the start node.

The syntax begins with "StartNode", an underscore "_", the override key name to match and the key value placed between the brackets "[]".

Nodes such as content, users, madia and setup are used as value.

This setting is case sensitive.

Examples

```
StartNode_classgroup[2]=users
```

5

ReturnType

Summary

This setting defines the type of the return values.

Usage

```
ReturnType=NodeID | ObjectID
```

Description

This setting defines the type of the return values.

Possible values are:

- NodeID: Returns the node ids of the selected items
- ObjectID: Returns the object ids of the selected items

This setting is case sensitive.

Examples

```
ReturnType=NodeID
```

5

SelectionType

Summary

Sets the type of selection for this action.

Usage

```
SelectionType=single | multiple
```

Description

Sets the type of selection for this action.

Possible values are:

- single: Allows for just one item to be selected
- multiple: Allows for multiple items to be selected

This setting is case sensitive.

Examples

```
SelectionType=single
```

5

StartNode

Summary

This setting sets the node from where the browsing should start.

Usage

```
StartNode=content | users | media | setup
```

Description

This setting sets the node from where the browsing should start.

This setting is case sensitive.

Examples

```
StartNode=content
```

5

[AddCustomer]

This action is used when a user should be selected in discount view.

ReturnType   (page 1688)
   This setting defines the type of the return values.

SelectionType   (page 1687)
   Sets the type of selection for this action.

StartNode   (page 1686)
   This setting sets the node from where the browsing should start when the action "AddCustomer" is used.

TopLevelNodes   (page 1685)
   Lists the top level nodes to show in browse page.

5

TopLevelNodes

Summary

Lists the top level nodes to show in browse page.

Usage

```
TopLevelNodes[]=<node>
```

Description

Lists the top level nodes to show in browse page for this action. Possible values are content, users, media and setup.

This setting is case sensitive.

Examples

```
TopLevelNodes[]=users
```

5

StartNode

Summary

This setting sets the node from where the browsing should start when the action "AddCustomer" is used.

Usage

```
StartNode=<node>
```

Description

This setting sets the node from where the browsing should start when the action "AddCustomer" is used.

Nodes such as content, users, media and setup are used as value.

This setting is case sensitive.

Examples

```
StartNode=users
```

"users" is the default value for this setting.

5

SelectionType

Summary

Sets the type of selection for this action.

Usage

```
SelectionType=single | multiple
```

Description

Sets the type of selection for this action.

Possible values are:

- single: Allows for just one item to be selected
- multiple: Allows for multiple items to be selected

This setting is case sensitive.

Examples

```
SelectionType=multiple
```

"multiple" is the default value for this setting.

5

ReturnType

Summary

This setting defines the type of the return values.

Usage

```
ReturnType=NodeID | ObjectID
```

Description

This setting defines the type of the return values.

Possible values are:

- NodeID: Returns the node ids of the selected items
- ObjectID: Returns the object ids of the selected items

This setting is case sensitive.

Examples

```
ReturnType=ObjectID
```

"ObjectID" is the default value for this setting.

5

5.14.4 collaboration.ini

5.14.5   collect.ini

The configuration blocks are documented in the following sections:

- [CollectionSettings] (page 1691)
- [DisplaySettings] (page 1698)
- [EmailSettings] (page 1705)
- [InfoSettings] (page 1709)

5

[CollectionSettings]

CollectAnonymousData   (page 1697)
  Sets if information from anonymous users can be collected.

CollectAnonymousDataAttribute   (page 1695)
  Sets which attribute in the object determines if collecting anonymous data is permitted.

CollectAnonymousDataList   (page 1696)
  Sets if information from anonymous users can be collected for specific classes.

CollectionUserData   (page 1694)
  Sets how information collection is handled in terms of user identification

CollectionUserDataAttribute   (page 1692)
  Sets which attribute in the object determines how information in collected.

CollectionUserDataList   (page 1693)
  Sets how information collection is handled in terms of user identification for specific classes.

5

CollectionUserDataAttribute

Summary

Sets which attribute in the object determines how information in collected.

Usage

```
CollectionUserDataAttribute=<attribute>
```

Description

Sets which attribute to look for in the object to determine how information in collected. If this attribute is found, the content of the attribute will be used as type.
This setting is case sensitive.

Examples

```
CollectionUserDataAttribute=collection_userdata
```

This is the default setting.

5

CollectionUserDataList

Summary

Sets how information collection is handled in terms of user identification for specific classes.

Usage

```
CollectionUserDataList[<class ID>]=multiple | unique | overwrite
```

Description

Just like the setting CollectionUserData in the block [CollectionSettings], this setting sets how information collection is handled in terms of user identification. But it overrides the setting CollectionUserData for specific classes by matching class ID's or identifiers to the information collection type.
The following values are possible:

- multiple: this setting allows each user to submit multiple data.

- unique: this setting allows each user to submit only one set of data. If a set of data already exists, a warning is given.

- overwrite: this setting allows each user to submit only one set of data. If a set of data already exists, the new entry will overwrite the old one.

This setting is case sensitive.

Examples

```
[CollectionSettings]

CollectionUserData=multiple

CollectionUserDataList[poll]=unique
```

By default collecting user information allows for users to submit multiple data. But CollectionUserDataList overrides this, so for polls users are only allowed to submit one set of data. If they give a second set of data a warning will be given.

5

CollectionUserData

Summary

Sets how information collection is handled in terms of user identification

Usage

```
CollectionUserData= multiple | unique | overwrite
```

Description

This setting sets how information collection is handled in terms of user identification. The following values are possible:

- multiple: this setting allows each user to submit multiple data.

- unique: this setting allows each user to submit only one set of data. If a set of data already exists, a warning is given.

- overwrite: this setting allows each user to submit only one set of data. If a set of data already exists, the new entry will overwrite the old one.

This setting is the default setting, an override can be defined in the setting CollectionUserDataList in the block [CollectionSettings].
This setting is case sensitive.

Examples

```
CollectionUserData=multiple
```

This is the default setting.

5

CollectAnonymousDataAttribute

Summary

Sets which attribute in the object determines if collecting anonymous data is permitted.

Usage

```
CollectAnonymousDataAttribute=<attribute>
```

Description

Sets which attribute to look for in the object to determine if collecting anonymous data is permitted. If this attribute is found, the content of the attribute will be used as type.
This setting is case sensitive.

Examples

```
CollectAnonymousDataAttribute=collection_anonymous
```

This is the default setting.

5

CollectAnonymousDataList

Summary

Sets if information from anonymous users can be collected for specific classes.

Usage

```
CollectAnonymousDataList[<information collection type>]=enabled | disabled
```

Description

Just like the setting CollectAnonymousData in the block [CollectionSettings], this setting defines if information from anonymous users can be collected. But CollectAnonymousDataList will in fact list the override settings of specific information collection types.
This setting is case sensitive.

Examples

```
[CollectionSettings]

CollectAnonymousData=enabled

CollectAnonymousDataList[poll]=disabled
```

When the setting CollectAnonymousData is set to enabled, collecting information from anonymous users will be possible by default. But CollectAnonymousDataList overrides the previous setting, therefore, in the case of this example, the system is not allowed to collect information from anonymous users for polls.

5

CollectAnonymousData

Summary

Sets if information from anonymous users can be collected.

Usage

```
CollectAnonymousData= enabled | disabled
```

Description

This setting defines if information from anonymous users can be collected. This is however the default value and an override can be set for specific information collection types with the setting CollectAnonymousDataList in the block [CollectionSettings].
This setting is case sensitive.

Examples

```
CollectAnonymousData=enabled
```

This is the default setting.

5

[DisplaySettings]

Display   (page 1704)
> This setting will set which display to show after information is collected.

DisplayAttribute   (page 1702)
> Sets which attribute of the object determines the display page.

DisplayList   (page 1703)
> Sets which display to show after information is collected for specific classes

RedirectURL   (page 1701)
> Sets the URL to redirect to.

RedirectURLAttribute   (page 1699)
> Sets which attribute in the object determines the URL to redirect to.

RedirectURLList   (page 1700)
> Sets the URL to redirect to for specific classes.

5

RedirectURLAttribute

Summary

Sets which attribute in the object determines the URL to redirect to.

Usage

```
RedirectURLAttribute=<attribute>
```

Description

Sets which attribute to look for in the object to determine which URL to redirect to. If this
attribute is found, the content of the attribute will be used as type.
This setting is case sensitive.

Examples

```
RedirectURLAttribute=collection_redirect
```

This is the default setting.

5

RedirectURLList

Summary

Sets the URL to redirect to for specific classes.

Usage

```
RedirectURLList[<information collection type>]=<URL>
```

Description

Just like the setting RedirectURL in the block [DisplaySettings], this setting sets the URL to redirect to, if the setting DisplayHandling in the block [DisplaySettings] is set to redirect. RedirectURLList however overrides the default setting RedirectURL, by matching the URL to specific collection types.
This setting is case sensitive.

Examples

```
[DisplaySettings]

RedirectURL=http://ez.no

RedirectURLList[feedback]=http://ez.no/doc
```

By default the page redirected to would be http://ez.no, but for the feedback class this is overridden to http://ez.no/doc.

5

RedirectURL

Summary

Sets the URL to redirect to.

Usage

```
RedirectURL=<URL>
```

Description

This setting sets the URL to redirect to, if the setting DisplayHandling in the block [DisplaySet-
tings] is set to redirect. This is however the default setting, and overrides can be set for specific
collection types with the setting
RedirectURLList in the block [DisplaySettings].
This setting is case sensitive.

Examples

```
RedirectURL=/
```

This is the default setting.

5

DisplayAttribute

Summary

Sets which attribute of the object determines the display page.

Usage

```
DisplayAttribute=<attribute>
```

Description

Sets which attribute to look for in the object to determine which display to show. If this attribute is found, the content of the attribute will be used as type.
This setting is case sensitive.

Examples

```
DisplayAttribute=collection_display
```

This is the default setting.

5

DisplayList

Summary

Sets which display to show after information is collected for specific classes

Usage

```
DisplayList[<class>]=result | redirect | node
```

Description

This setting will set which display to show after information is collected for specific classes, and will thus override the setting defined in Display in block [DisplaySettings].
It does this by matching class ID or identifier to an information collection type.
Possible values are:

- result: this value will display the Information collection result, for instance the poll result or form data.

- redirect: this value will redirect you to a specific URL defined in the setting RedirectURL in the block [DisplaySettings]

- node: this value will redirect you back to the content node

This setting is case sensitive.

Examples

```
[DisplaySettings]

Display=redirectDisplayList[poll]=result
```

By setting Display to redirect, the default display setting will be to redirect to an URL specified in RedirectURL in the block [DisplaySettings]. But for the class poll, the default is overridden to display the result instead.

5

Display

Summary

This setting will set which display to show after information is collected.

Usage

```
Display= result | redirect | node
```

Description

This setting will set which display to show by default after information is collected. An override can be set with DisplayList in block [DisplaySettings].
Possible values are:

- result: this value will display the information collection results, for instance the poll result or form data.

- redirect: this value will redirect you to a specific URL defined in the setting RedirectURL in the block [DisplaySettings]

- node: this value will redirect you back to the content node

This setting is case sensitive.

Examples

```
Display=result
```

This is the default setting.

5

[EmailSettings]

SendEmail   (page 1708)
    Sets if the installation is allowed to send out emails.

SendEmailAttribute   (page 1706)
    Sets which attribute in the object determines if an email should be sent.

SendEmailList   (page 1707)
    Sets if the installation is allowed to send out emails for specific classes.

5

SendEmailAttribute

Summary

Sets which attribute in the object determines if an email should be sent.

Usage

```
SendEmailAttribute=<attribute>
```

Description

Sets which attribute to look for in the object to determine if an email should be sent. If this attribute is found, the content of the attribute will be used as type.
This setting is case sensitive.

Examples

```
SendEmailAttribute=collection_email
```

This is the default setting.

5

SendEmailList

Summary

Sets if the installation is allowed to send out emails for specific classes.

Usage

```
SendEmailList[<class>]=enabled | disabled
```

Description

This setting list the information collection classes, and defines if sending email will be disabled or enabled for specific class id or identifier.
This setting overrides the default defined in SendEmail in block [EmailSettings] for specific classes.
This setting is case sensitive.

Examples

```
[EmailSettings]

SendEmail=enabled

SendEmailList[poll]=disable
```

Although by default SendEmail is set to enabled, this setting will disabled this feature for polls.

5

SendEmail

Summary

Sets if the installation is allowed to send out emails.

Usage

```
SendEmail= enabled | disabled
```

Description

Sets if the installation is allowed to send out emails.
An override of this setting can be defined with the setting SendEmailList in block [EmailSettings].
This setting is case sensitive.

Examples

```
SendEmail=enabled
```

This is the default setting.

5

[InfoSettings]

Type   (page 1712)
     Sets the default information collection type.

TypeAttribute   (page 1710)
     Sets which attribute in an object determines the information collection type.

TypeList   (page 1711)
     Sets the information collection type for specific class ID's or identifiers.

5

TypeAttribute

Summary

Sets which attribute in an object determines the information collection type.

Usage

```
TypeAttribute=<attribute>
```

Description

Sets which attribute to look for in the object to determine the information collection type to use.
If this attribute is found, the content of the attribute will be used as type.
This setting is case sensitive.

Examples

```
TypeAttribute=collection_type
```

This is the default setting.

5

TypeList

Summary

Sets the information collection type for specific class ID's or identifiers.

Usage

```
TypeList[<class ID | class identifier>]=<collection type>
```

Description

Similar to the setting Type in the block [InfoSettings], this setting lists the information collection types, but matches them to class ID's or identifiers. By doing this an override can be set for the setting Type for specific classes.
The collection types used as value are:

- poll

- form

- feedback

All values have their own template located here:

```
<eZ Publish installation>\templates\content\collectedinfo
```

This setting is case sensitive.

Examples

```
[InfoSettings]

Type=form

TypeList[poll]=poll
TypeList[form]=form
TypeList[feedback]=feedback
```

This is the default setting. By defining TypeList, the default information collection type, form, will be overridden for some classes. The class poll is matched to the information collection type poll, the class form to the information collection type form and the class feedback to the information collection type feedback.

Type

Summary

Sets the default information collection type.

Usage

```
Type=<collection type>
```

Description

This setting sets the default information collection type to use. An override can be set with the setting TypeList in the block [InfoSettings].
By default the following values are possible:

- poll

- feedback

- form: the default value

All values have their own template located here:
<eZ Publish installation>\templates\content\collectedinfo
This setting is case sensitive.

Examples

```
Type=form
```

This is the default setting.

5

5.14.6   config.php

This page contains a documented list of the few configuration points available from *config.php*. The proposed default values in the following sub-chapters are meant to be the most optimized ones, or meaningful examples of what can be achieved.

In order to have the present settings taken into account, you will need to rename this file into *config.php*, and keep it placed at the root of eZ Publish, where *config.php-RECOMMEND* is currently placed. Plus, you will need to *uncomment* the proposed settings.

Please, refer to the information available in the sub-chapters to correctly set up your *config.php* file.

The configuration blocks are documented in the following sections:

\begin{itemize}   \item{}   [ConfigSettings]\refer{labelfd3f7cbe00273919aedaf40715cc1c3e} \end{itemize}

5

[ConfigSettings]

5

PATH_TO_EZCOMPONENTS

Summary

Path to eZ Components

Description

config.php can set the components path like:

```
ini_set( 'include_path', ini_get( 'include_path' ). PATH_SEPARATOR . '../
ezcomponents/trunk' )
```

5

USING_BUNDLED_COMPONENTS

Summary

Using Bundled Components

Description

If you are using a distribution of eZ Publish with which the necessary eZ Components are bundled (in lib/ezc), then you can use this setting to control if the bundled eZ Components should be used (true) or not (false).

By default, when this setting is not present and the bundled eZ Components are present, they will be used. If you're using the bundled eZ Components it's recommended to define EZP_USE_BUNDLED_COMPONENTS as a boolean true anyway, for optimal speed.

```
define( 'EZP_USE_BUNDLED_COMPONENTS', true );
```

If you are not using the bundled eZ Components, then for optimal speed it is recommended to set EZC_BASE_PATH to either ezc/Base/base.php or Base/src/base.php, depending on how you installed the eZ Components.By default, if this setting is not present, eZ Publish first tries to include ezc/Base/base.php in the standard php include path. If that fails Base/src/base.php is included.

```
define( 'EZC_BASE_PATH', '/usr/lib/ezc/Base/base.php' );
```

5

TIMEZONES

Summary

Timezones

Description

The recommended way of setting timezones is via *php.ini*. However, you can set the default timezone for your eZ Publish application as shown below. More on supported timezones : http://www.php.net/manual/en/timezones.php.

```
date_default_timezone_set( 'Europe/Paris' )
```

5

INI_FILE_OPTIMIZATIONS

Summary

INI File Optimizations

Description

This new setting controls whether eZINI should check if the settings have been updated in the *.ini-files*. If this check is disabled eZ Publish will always use the cached values, lowering the amount of stat-calls to the file system, and thus increasing performance.Set EZP_INI_FILEMTIME_ CHECK constant to 'false 'to improve performance by not checking modified time on 'ini files'. You can also set it to the name of an 'ini file' you still want to check modified time. For example, set it to 'site.ini' to make the system still check if that file has been modified, while ignoring changes for any other file.Note that independently of this setting, new INI override files will NEVER be found automatically. You will need to clear at least the INI cache when a new INI file override is added to the structure. Note that new files are not affected, but this is an edge situation.

```
define( 'EZP_INI_FILEMTIME_CHECK', false )
```

5

KERNEL_OVERRIDES

Summary

Kernel Overrides

Description

This setting controls of eZ Publish's autoload system should allow, and load classes, which override kernel classes from extensions.** **

Note 1: Not all classes can be overridden because of existing includes, for e.g., handlers.** **

Note 2: This feature is not covered by support and any kind of kernel overrides needs to be explicitly informed to eZ Systems on support requests.

```
define( 'EZP_AUTOLOAD_ALLOW_KERNEL_OVERRIDE', false )
```

Set EZP_INI_FILE_PERMISSION constant to the permissions you want saved ini and cache files to have. This setting also applies to generated autoload files. Do keep an octal number as value here (meaning it starts with a 0).

```
define( 'EZP_INI_FILE_PERMISSION', 0666 )
```

5

CUSTOM_AUTOLOAD_MECHANISM

Summary

Custom Autoload Mechanism

Description

It is also possible to push a custom autoload method to the autoload function stack. Remember to check for class prefixes in such a method, if it will not serve classes from eZ Publish and eZ Components.Here is an example code snippet to be placed right in this file, in the case you would be using your custom Foo framework, in which all PHP classes would be prefixed by 'Foo':

```
ini_set( 'include_path', ini_get( 'include_path' ). ':/usr/lib/FooFramework/
' );
require 'Foo/ClassLoader.php';

function fooAutoload( $className )
{
    if ( 'Foo' == substr( $className, 0, 3 ) )
    {
        FooClassLoader::loadClass( $className );
    }
}

spl_autoload_register( 'fooAutoload' );
```

5

5.14.7 content.ini

The configuration blocks are documented in the following sections:

\begin{itemize} \item{} [ActionSettings]\refer{labelc51f5a735e536f95083a0f0d7d9ece79}
\item{} [ContentOverrideSettings]\refer{label20dd066a66c4073c7b9a9ca2d53de330}
\item{} [DataTypeSettings]\refer{labelf3813a6a1732f4501889ccf3ae59fdbc}
\item{} [EditSettings]\refer{label7aa324938b4b0bfa92b2fa00adb8bc13}
\item{} [embed-inline]\refer{label611ca477b042711d2a0c64310f55c79e}
\item{} [embed]\refer{label2a205d7334a9885712b29f984ecef514} \item{}
[HideSettings]\refer{labelec59413986c7d1307603535a986c7e0d} \item{}
[ISBNSettings]\refer{labeladd4ead00cde170933dd136516cb3699} \item{}
[link]\refer{label10eba28de723d2edf78fe07e19505709} \item{} [name_
of_XML_tag]\refer{label8b1159dcbfddb7d07109b6145adbc79b} \item{}
[NodeSettings]\refer{labelb6eb21734d270523bd16a056c7d88202} \item{}
[RelationAssignmentSettings]\refer{labelac5d79391bbf43d5008cd3ef5c457c6d}
\item{} [RemoveSettings]\refer{labelaa65a670d8e5d77242550e3c34986fe0}
\item{} [TestingSettings]\refer{labelb4e4481f2bea2a2b04bcc4b41c3ea8a8} \item{}
[UnpublishSettings]\refer{label9270c4808992006c692504c62687eec4} \item{}
[VersionManagement]\refer{label3c477f07545714a19ee18ed60a544431} \item{}
[VersionView]\refer{label22db16db6a2afb2835cc45af0c7edc80} \end{itemize}

5

[ActionSettings]

ExtensionDirectories (page 1723)
Sets the directories where eZ Publish will look for action extensions.

5

ExtensionDirectories

Summary

Sets the directories where eZ Publish will look for action extensions.

Usage

```
ExtensionDirectories[]= directory1
ExtensionDirectories[]= directory2
...
```

Description

eZ Publish will look for actions in " ExtensionDirectory (page 2380)/ *directory*/actions/ content_actionhandler.php". In that file eZ Publish will look for the function *directory*_ ContentActionHandler. This function will be invoked when your action is triggered.
The function must accept three parameters: "module", "http" and "objectID".

5

[ContentOverrideSettings]

EnableClassGroupOverride   (page 1725)

    Sets whether it should be possible to override templates based on the "class_group" condition.

5

EnableClassGroupOverride

Summary

Sets whether it should be possible to override templates based on the "class group" condition.

Usage

```
EnableClassGroupOverride=true|false
```

Description

This directive sets whether it should be possible to override templates based on the "class group" condition. (Refer to the "Template override conditions (page 1444)" section of the "Reference" chapter for information about which templates that can be overriden based on this condition.) The default value is "false", which means that the €class group€ condition will not work.

5

[DataTypeSettings]

AvailableDatatypes   (page 1727)
> Sets the datatypes available to eZ Publish.

ExtensionDirectories   (page 1728)
> Sets the extensions that contains datatypes.

RepositoryDirectories   (page 1729)
> Sets the directories where eZ Publish should look for datatypes.

5

AvailableDatatypes

Summary

Sets the datatypes available to eZ Publish.

Usage

```
AvailableDatatypes[]= datatype1
AvailableDatatypes[]= datatype2
...
```

Description

eZ Publish will look for the datatypes in the directories specified by RepositoryDirectories (page 1729) and ExtensionDirectories (page 1728).

Examples

```
ExtensionDirectory[]=my_extension
AvailableDatatypes[]=my_datatype
```

This example shows a typical configuration of content.ini.append in an extension with a datatype. If extensions are located in the "extension" directory (default) these settings will make eZ Publish look for the datatype extension in the directory "/extension/my_extension/datatypes/my_datatype/".

5

ExtensionDirectories

Summary

Sets the extensions that contains datatypes.

Usage

```
ExtensionDirectories[]= directory1
ExtensionDirectories[]= directory2
...
```

Description

eZ Publish will look for datatypes in " ExtensionDirectory (page 2380)/ *directory*/datatypes/".
You must also add your datatype to AvailableDataTypes (page 1727) in order to make eZ Publish
recognize it.

5

RepositoryDirectories

Summary

Sets the directories where eZ Publish should look for datatypes.

Usage

```
RepositoryDirectories[]= directory1
RepositoryDirectories[]= directory2
```

Description

Directories should be specified relative to your eZ Publish root directory.

This setting is for kernel datatypes only. If you add your own datatype you should put it in an extension and use the ExtensionDirectories (page 1728) setting to tell eZ Publish where to find it.

5

[EditSettings]

\begin{description} \item[AdditionalTemplates]\refer{labeld2ae1d0cc08bf771368c67b5b20b990e}\
\\ A list of templates to include in the content/edit form.

\end{description}

5

AdditionalTemplates

Summary

A list of templates to include in the content/edit form.

Usage

```
AdditionalTemplates[]=content/edit/autosave.tpl
```

Description

This setting will include the template "design:content/edit/autosave.tpl" from "design:content/edit.tpl".

Note: This is relevant to the 4.7 version, and later versions, of eZ Publish.

Examples

Example: "content.ini.append.php"

```
[EditSettings]
AdditionalTemplates[]=content/edit/autosave.tpl
```

5

[embed-inline]

AvailableViewModes

Summary

sets the view parameter options users can choose in the Online Editor

Usage

```
AvailableViewModes[]=
```

Description

This setting is new and available since eZ Publish 4.1.

This setting allows you to define the view parameter options that users can choose from in the Online Editor 5.x. This setting is also used in [link] and [embed] in the content.ini file.

5

[embed]

AvailableViewModes

Summary

sets the view parameter options users can choose in the Online Editor

Usage

```
AvailableViewModes[]=
```

Description

This setting is new and available since eZ Publish 4.1.

This setting allows you to define the view parameter options that users can choose from in the Online Editor 5.x. This setting is also used in [link] and [embed-inline] in the content.ini file.

5

[HideSettings]

HideDateAttributeList   (page 1737)
  Sets which date/time attributes the hide cronjob should use.

RootNodeList   (page 1738)
  Sets which subtrees that will be affected by the "hide.php" cronjob.

5

HideDateAttributeList

Summary

Sets which date/time attributes the hide cronjob should use.

Usage

```
HideDateAttributeList[class_id1]=attribute_id1
HideDateAttributeList[class_id2]=attribute_id2
...
```

Description

This directive can be used to tell the "hide.php" cronjob about which date/time values it should look for when hiding nodes. Both the class and the attribute (which needs to be represented by the "Date and time (page 524)" datatype) must be specified using identifiers (not ID numbers). In addition, the "RootNodeList (page 1738)" directive must be used to specify which subtrees that can have their nodes affected by the cronjob script.

Examples

Example 1

```
HideDateAttributeList[article]=hide_date
```

The hide cronjob will check date/time values specified in the "hide_date" attribute of your articles.

5

RootNodeList

Summary

Sets which subtrees that will be affected by the "hide.php" cronjob.

Usage

```
RootNodeList[]=node_id1
RootNodeList[]=node_id2
...
```

Description

If you are using the "hide.php" cronjob to control the visibility of your nodes (based on a date/ time-attribute), you must use this directive to specify which subtrees that can have their children (recursively) affected by this feature. Please note that the default value is empty, which means that no nodes will be hidden.

Examples

Example 1

```
RootNodeList[]=2
```

This tells the system that the "hide.php" cronjob can affect all items in the content node tree.

5

[ISBNSettings]

StrictValidation   (page 1740)
   Enables or disables the strict validation of the ISBN-13 numbers.

5

StrictValidation

Summary

Enables or disables the strict validation of the ISBN-13 numbers.

Usage

```
StrictValidation=true|false
```

Description

This directive makes it possible to enable or disable strict validation of ISBN-13 numbers. All ISBN-13 numbers consist of five elements called "Prefix", "Registration group", "Registrant", "Publication" and "Check digit"; they are separated by dashes (refer to the documentation of the "ISBN (page 543)" datatype for more information).

The default value of this setting is "true", which means that strict validation is enforced. All elements of each ISBN-13 number will be validated according to the rules stored in three range tables of the database. Note that the rules may be outdated. It is possible to add new areas (regions) to the list of group identifiers. You can update the ranges stored in the database by running the "updateisbn13.php" script. If an incorrect ISBN-13 number is entered, it will not be accepted and a warning will be shown. For example, currently the system does not accept "978-650-123-456-4" because the area number "650" doesn't exist in the corresponding range table.

If set to "false", only the "Prefix" and "Check digit" elements will be validated. In other words, the system will accept input regardless if the "Registration group", "Registrant" and "Publication" elements are correct or not. This is typically useful if you need to enter ISBN-13 numbers containing values that aren't present in the range tables at the moment.

Note that the system will automatically add missing dashes and/or correct misplaced ones. However, this will only happen if the range exists in the database. If it doesn't, you will need to separate the ISBN-13 elements by entering the dashes manually.

5

[link]

AvailableViewModes

Summary

sets the view parameter options users can choose in the Online Editor

Usage

```
AvailableViewModes[]=
```

Description

This setting is new and available since eZ Publish 4.1.

This setting allows you to define the view parameter options that users can choose from in the Online Editor 5.x. This setting is also used in [embed] and [embed-inline] in the content.ini file.

5

[name_of_XML_tag]

AvailableClasses   (page 1745)
    Sets the available classes for XML tags of the "name_of_XML_tag" type.

CustomAttributes   (page 1744)
    Sets which custom attributes that can be used within XML tags of this particular type.

5

CustomAttributes

Summary

Sets which custom attributes that can be used within XML tags of this particular type.

Usage

```
CustomAttributes[]=attribute1
CustomAttributes[]=attribute2
...
```

Description

This setting makes it possible to specify which custom attributes that can be used within XML tags of the "name_of_XML_tag" type. (From 3.9, custom attributes are supported for all XML tags.) It is an array of custom attribute names. If empty, the system will not allow the use of custom attributes within XML tags of the "name_of_XML_tag" type.

Examples

Example 1

If you wish to use a specific summary field in your tables, you can add a custom attribute called "summary" to the "table" XML tag by adding the following lines to the [table] section of your "content.ini.append.php" file:

```
CustomAttributes[]
CustomAttributes[]=summary
```

Now, if you create a table using an XML code like this:

```
<table summary="The meaning of my favourite smiles">
...
</table>
```

and put the following code into an override template for the "content/datatype/view/ezxmltags/table.tpl" template:

```
summary: {$summary}
```

the following output will be produced: "summary: The meaning of my favourite smiles".

5

AvailableClasses

Summary

Sets the available classes for XML tags of the "name_of_XML_tag" type.

Usage

```
AvailableClasses[]=class1
AvailableClasses[]=class2
...
```

Description

This setting makes it possible to specify which custom classes that can be used within XML tags of the "name_of_XML_tag" type. It is an array of values that can be assigned to the "class" attribute of the XML tags. If empty, the system will not allow the use of classified "name_of_XML_tag" tags. Note that the classification of anchor tags and custom tags is not supported.

Examples

Example 1

If you wish to use "pRed" and "pBlue" classified paragraphs in your XML blocks, make sure that the [paragraph] section in "content.ini.append.php" contains the following lines:

```
AvailableClasses[]
AvailableClasses[]=pRed
AvailableClasses[]=pBlue
```

5

[NodeSettings]

\begin{description} \item[DesignRootNode]\refer{label6b1d5d0a38626e2a116430fb117f6206}\ \\ The node ID for the design tree

\item[MediaRootNode]\refer{label7536eacd2fb965808c30d32b2fb7bc4d}\ \\ The node ID for the media tree

\item[RootNode]\refer{labelc51db921c139edb1bde7b6452c589c6b}\ \\ The node ID of the normal content tree

\item[SetupRootNode]\refer{labelb0a18a23646d9de5c8492cfd660a5e70}\ \\ The node ID for the setup tree

\item[UserRootNode]\refer{label93a75510e1df052f86b8b8b562d1143a}\ \\ The node ID of the user tree

\end{description}

5

RootNode

Summary

The node ID of the normal content tree

Usage

```
RootNode=2
```

Description

This setting sets the node ID of the normal content tree.

Note

It is recommended that you never change any of the settings in the [NodeSettings] group, because this could make the site unstable.

Examples

```
RootNode=2
```

5

UserRootNode

Summary

The node ID of the user tree

Usage

```
UserRootNode=5
```

Description

This setting sets the node ID of the user tree.

Note

It is recommended that you never change any of the settings in the [NodeSettings] group, because this could make the site unstable.

Examples

```
UserRootNode=5
```

5

MediaRootNode

Summary

The node ID for the media tree

Usage

```
MediaRootNode=43
```

Description

This setting sets the node ID for the media tree.

Note

It is recommended that you never change any of the settings in the [NodeSettings] group, because this could make the site unstable.

Examples

```
MediaRootNode=43
```

5

SetupRootNode

Summary

The node ID for the setup tree

Usage

```
SetupRootNode=48
```

Description

This setting sets the node ID for the setup tree.

Note

It is recommended that you never change any of the settings in the [NodeSettings] group, because this could make the site unstable.

Examples

```
SetupRootNode=48
```

5

DesignRootNode

Summary

The node ID for the design tree

Usage

```
DesignRootNode=58
```

Description

This setting sets the node ID for the design tree.

Note

It is recommended that you never change any of the settings in the [NodeSettings] group, because this could make the site unstable.

Examples

```
DesignRootNode=58
```

5

[RelationAssignmentSettings]

ClassSpecificAssignment   (page 1753)
    Sets the rules for automatic location assignment when uploading related objects.

DefaultAssignment   (page 1755)
    Sets where the newly created related objects should be placed in the node tree.

5

ClassSpecificAssignment

Summary

Sets the rules for automatic location assignment when uploading related objects.

Usage

```
ClassSpecificAssignment[]= list_of_classes_and_class_groups1;list_of_
locations1;main_location1
ClassSpecificAssignment[]= list_of_classes_and_class_groups2;list_of_
locations2;main_location2
...
```

Description

While the "DefaultAssignment (page 1755)" directive sets the default location for newly created (uploaded) related objects, the "ClassSpecificAssignment" directive makes it possible to specify a set of override rules. These rules make it possible to place specific types of objects at desired locations in the tree.

The value of this setting has to be provided as an array where each element must contain two or three values separated by semicolons.

The first value sets which types of objects that should be affected. It has to be a class and/or class group or a comma separated list of classes and/or class groups (both identifiers and ID numbers can be used). Note that classes should come before class groups because it is always the first match that will be used.

The second value tells the system where to place newly uploaded objects of the specified type. It has to be a location or a comma separated list of locations (both paths and ID numbers of the desired nodes can be used). If multiple locations are specified, you can use the optional third parameter to set which of the locations that should be the main node. If the third (main node) parameter is omitted, the first location in the list will be the main node.

From 3.9, the "ClassSpecificAssignment" array may contain several entries using the same first value. This means that you can specify multiple sets of locations for the same class. The system will use the one (the others will be skipped) under which the current user has sufficient "create" privileges.

If the class is not mentioned anywhere in the "ClassSpecificAssignment" setting or if it is mentioned but the current user does not have sufficient "create" privileges at the specified locations, the value (location) specified by the "DefaultAssignment" setting will be used.

5

Examples

Example 1

```
ClassSpecificAssignment[]=flash;media/movies,media/multimedia,media/myflash;
media/multimedia
```

With this configuration, newly uploaded flash files will be placed at three locations: "media/ movies", "media/myflash" and "media/multimedia". The main node will be located under "media/multimedia".

Example 2

Let's say that you have several folders called "computers", "news", "company" and "john_photo" located under the "Media" top level node. In addition, you create the following rules:

```
ClassSpecificAssignment[]=image;media/computers,media/news
ClassSpecificAssignment[]=image;media/company
ClassSpecificAssignment[]=image;media/john_photo
```

In this case, the system will choose a location for uploaded images according to the current user's privileges. If the user has sufficient "create" privileges under both "media/computers" and "media/news", all images uploaded by this user will be put to these folders (the system will create main nodes under "media/computers" and additional locations under "media/news"). However, if the user does not have the necessary privileges, the system will skip this line and go to the next one. If the user has sufficient "create" privileges under "media/company", the images will be put inside that folder. If not, the system will skip that line as well and check the next one. If the user has sufficient "create" privileges under "media/john_photo", the images will be put in that folder. Otherwise, the default location (page) will be used.

5

DefaultAssignment

Summary

Sets where the newly created related objects should be placed in the node tree.

Usage

```
DefaultAssignment= location
```

Description

When you add related objects using the "Upload new" button located in the "Related objects" window in the object edit interface, the "Location" dropdown in the upload dialog allows you to choose where the uploaded file should be placed in the node tree. If "Automatic" is selected, then the location specified by the "DefaultAssignment" setting will be used.

The default value of this setting is "root", which means that related objects will be placed directly under the "Content" top level node. The following options can be also used:

- The ID number or the desired parent node (for example "2").

- The path to the desired parent node in the content tree (for example "users", "media" or "media/files").

Note that this setting alone doesn't allow you to specify multiple locations for related objects that are being uploaded. Multiple locations can however be specified using the "ClassSpecificAssignment" (page 1753) directive; making it possible to set up rules for different types of objects that override the value of the "DefaultAssignment" directive. If the current user doesn't have sufficient "create" privileges under the locations specified using the "ClassSpecificAssignment" , the system will fallback to and use the value of the "DefaultAssignment" directive.

Examples

Example 1

```
DefaultAssignment=media
```

Newly created (uploaded) related objects will (by default) be placed under the "Media" top level node. This will be the case unless specified otherwise by the "ClassSpecificAssignment" directive or if the current user doesn't have sufficient "create" privileges at the locations specified by the "ClassSpecificAssignment" directive.

5

[RemoveSettings]

DefaultRemoveAction   (page 1759)
    Sets the default remove action for objects (move to trash or delete).

HideRemoveConfirmation   (page 1757)
    Sets whether the removal confirmation dialog should be shown or not.

MaxNodesRemoveSubtree   (page 1760)
    Sets the maximum number of nodes that subtrees subject to removal can contain.

ShowRemoveToTrashCheck   (page 1758)
    Sets whether the "Move to trash" checkbox is displayed or not.

5

HideRemoveConfirmation

Summary

Sets whether the removal confirmation dialog should be shown or not.

Usage

```
HideRemoveConfirmation=true|false
```

Description

This setting controls wether the system should ask for confirmation when objects are being re-moved. The default value is "false", which means that the removal confirmation dialog will be shown. If set to "true", the system will not display the dialog. When the dialog is not shown (or if the "Move to trash" checkbox is hidden), it is the DefaultRemoveAction (page 1759)" directive that controls whether objects should be moved to the trash or permanently deleted.

Examples

Example 1

```
HideRemoveConfirmation=true
```

With this configuration, the system will not ask for confirmation when objects are being removed. If you specify "DefaultRemoveAction=delete", the system will permanently delete objects (they will not be put into the trash).

5

ShowRemoveToTrashCheck

Summary

Sets whether the "Move to trash" checkbox is displayed or not.

Usage

```
ShowRemoveToTrashCheck=true|false
```

Description

This setting controls the visibility of the "Move to trash" checkbox in the removal confirmation dialog. The default value of this setting is "true", which means that the "Move to trash" checkbox will be displayed.

The removal confirmation dialog is shown when objects are being removed and only if the dialog is enabled (default behavior). When the dialog is not shown (or if the "Move to trash" checkbox is hidden), it is the DefaultRemoveAction (page 1759)" directive that controls whether objects should be moved to the trash or permanently deleted.

Examples

Example 1

```
HideRemoveConfirmation=false
ShowRemoveToTrashCheck=false
```

With this configuration, the system will always ask for confirmation when objects are being removed by users. However, there will be no "Move to trash" checkbox in the removal confirmation dialog. (You can use the "DefaultRemoveAction" directive to control whether objects should be moved to trash or permanently deleted.)

5

DefaultRemoveAction

Summary

Sets the default remove action for objects (move to trash or delete).

Usage

```
DefaultRemoveAction=trash|delete
```

Description

This directive controls the default behavior of the system when objects are being removed. If a user doesn't specify whether an object should be removed completely or moved to trash (either because the dialog is not being shown or the "Move to trash" checkbox is hidden), the system will behave according to the value of "DefaultRemoveAction".

The default value of this setting is "trash", which means that removed objects will be moved to trash instead of being deleted. If the directive is set to "delete", the system will not move removed objects to the trash (they will be deleted/lost forever).

If the dialog and the checkbox is being shown, the users are allowed to override the default setting and thus control whether objects should be deleted or moved to trash.

5

MaxNodesRemoveSubtree

Summary

Sets the maximum number of nodes that subtrees subject to removal can contain.

Usage

```
MaxNodesRemoveSubtree= number
```

Description

This directive makes it possible to disallow the removal of big subtrees (for all users). When using the administration interface and trying to remove subtrees containing more nodes than "MaxNodesRemoveSubtree" allows, a warning will be shown while the subtrees will be displayed in red. In such cases, the user will be suggested to use the "ezsubtreeremove.php" script located in the "bin/php/" directory. Subtrees containing more nodes than the value of "MaxNodesRemoveSubtree" can and will not be removed through HTTP (using a web browser to access a siteaccess).

If several subtrees are selected for removal, only the ones containing less nodes than allowed will be removed when the "OK" button is clicked within the removal dialog of the administration interface. The default value of this setting is 100, which means that subtrees containing more than 100 nodes can't be removed through HTTP, not even by users with administrator privileges.

This setting has been introduced in order to avoid cases which might lead to corrupt databases. The time it takes to remove a subtree is proportional with the number of nodes that it contains. If the required processing time exceeds the maximum execution time for PHP scripts, the connection will be lost and thus the database will most likely be left in an inconsistent state.

Examples

Example 1

```
MaxNodesRemoveSubtree=50
```

This configuration will not allow the removal of subtrees that contain more than 50 nodes over HTTP. Subtrees containing more than 50 nodes will have to be removed manually using the "ezsubtreeremove.php" script.

5

[TestingSettings]

Settings related to multivariate testing feature added in 4.7.

\begin{description} \item[EnabledForSiteAccessList]\refer{labelea2d69e374b54ac7c441b622600e186a}\ \\ List of site accesses multivariate testing is enabled for

\item[MultivariateTesting]\refer{label79b1f8b08f6c9530e904735fbeab4e2c}\ \\ Enable or disable multivariate testing feature

\item[MultivariateTestingHandlerClass]\refer{label05028b9cc13469b27f30f56d72b6622e}\ \\ A handler class which implements $<$ezpMultivariateTestHandlerInterface$>$ interface

\end{description}

5

MultivariateTesting

Summary

Enable or disable multivariate testing feature

Usage

```
Master switch to enable / disable multivariate testing feature.
```

Examples

```
[TestingSettings]
# Enable or disable multivariate testing feature
MultivariateTesting=enabled
```

5

MultivariateTestingHandlerClass

Summary

A handler class which implements <ezpMultivariateTestHandlerInterface> interface

Description

Optionally you can define your custom handler class (more information how to create your own handler class in the €Create custom handler€ section of ezmultivariatetest doc bundled in the extension). This step can be skipped, then default implementation will be used.

Examples

```
# A handler class which implements <ezpMultivariateTestHandlerInterface>
interface
MultivariateTestingHandlerClass=ezpMultivariateTestHandler
```

5

EnabledForSiteAccessList

Summary

List of site accesses multivariate testing is enabled for

Description

Define list (array) of public siteaccesses multivariate testing feature will be enable for.

Examples

```
# List of site accesses multivariate testing is enabled for.
EnabledForSiteAccessList[]Creating test scenarios
```

5

[UnpublishSettings]

ClassList   (page 1766)
  Sets the content classes that use the "Date and time" datatype (used by the "unpublish.php" cronjob).

RootNodeList   (page 1767)
  Controls which subtrees that will be affected by the "unpublish.php" cronjob.

5

ClassList

Summary

Sets the content classes that use the "Date and time" datatype (used by the "unpublish.php" cronjob).

Usage

```
ClassList[]=class_id1
ClassList[]=class_id2
...
```

Description

This setting reveals the content classes that contain attributes which can be used by the "unpublish.php" script. Please note that you'll have to use ID numbers here, not identifiers. Only objects that are of these classes will be affected by the script. In addition, the "RootNodeList (page 1767)" directive must be used to specify which subtrees that can have their nodes affected by the cronjob script.

Examples

Example 1

```
ClassList[]=2
```

Assuming that the ID number of your article class is 2, this configuration will tell the "unpublish.php" cronjob to check the date and time values specified in the "unpublish_date" attribute of the article(s).

5

RootNodeList

Summary

Controls which subtrees that will be affected by the "unpublish.php" cronjob.

Usage

```
RootNodeList[]=node_id1
RootNodeList[]=node_id2
...
```

Description

If you are using the "unpublish.php" cronjob to delete (move to Trash) your content objects when a specified date and time is reached, you can use this directive to specify which parent nodes that will have their children (recursively) affected by the unpublish feature. The "RootNodeList" configuration array is empty by default, i.e. no objects will be removed.

Examples

Example 1

```
RootNodeList[]=2
```

This will tell the system that the "unpublish.php" cronjob can be applied to any item in the content tree.

5

[VersionManagement]

Read more (page 111) about the versioning system in eZ Publish. \begin{description}
\item[DefaultVersionHistoryLimit]\refer{labele76f1671885e056db753b432a682d420}\ \\ Sets the number of concurrent versions that can exist of an object.

\item[DeleteDrafts]\refer{label659f28e407eb0acf7c96bde8f58b094b}\ \\ Sets if drafts should be discarded if there is no room to create a new draft.

\item[DraftsCleanUpLimit]\refer{labelf5b2fad733c1811a69fb67d3b6b30580}\ \\ Sets the maximum number of drafts to remove at one call of the "old_drafts_cleanup.php" cronjob script.

\item[DraftsDuration]\refer{label21d4f90e9c737f8ad09ab1025e73d5fd}\ \\ Sets how long a draft can exist in the system before it is considered old and can be removed.

\item[InternalDraftsCleanUpLimit]\refer{label66b762c08a1756443418c47bf1df3d66}\ \\ Sets the maximum number of internal drafts to remove at one call of the "internal_drafts_cleanup.php" cronjob script.

\item[InternalDraftsDuration]\refer{labele2b0499a64ed42c4f93b8a6e0d557307}\ \\ Sets how long an internal draft can exist in the system before it is considered old and can be removed.

\item[VersionHistoryClass]\refer{label8092f0c3ec526319940be8ad218caf5d}\ \\ Sets the number of concurrent versions that can exist of an object per class.

\end{description}

5

InternalDraftsCleanUpLimit

Summary

Sets the maximum number of internal drafts to remove at one call of the "internal_drafts_cleanup.php" cronjob script.

Usage

```
InternalDraftsCleanUpLimit=number
```

Description

This setting allows to limit the number of internal drafts that can be removed at one call of the "internal_drafts_cleanup.php" cronjob script.

Examples

Example 1

```
InternalDraftsCleanUpLimit=100
```

This will tell the system that the "internal_drafts_cleanup.php" cronjob script should not remove more than 100 drafts at a time.

5

DraftsCleanUpLimit

Summary

Sets the maximum number of drafts to remove at one call of the "old_drafts_cleanup.php" cronjob script.

Usage

```
DraftsCleanUpLimit=number
```

Description

This setting allows to limit the number of drafts that can be removed at one call of the "old_drafts_cleanup.php" cronjob script.

Examples

Example 1

```
DraftsCleanUpLimit=100
```

This will tell the system that the "old_drafts_cleanup.php" cronjob script should not remove more than 100 drafts at a time.

5

InternalDraftsDuration

Summary

Sets how long an internal draft can exist in the system before it is considered old and can be removed.

Usage

```
InternalDraftsDuration[days]=number_of_days
InternalDraftsDuration[hours]=number_of_hours
InternalDraftsDuration[minutes]=number_of_minutes
InternalDraftsDuration[seconds]=number_of_seconds
```

Description

Use this setting to specify the number of days, hours, minutes and seconds before an internal draft is considered old and can be removed. Note that the "internal_drafts_cleanup.php" cronjob script must be run periodically for this setting to have any effect.

Examples

Example 1

```
InternalDraftsDuration[]
InternalDraftsDuration[days]=0
InternalDraftsDuration[hours]=20
InternalDraftsDuration[minutes]=0
InternalDraftsDuration[seconds]=0
```

With this configuration, the "internal_drafts_cleanup.php" cronjob script will remove untouched drafts that have been in the system for over 20 hours.

5

DraftsDuration

Summary

Sets how long a draft can exist in the system before it is considered old and can be removed.

Usage

```
DraftsDuration[days]=number_of_days
DraftsDuration[hours]=number_of_hours
DraftsDuration[minutes]=number_of_minutes
DraftsDuration[seconds]=number_of_seconds
```

Description

Use this setting to specify the number of days, hours, minutes and seconds before a draft is considered old and can be removed. Note that the "old_drafts_cleanup.php" cronjob script must be run periodically for this setting to have any effect.

Examples

Example 1

```
DraftsDuration[]
DraftsDuration[days]=30
DraftsDuration[hours]=0
DraftsDuration[minutes]=0
DraftsDuration[seconds]=0
```

With this configuration, the "old_drafts_cleanup.php" cronjob script will remove drafts that have been in the system for over 30 days.

5

DeleteDrafts

Summary

Sets if drafts should be discarded if there is no room to create a new draft.

Usage

```
DeleteDrafts=enabled|disabled
```

Description

This settings controls what eZ Publish should do if a user wants to create a new draft but there are no "empty" versions.

- **disabled** - (default) The oldest archived version is discarded. If there are no archived versions to discard the user is shown an error screen and is given the possibility to remove drafts.

- **enabled** - The oldest archived version is discarded. If there are no archived versions to discard eZ Publish discards the oldest draft.

5

VersionHistoryClass

Summary

Sets the number of concurrent versions that can exist of an object per class.

Usage

```
VersionHistoryClass[ class_id1]= number
VersionHistoryClass[ class_id2]= number
...
```

Description

This setting is similar to DefaultVersionHistoryLimit (page 1775) but allows you to set the maximum number of versions per class giving you more fine grained control.

A typical scenario is to limit the number of versions of posts in your public forum, but not for the other classes in the system.

If the limit set here is reached, the oldest archived version will automatically be deleted.

Note: When decreasing "content.ini/[VersionManagement]/VersionHistoryClass" you need to run the script script "*bin/php/cleanupversions.php*" to remove exceeded archived versions.

Examples

```
VersionHistoryClass[1]=5
VersionHistoryClass[3]=3
```

Using these settings eZ Publish will allow 5 versions for objects of classes with id 1 and 3 versions for objects of class 3. Other objects in your system will use the global DefaultVersionHistoryLimit (page 1775) to determine the maximum number of versions.

5

DefaultVersionHistoryLimit

Summary

Sets the number of concurrent versions that can exist of an object.

Usage

```
DefaultVersionHistoryLimit= number
```

Description

This setting globally determines how many versions eZ Publish will store of a specific object including drafts.
You should never set this setting below 2 since you need to allow at least one published version and one draft.

If the limit set here is reached, the oldest archived version will automatically be deleted.

Note: When decreasing "content.ini/[VersionManagement]/DefaultVersionHistoryLimit" you need to run the script script "*bin/php/cleanupversions.php*" to remove exceeded archived versions.

Examples

```
DefaultVersionHistoryLimit=10
```

10 is the default value for this setting.

5

[VersionView]

AllowChangeButtons   (page 1779)
    This setting is not used anymore.

AllowVersionsButton   (page 1778)
    This setting is not used anymore.

AvailableSiteDesignList   (page 1781)
    DEPRECATED (Sets the sitedesigns that are used by the complete site.)

DefaultPreviewDesign   (page 1780)
    Sets the default preview design for eZ Publish releases prior to 3.6.

VersionHistoryClass   (page 1777)
    Defines the maximum number that can be managed

5

VersionHistoryClass

Summary

Defines the maximum number that can be managed

Usage

```
VersionHistoryClass[class_identifier]=maximum_nr
```

Description

This setting define the maximum amount of versions that can be managed. In order to specify the number of object versions to keep per class, specified by the class ID, you can use the syntax:

```
VersionHistoryClass[class_id]=maximum_nr
```

Since eZ Publish 4.1.1, you can also specify the class by its identifier or class_identifier. The syntax is this:

```
VersionHistoryClass[class_identifier]=maximum_nr
```

If no value is specified, the value of DefaultVersionHistoryLimit will be used.

Examples

Example 1

```
VersionHistoryClass[folder]=1
```

For the folder class only one version will be kept.

5

AllowVersionsButton

Summary

This setting is not used anymore.

Usage

```
AllowVersionsButton=enabled|disabled
```

5

AllowChangeButtons

Summary

This setting is not used anymore.

Usage

```
AllowChangeButtons=enabled|disabled
```

5

DefaultPreviewDesign

Summary

Sets the default preview design for eZ Publish releases prior to 3.6.

Usage

```
DefaultPreviewDesign= design_name
```

5

AvailableSiteDesignList

Summary

DEPRECATED (Sets the sitedesigns that are used by the complete site.)

Usage

```
AvailableSiteDesignList[]=design1
AvailableSiteDesignList[]=design2
...
```

Description

Note: This setting is deprecated and no longer used.

This setting defines the complete set of designs (page 154) that is used by your site. The value is used by the caching system to clear the correct caches when your site is updated. In addition the designs listed here will be available when you are previewing content.

eZ Publish will only clear the caches of the listed designs when content is published. To ensure the correct operation of your site it is important that you list all the designs you are using with this setting.

Examples

AvailableSiteDesignList[]=standard
AvailableSiteDesignList[]=base
AvailableSiteDesignList[]=admin

5

5.14.8   contentstructuremenu.ini

The configuration blocks are documented in the following sections:

- [TreeMenu] (page 1783)

5

[TreeMenu]

Dynamic   (page 1784)
        Sets which implementation of the tree menu that should be used.

MaxDepth   (page 1785)
        Sets the maximal depth for the tree menu.

RootNodeID   (page 1786)
        Sets the root node for the tree menu.

ShowClasses   (page 1787)
        Sets which types of nodes to show in the tree menu.

5

Dynamic

Summary

Sets which implementation of the tree menu that should be used.

Usage

```
Dynamic=enabled|disabled
```

Description

This directive can be used to switch between the original and the new implementation of the tree menu. If you have a large site with many nodes, make sure that this switch is enabled for your administration siteaccess; it will boost the performance of the tree menu and eliminate unoptimized usage of network bandwidth by reducing the amount of traffic involved.

The dynamic tree menu has been introduced in eZ Publish 3.10. It makes use of AJAX technology, which is supported by all modern browsers. The new implementation makes it possible to avoid problems with the tree menu on sites that have huge amount of nodes. Unlike the original implementation, instead of fetching information about the whole content structure tree for every page that is requested, the new one generates menu structures on the fly. The amount of data transferred between the client and the server is minimized. Since the tree menu structure is cached in the browser, only necessary information is transferred through the network.

Refer to the specifications for more information about this feature.

5

MaxDepth

Summary

Sets the maximal depth for the tree menu.

Usage

```
MaxDepth=number
```

Description

Sets the maximum level of depth that should be explored. The default value is 0 which means
"unlimited". If a non-zero number is specified, the tree will only go to this depth.

Note that this setting will be ignored when the "Dynamic (page 1784)" switch is enabled.

Examples

Example 1

```
MaxDepth=1
```

This will tell the system to show only the "Content" top level node without subitems.

Example 2

```
MaxDepth=2
```

This will instruct the system to show only the root node and its children without further exploring.

5

RootNodeID

Summary

Sets the root node for the tree menu.

Usage

```
RootNodeID=node_id
```

Description

This setting controls which node to use as the root one for the tree menu. The default value is 2 which means that the "Content structure" menu will display a tree containing the nodes that belong to the "Content" top level node (node ID= 2).

5

ShowClasses

Summary

Sets which types of nodes to show in the tree menu.

Usage

```
ShowClasses[]=class_identifier1
ShowClasses[]=class_identifier2
...
```

Description

The "Content structure" tree in the administration interface only shows certain types of nodes by default. The "ShowClasses" setting determines which types of nodes to show. If a content class is not listed in this array, then a node that encapsulates an object of this class will not be shown in the tree menu.

Examples

Example 1

```
ShowClasses[]
ShowClasses[]=folder
ShowClasses[]=product
ShowClasses[]=article
```

This will tell the system to show folders, products and articles in the tree menu. Any other nodes will not be shown.

Example 2

```
ShowClasses[]
```

This allows the tree menu to show all kinds of nodes.

5.14.9   country.ini

The configuration blocks are documented in the following sections:

\begin{itemize} \item{} [two-letter_country_code]\refer{labelb8eb45bb0c32538ab158ba9f769a0efe} \end{itemize}

5

[two-letter_country_code]

Alpha2   (page 1792)
     Sets the alpha-2 code of the country.

Alpha3   (page 1791)
     Sets the alpha-3 code of the country.

IDC   (page 1790)
     Sets the International dialing code of the country.

Name   (page 1793)
     Sets the name of the country.

5

IDC

Summary

Sets the International dialing code of the country.

Usage

```
IDC=number
```

Description

Use this directive to specify the International dialing code of the country that is represented by
"two-letter_country_code" (name of the configuration block).

5

Alpha3

Summary

Sets the alpha-3 code of the country.

Usage

```
Alpha3=code
```

Description

Use this directive to specify the ISO 3166-1 alpha-3 code of the country that is represented by
"two-letter_country_code" (name of the configuration block).

5

Alpha2

Summary

Sets the alpha-2 code of the country.

Usage

```
Alpha2=code
```

Description

Use this directive to specify the ISO 3166-1 alpha-2 code of the country that is represented by
"two-letter_country_code" (name of the configuration block).

Examples

Example 1

```
[NO]
Name=Norway
Alpha2=NO
Alpha3=NOR
IDC=47
```

With this configuration, the system will know that ISO 3166-1 alpha-2 code for Norway is "NO",
alpha-3 code is "NOR" and International dialing code is +47.

5

Name

Summary

Sets the name of the country.

Usage

```
Name=name_of_country
```

Description

Use this directive to specify the name of the country that is represented by "two-letter_country_code" (name of the configuration block).

5

5.14.10 cronjob.ini

The configuration blocks are documented in the following sections:

- [CronjobPart-group_of_tasks] (page)
- [CronjobPart-unlock] (page)
- [CronjobSettings] (page)
- [linkCheckSettings] (page)

5

[CronjobPart-unlock]

Scripts

Summary

this setting relates to the new object states system

Usage

```
Scripts[]=
```

Description

A new "unlock" cronjob script has been added since eZ Publish 4.1. This relates to the new object states system.

Examples

```
Scripts[]=unlock.php
```

5

[CronjobPart-group_of_tasks]

Scripts   (page 1798)
     Specifies the cronjob scripts that will be executed for this cronjob part.

5

Scripts

Summary

Specifies the cronjob scripts that will be executed for this cronjob part.

Usage

```
Scripts[]=cronjob_script1
Scripts[]=cronjob_script2
...
```

Description

This setting contains a list of cronjob scripts that will be run when the "group_of_tasks" set of cronjobs is executed. Please refer to the "Configuring cronjobs (page 364)" section for more information and examples.

5

[linkCheckSettings]

SiteURL   (page)
Sets the site URLs to use when validating internal links by the linkcheck cronjob.

5

SiteURL

Summary

Sets the site URLs to use when validating internal links by the linkcheck cronjob.

Usage

```
SiteURL[]=URL1
SiteURL[]=URL2
...
```

Description

Specify your site URLs so that the "linkcheck.php" cronjob will handle relative URLs (internal links) properly.

Examples

```
SiteURL[]=http://admin.mysite.com
SiteURL[]=http://mysite.com
```

This example will tell the linkcheck script to use "http://admin.mysite.com" and "http://mysite.com" site URLs when checking relative URLs like "/products/black_box".

5

[CronjobSettings]

ExtensionDirectories   (page 1802)

Sets one or multiple extension directories that may contain cronjobs.

ScriptDirectories   (page 1804)

Sets one or multiple directories that may contain cronjobs.

Scripts   (page 1803)

Specifies the cronjob scripts that will be executed.

5

ExtensionDirectories

Summary

Sets one or multiple extension directories that may contain cronjobs.

Usage

```
ExtensionDirectories[]= extension1
ExtensionDirectories[]= extension2
...
```

Description

The cronjob script will search through all the "cronjobs" directories of the specified extensions.

Examples

```
Scripts[]=workflow.php
Scripts[]=notification.php
Scripts[]=linkcheck.php
ExtensionDirectories[]=myExtension
```

This example will search through the myExtension/cronjobs directories to find the workflow, notification, and linkcheck cronjobs.

5

Scripts

Summary

Specifies the cronjob scripts that will be executed.

Usage

```
Scripts[]= cronjob_script
Scripts[]= cronjob_script2
...
```

Examples

```
ScriptDirectories[]=cronjobs
ScriptDirectories[]=cronjobs2
Scripts[]=workflow.php
Scripts[]=notification.php
Scripts[]=basket_cleanup.php
Scripts[]=linkcheck.php
```

This example will search through the cronjobs and cronjobs2 directories to find the workflow, notification, basket cleanup and linkcheck cronjobs.

5

ScriptDirectories

Summary

Sets one or multiple directories that may contain cronjobs.

Usage

```
ScriptDirectories[]= directory
ScriptDirectories[]= directory2
...
```

Description

The cronjob script will search through all the ScriptDirectories to find cronjobs.

Examples

```
ScriptDirectories[]=cronjobs
ScriptDirectories[]=cronjobs2
Scripts[]=workflow.php
Scripts[]=notification.php
Scripts[]=linkcheck.php
```

This example will search through the cronjobs and cronjobs2 directories to find the workflow, notification, and linkcheck cronjobs.

5

5.14.11   datatype.ini

The configuration blocks are documented in the following sections:

- [ClassEditSettings] (page 1806)
- [CollectionSettings] (page 1810)
- [EditSettings] (page 1812)
- [ResultSettings] (page 1808)
- [ViewSettings] (page 1814)

5

[ClassEditSettings]

This configuration block contains settings related to the behavior of datatypes when class editing.

GroupedInput   (page 1807)
Lists the datatypes that needs to group their input fields when displayed in class edit mode.

5

GroupedInput

Summary

Lists the datatypes that needs to group their input fields when displayed in class edit mode.

Usage

```
GroupedInput[]=<datatype>
```

Description

This setting lists the datatypes that needs to group their input fields when displayed in class edit mode.
But keep in mind that the datatype may override this using PHP code.
Possible values are datatypes. A view the datatypes used by your installation visit the following location

```
<eZ  Publish installation>\kernel\classes\datatypes
```

By default, no datatypes are set to group input data when displayed in the class edit mode.

This setting is case sensitive.

5

[ResultSettings]

This configuration block contains settings related to the behavior of datatypes in relation to results, after information collection.

GroupedInput   (page 1809)

Lists the datatypes that need to group their data when displayed after the information collection mode.

5

GroupedInput

Summary

Lists the datatypes that need to group their data when displayed after the information collection mode.

Usage

```
GroupedInput[]=<datatype>
```

Description

This setting lists the datatypes that need to group their data when displayed after the information collection mode.
But keep in mind that the datatype may override this using PHP code.
Possible values are datatypes. A view the datatypes used by your installation visit the following location:

```
<eZ Publish installation>\kernel\classes\datatypes
```

By default, no datatypes are set to group input data when displayed after the information collection mode.

This setting is case sensitive.

5

[CollectionSettings]

This configuration block contains settings related to the behavior of datatypes when content information collection.

GroupedInput   (page 1811)

Lists the datatypes that need to group their input fields when displayed in information collection mode.

5

GroupedInput

Summary

Lists the datatypes that need to group their input fields when displayed in information collection mode.

Usage

```
GroupedInput[]=<datatype>
```

Description

This setting lists the datatypes that need to group their input fields when displayed in information collection mode.
But keep in mind that the datatype may override this using PHP code.
Possible values are datatypes. A view the datatypes used by your installation visit the following location:

```
<eZ  Publish installation>\kernel\classes\datatypes
```

This setting is case sensitive.

Examples

```
GroupedInput[]
GroupedInput[]=ezauthor
GroupedInput[]=ezbinaryfile
GroupedInput[]=ezdate
GroupedInput[]=ezdatetime
GroupedInput[]=ezimage
GroupedInput[]=ezinisetting
GroupedInput[]=ezmatrix
GroupedInput[]=ezmedia
GroupedInput[]=ezmultioption
GroupedInput[]=ezobjectrelationlist
GroupedInput[]=ezpackage
GroupedInput[]=ezrangeoption
GroupedInput[]=ezselection
GroupedInput[]=eztime
GroupedInput[]=ezurl
GroupedInput[]=ezuser
```

By default, these datatypes are set to group input data when displayed in information collection mode.

5

[EditSettings]

This configuration block contains settings related to the behavior of datatypes when content editing.

GroupedInput (page 1813)

Lists the datatypes that need to group their input fields when displayed in edit mode.

5

GroupedInput

Summary

Lists the datatypes that need to group their input fields when displayed in edit mode.

Usage

```
GroupedInput[]=<datatype>
```

Description

This setting lists the datatypes that need to group their input fields when displayed in edit mode. But keep in mind that the datatype may override this using PHP code.
Possible values are datatypes. A view the datatypes used by your installation visit the following location:

```
<eZ Publish installation>\kernel\classes\datatypes
```

This setting is case sensitive.

Examples

```
GroupedInput[]=ezauthor
GroupedInput[]=ezbinaryfile
GroupedInput[]=ezdate
GroupedInput[]=ezdatetime
GroupedInput[]=ezimage
GroupedInput[]=ezinisetting
GroupedInput[]=ezmatrix
GroupedInput[]=ezmedia
GroupedInput[]=ezoption
GroupedInput[]=ezmultioption
GroupedInput[]=ezobjectrelationlist
GroupedInput[]=ezpackage
GroupedInput[]=ezrangeoption
GroupedInput[]=eztime
GroupedInput[]=ezurl
GroupedInput[]=ezuser
GroupedInput[]=ezprice
```

By default, these datatypes are set to group input data when displayed in edit mode

5

[ViewSettings]

This configuration block contains settings related to the behavior of datatypes when in content view.

CustomActionMap   (page 1815)

> List of custom datatype actions and corresponding module or view or just view in case of content module where they should be executed

GroupedInput   (page 1817)

> Lists datatypes that need to group input data when displayed in view mode.

5

CustomActionMap

Summary

List of custom datatype actions and corresponding module or view or just view in case of content
module where they should be executed

Usage

```
CustomActionMap[ActionAddComment]=comment/add
```

Description

List of custom datatype actions and corresponding <module>/<view> or just <view> in case
of content module where they should be executed, eg:

- CustomActionMap[ActionAddToBasket]=shop/basket

- CustomActionMap[ActionCollectInformation]=collectinformation

Note: Action definition should be defined in contentActionList function of the datatype, see
ez[multi]pricetype.php for code example.

eZ Comments, or any other extension, might want to provide a custom action, in the eZ Comments case the possibility to comment on content. With custom actions you don't get form in
form issues, as all actions share one form, thus you can have both information collection, shop
and comment actions on the same object. The form goes to content/action which then executes
the <module>/<view> you defined in CustomActionMap, if it can't be found you'll get an error.

Examples

Example

```
extension/ezcomments/settings/datatype.ini:
<code>
#?ini charset="utf-8"?

[ViewSettings]
CustomActionMap[ActionAddComment]=comment/add
</code>

extension/ezcomments/datatypes/ezcomcomments/ezcomcommentstype.php:
<code>
    /**
     * Return content action(s) which can be performed on object containing
```

5

```
    * the current datatype. Return format is array of arrays with key 'name'
    * and 'action'. 'action' can be mapped to url in datatype.ini
    *
    * @param eZContentClassAttribute $classAttribute
    * @return array
   */
   function contentActionList( $classAttribute )
   {
       $actionList = parent::contentActionList( $classAttribute );
       $actionList[] = array( 'name' => ezpI18n::tr( 'ezcomments/
datatype', 'Comment on content' ),
                              'action' => 'ActionAddComment'
       );
       return $actionList;
   }
</code>
```

5

GroupedInput

Summary

Lists datatypes that need to group input data when displayed in view mode.

Usage

```
GroupedInput[]=<datatype>
```

Description

This setting lists the datatypes that need to group input data when displayed in view mode. But keep in mind that the datatype may override this using PHP code.

Possible values are datatypes. A view the datatypes used by your installation visit the following location:

```
<eZ Publish installation>\kernel\classes\datatypes
```

This setting is case sensitive.

Examples

```
GroupedInput[]=ezinisetting
GroupedInput[]=ezmultioption
GroupedInput[]=ezoption
GroupedInput[]=ezrangeoption
GroupedInput[]=ezpackage
GroupedInput[]=ezuser
```

By default, these datatypes are set to group input data when displayed in view mode.

5

5.14.12 datetime.ini

5.14.13   dbschema.ini

The configuration blocks are documented in the following sections:

- [SchemaSettings] (page 1820)

5

[SchemaSettings]

SchemaHandlerClasses

Summary

sets the schema handler classes

Usage

```
SchemaHandlerClasses[]=
```

Description

This setting was added in eZ Publish 4.1, to add support for mysql.

Examples

```
SchemaHandlerClasses[mysql]=eZMysqlSchema
```

5

SchemaPaths

Summary

sets paths

Usage

```
SchemaPaths[]=
```

Description

This setting was added in eZ Publish 4.1, to add support for mysql.

Examples

```
SchemaPaths[mysql]=lib/ezdschema/classes/ezmysqlschema.php
```

5

5.14.14 debug.ini

The configuration blocks are documented in the following sections:

\begin{itemize}   \item{}   [DebugSettings]\refer{label8a5c29ff34f83e92fbe2f9120f2110b9}
\item{} [GeneralCondition]\refer{label85e3d16e0a7459b0c0449d05de539fb6} \end{itemize}

5

[GeneralCondition]

{condition}   (page)
      This settings sets debugging for specific conditions.

5

{condition}

Summary

This settings sets debugging for specific conditions.

Usage

```
{condition}=enabled | disabled
```

Description

This setting allows you to enable or disable debugging for specific conditions.
 When specifying conditions a "-" (dash) is used, for example "kernel-content-view".
 By default these are turned off in the global debug.ini file and if needed turned on in override files.

Keep in mind that site.ini [DebugSettings] also contains settings related to debugging.

This setting is case sensitive.

Examples

```
kernel-content-object-copy=disabled
 workflow-event=disabled
 workflow-process=disabled
 workflow-type=disabled
 module-run=disabled
 kernel-content-view=disabled
 kernel-content-view-cache=disabled
 kernel-content-class=disabled
 kernel-urltranslator=disabled
 kernel-search-ezsearch=disabled
 kernel-search-openfts=disabled
 kernel-content-treenode=disabled
 kernel-content-object-related-objects=disabled
 kernel-content-object-limitation=disabled
 kernel-policy-limitation=disabled
 collaboration-group-tree=disabled
 collaboration-item-list=disabled
 kernel-siteaccess=disabled
 kernel-db-postgresql=disabled
 lib-ezutil-web=disabled
 kernel-workflow-approve=disabled
 eztemplate-tree-cache=disabled
 eztemplate-compile=disabled
```

5

```
lib-ezutils-dir=disabled
kernel-user=disabled
kernel-content-edit=disabled
kernel-notification=disabled
i18n-tstranslator=disabled
ezfile-create=disabled
lib-ezfile-seek=disabled
lib-ezfile-openclose=disabled
lib-ezfile-readwrite=disabled
lib-ezfile-copy=disabled
lib-ezfile-gziplibz=disabled
kernel-ezpackage-store=disabled
lib-ezimage-shell=disabled
```

 These are the default settings, but by default these are turned off in debug.ini and if needed turned on in override files.

5

[DebugSettings]

ConditionDebug   (page 1828)
    Sets condition debugging.

5

ConditionDebug

Summary

Sets condition debugging.

Usage

```
ConditionDebug= enabled | disabled
```

Description

This setting allows you to enable the master switch for conditional debugging, if disabled no conditions are checked.
Keep in mind that site.ini [DebugSettings] also contains settings related to debugging.

This setting is case sensitive.

Examples

```
ConditionDebug=enabled
```

This is the default setting.

5

5.14.15 design.ini

The configuration blocks are documented in the following sections:

- [ExtensionSettings] (page 1836)
- [JavaScriptSettings] (page 1830)
- [StylesheetSettings] (page 1832)

5

[JavaScriptSettings]

JavaScriptList   (page 1831)
Sets the JavaScript files to include in the pagelayout.

5

JavaScriptList

Summary

Sets the JavaScript files to include in the pagelayout.

Usage

```
JavaScriptList[]= javascript1
JavaScriptList[]= javascript2
...
```

Description

You should provide the full path relative to the design directory.

Examples

```
JavaScriptList[]=javascipts/mynewticker.js
```

Using this configuration eZ Publish will load the file "/design/ *mydesign*/javascripts/ mynewsticker.js"

5

[StylesheetSettings]

ClassesCSS   (page 1834)
      Sets the CSS files which contains class definitions for the base layout.

CSSFileList   (page 1833)
      Sets the CSS files to include in the pagelayout.

SiteCSS   (page 1835)
      Sets the global CSS file for designs based on the base layout.

5

CSSFileList

Summary

Sets the CSS files to include in the pagelayout.

Usage

```
CSSFileList[]= css_file1
CSSFileList[]= css_file2
...
```

Description

Sets the CSS files that should automatically be included in the pagelayout. You should provide the full path relative to the stylesheets subdirectory in the design directory.

Examples

```
CSSFileList[]=mycss.css
```

Using this configuration eZ Publish will load the file "/design/ *mydesign*/stylesheets/mycss.css".

5

ClassesCSS

Summary

Sets the CSS files which contains class definitions for the base layout.

Usage

```
ClassesCSS= path_to_CSS_file
```

5

SiteCSS

Summary

Sets the global CSS file for designs based on the base layout.

Usage

```
SiteCSS= path_to_css_file
```

5

[ExtensionSettings]

DesignExtensions (page)
    Sets which extensions that have designs.

5

DesignExtensions

Summary

Sets which extensions that have designs.

Usage

```
DesignExtensions[]= extension1
DesignExtensions[]= extension2
...
```

Description

By default eZ Publish will only look for designs in the "/design" directory. If your extensions provide designs you must tell eZ Publish by providing the extension name in the DesignExtensions setting.

eZ Publish will look for designs in the directory "/extension/ *extension_name*/design/"

This setting is typically overriden in the design.ini.append in extensions.

Examples

```
DesignExtensions[]=myextension
```

eZ Publish will now search for designs in "/extension/myextension/design".

5

5.14.16   error.ini

5.14.17 extendedattributefilter.ini

5.14.18   ezxml.ini

The configuration blocks are documented in the following sections:

- [InputSettings] (page 1846)
- [OutputSettings] (page 1843)
- [TagSettings] (page 1841)

5

[TagSettings]

TagPresets

Summary

setting to define sets of tags

Usage

```
TagPresets[]=
```

Description

This setting is available since eZ Publish 4.1. It enables you to define different sets of tags for different levels or types of content editors when using the Online Editor. For example, you might have a full set of tags for an administrator but a subset of tags for specific editors. Note that this is only used by Online Editor 5.0 to limit visible buttons.

Examples

Example 1

```
TagPresets[full]=All tags
```

5

[OutputSettings]

AliasClasses

Summary

sets which alias to use

Usage

```
AliasClasses[]=
```

Description

Since eZ Publish 4.1 this setting replaces the "Alias" configuration setting.

Examples

Example 1

```
AliasClasses[eZXHTMLXMLOutput]=eZPlainXMLOutput
```

HandlerClass

Summary

sets which output handler to use

Usage

```
HandlerClass=
```

Description

Since eZ Publish 4.1 this setting replaces the "Handler" configuration setting.

This setting allows you to set which output handler to use. As value it takes a PHP class name as the output handler. The default setting is eZXHTMLXMLOutput. eZPlainXMLOutput can be used to see raw XML data.

Examples

Example 1

```
HandlerClass=eZPlainXMLOutput
```

5

[InputSettings]

AliasClasses   (page 1847)
      sets which alias to use

AllowNumericEntities   (page 1849)
      Controls wether numerical HTML entities should be allowed.

HandlerClass   (page 1848)
      sets which input handler to use

5

AliasClasses

Summary

sets which alias to use

Usage

```
AliasClasses[]=
```

Description

Since eZ Publish 4.1 this setting replaces the "Alias" configuration setting.

Examples

Example 1

```
AliasClasses[eZSimplifiedXMLInput]=myXMLInputHandler
```

5

HandlerClass

Summary

sets which input handler to use

Usage

```
HandlerClass=
```

Description

Since eZ Publish 4.1 this setting replaces the "Handler" configuration setting.

This setting allows you to set which input handler to use. As value it takes a PHP class name as the input handler. The default setting is eZSimplifiedXMLInput.

Examples

Example 1

```
HandlerClass=eZSimplifiedXMLInput
```

5

AllowNumericEntities

Summary

Controls wether numerical HTML entities should be allowed.

Usage

```
AllowNumericEntities=enabled|disabled
```

Description

This directive can be used to control wether numerical HTML entities entered into XML fields (when using the XML block datatype) should be translated to their corresponding characters/ symbols. By default, numerical HTML entities are disabled, but this doesn't disallow the direct entry of entities in XML fields. What happens is that if you enter the following...

```
&#1234;
```

...you will see the exact same text when the page containing the XML is being shown.

When enabled, the entered entities will be translated and thus the system will show their corresponding symbols/characters/letters instead of the entered values. The syntax for entities is standard HTML: an ampersand followed by a hash, the number of the entity and a semicolon.

This feature is useful when there is a need to enter single special characters like for example Greek letters even if they are not supported by the character set being used. For example, the following...

```
&#937;
```

...the system will produce a Greek capital omega letter instead of showing the code.

5

5.14.19   fetchalias.ini

5.14.20   file.ini

The configuration blocks are documented in the following sections:

- [ArchiveSettings] (page 1860)

- [BinaryFileSettings] (page 1862)

- [ClusteringSettings] (page 1866)

- [eZDFSClusteringSettings] (page 1854)

- [FileSettings] (page 1858)

- [PassThroughSettings] (page 1852)

5

[PassThroughSettings]

ContentDisposition   (page 1853)

    Controls the way binary files should be served over HTTP, either as downloadable as an attachment, or displayed inline in the browser.

5

ContentDisposition

Summary

Controls the way binary files should be served over HTTP, either as downloadable as an attachment, or displayed inline in the browser.

Usage

```
ContentDisposition[Mime-Type]=inline | attachment
```

Description

Controls the way binary files should be served over HTTP, either as downloadable as an attachment, or displayed
inline in the browser.

Possible values are:

- "attachment": default value

- "inline"

This setting is case sensitive.

Examples

```
ContentDisposition[application/x-shockwave-flash]=inline

ContentDisposition[application/x-pdf]=attachment
```

5

[eZDFSClusteringSettings]

This blocks contains settings required when clustering with an eZ DFS File Handler.

\begin{description} \item[DBBackend]\refer{label0056a41664a8ca0271e01a7c2151101b}\ \\
Sets the database back-end which should be used by the eZ DB File Handler.

\item[DBSocket]\refer{label738339c93c1d8c18a10a0d4bd9a2cf3b}\ \\ Enables the use of
sockets by eZ DFS FIle Handler.

\item[MountPointPath]\refer{labela4b421139dd7d18e73b29bd79f4ff082}\ \\ This setting de-
fines the path to the NFS Mount Point, which should be used when clustering with an eZ DFS
File Handler.

\end{description}

5

DBSocket

Summary

Enables the use of sockets by eZ DFS FIle Handler.

Usage

```
DBSocket=
```

Description

When using eZDFS File Handler it isn't possible to use a socket, for this reason "DBSocket"
needs to be left empty and any other value will cause a fatal error of type "unable to connect to
database...".

Examples

```
DBSocket=False
```

5

DBBackend

Summary

Sets the database back-end which should be used by the eZ DB File Handler.

Usage

```
DBBackend=eZDFSFileHandlerMySQLBackend
```

Description

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, therefore the only possible value at the moment is "eZDFSFileHandlerMySQLBackend".

This setting is case sensitive.

Examples

```
DBBackend=eZDFSFileHandlerMySQLBackend
```

5

MountPointPath

Summary

This setting defines the path to the NFS Mount Point, which should be used when clustering with an eZ DFS File Handler.

Usage

```
MountPointPath=<path to NFS mount point>
```

Description

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.

Also each eZ Publish installation should have a local mount point to the same NFS export. The NFS server has to be available and writeable by the webserver's user on each eZ Publish server.

Also it 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.

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. The location of the local mount point must be set with this setting and it can be relative or absolute.

This setting is case sensitive.

Examples

```
MountPointPath=var/nfsmount
```

5

[FileSettings]

Handler   (page)
    Use this setting to define which handler to use.

5

Handler

Summary

Use this setting to define which handler to use.

Usage

```
Handler[<handler name>]=<class name handler>
```

Description

Use this setting to define which handler to use.

Use the name of the handler as key:

- Use [gzip] when using the generic gzip handler

- Use [gziplib] or [gzibshell] when using a specific gzip handler

The values are the class names of the handlers available at lib/ezfile/classes:

- "ezgzipzlibcompressionhandler" handles files compressed with gzip using the zlib extension. More information on the zlib extension can be found here: http://www.php.net/manual/en/ref.zlib.php

- "ezgzipshellcompressionhandler" handles files compressed with gzip using the shell commands. It will handle GZIP compression by executing the 'gzip' executable, without this executable the handler can't work.

- "ezgzipcompressionhandler" handles files compressed with gzip. This handler is a wrapper of the "eZGZIPZLIBCompressionHandler" and "eZGZIPShellCompressionHandler".

This setting is case sensitive.

Examples

```
Handler[gzip]=ezgzipcompressionhandler

Handlers[gzipzlib]=ezgzipzlibcompressionhandler

Handlers[gzipshell]=ezgzipshellcompressionhandler
```

[ArchiveSettings]

This block contains settings related to file archive handling.

Handler   (page 1861)
    DEPRECATED since eZ Publish 4.2.

Handler

Summary

DEPRECATED since eZ Publish 4.2.

Usage

```
Handler[<archive type>]=<name handler>
```

Description

This setting sets the archive handler (located here "lib\ezfile\classes").

Possible handlers which can be used as values:

- "eztararchivehandler" creates a (compressed) Tar archive
- "ezarchivehandler" handles various kinds of archive formats.

This setting is case sensitive.

Examples

```
Handler[tar]=eztararchivehandler
```

5

[BinaryFileSettings]

This block contains settings related to binary file handling.

ExtensionRepositories   (page)
> DEPRECATED since eZ Publish 4.1

Handler   (page)
> This setting sets which binary file handler to use.

Repositories   (page)
> DEPRECATED since eZ Publish 4.1

5

ExtensionRepositories

Summary

DEPRECATED since eZ Publish 4.1

Usage

```
ExtensionRepositories[<key>]=<name extension>
```

Description

This setting lists the extensions which have binaryhandlers.

It is common use to create a settings/override/file.ini.append.php file in your extension directory, where you add the extension name to the "ExtensionRepositories" setting. This must be done in order to automatically get the handlers from the extension when the extension in enabled. Also all handlers need to be placed in the folder:

```
kernel/classes/binaryhandlers.
```

This setting is case sensitive.

5

Repositories

Summary

DEPRECATED since eZ Publish 4.1

Usage

```
Repositories[<key>]=<class name repositories>
```

Examples

```
Repositories[]=kernel/classes
```

5

Handler

Summary

This setting sets which binary file handler to use.

Usage

```
Handler=<class name file handler>
```

Description

Possible values are:

- "eZFilePassthroughHandler" handles file downloading by passing the file through PHP. The previous value "ezfilepassthrough" should no longer be used.

- "eZFileDirectHandler" and the previous value "ezfiledirect" are deprecated. The eZFileDirectHandler handled file downloading by passing an URL directly to the file. If you wish to enable direct downloads, use the value "eZFilePassthroughHandler". But keep in mind that when using direct downloads role permissions will not be checked.

This setting is case sensitive.

Examples

```
Handler=eZFilePassthroughHandler
```

5

[ClusteringSettings]

Contains settings required when clustering

DBBackend   (page 1877)
> Sets which database back end to use when clustering with the eZDB File Handler.

DBChunkSize   (page 1870)
> This determines the byte size of the blocks into which files are split when they are fetched and inserted from the database, when clustering with the eZDB File Handler.

DBConnectRetries   (page 1869)
> This sets the permitted amount of connection retries when trying to connect to the database, when clustering with the eZDB File Handler.

DBExecuteRetries   (page 1868)
> This sets the permitted amount of executer retries.

DBHost   (page 1876)
> Sets which database host to use when clustering with the eZDB File Handler.

DBName   (page 1873)
> This sets which database to use when clustering with the eZ DB File Handler.

DBPassword   (page 1871)
> This sets which password to use for your database when clustering with the eZDB File Handler.

DBPort   (page 1875)
> This setting sets which port to use when clustering with the eZDB File Handle

DBSocket   (page 1874)
> This sets which socket to use when clustering with the eZDB File Handler

DBUser   (page 1872)
> This sets which username to use for your database when clustering with the eZDB File Handler.

ExtensionDirectories   (page 1867)
> DEPRECATED since eZ Publish 4.1.

FileHandler   (page 1878)
> This setting sets the cluster file handler.

NonExistentStaleCacheHandling   (page 1879)
> This setting defines what happens when a requested cache file is already being generated and no expired cache file exists.

5

ExtensionDirectories

Summary

DEPRECATED since eZ Publish 4.1.

Usage

```
ExtensionDirectories[]=<name extension>
```

Description

This setting is deprecated. It was intended to allow users to use their own cluster file handler. After locating the cluster file handler for an extension at this location:

```
extension/<your_extension>/clusterfilehandlers
```

The extension name needs to be added to the ExtensionDirectories list.

This setting is case sensitive.

5

DBExecuteRetries

Summary

This sets the permitted amount of executer retries.

Usage

```
DBExecuteRetries=<integer>
```

Description

This sets the permitted amount of executer retries.

When using an eZ DFS File Handler please refer to the settings in file.ini [eZDFSClusteringSettings].

By default this is set to "20".

Examples

```
DBExecuteRetries=20
```

5

DBConnectRetries

Summary

This sets the permitted amount of connection retries when trying to connect to the database, when clustering with the eZDB File Handler.

Usage

```
DBConnectRetries=<integer>
```

Description

This sets the permitted amount of connection retries when trying to connect to the database.

When using an eZ DFS File Handler please refer to the settings in file.ini [eZDFSClusteringSettings].

By default this is set to "3".

Examples

```
DBConnectRetries=3
```

5

DBChunkSize

Summary

This determines the byte size of the blocks into which files are split when they are fetched and inserted from the database, when clustering with the eZDB File Handler.

Usage

```
UsageDBChunkSize=<integer>
```

Description

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.

By default this is set to "65535".

Examples

```
DBChunkSize=65535
```

5

DBPassword

Summary

This sets which password to use for your database when clustering with the eZDB File Handler.

Usage

DBPassword=<password>

Description

This sets which password to use for your database when clustering with the eZDB File Handler. Use the actual password as value, but you can also leave the password field empty.

In most cases this value will be the same as the "Password" setting specified under the [DatabaseSettings] (page 2339) block of your "site.ini.append.php" configuration file.
When using an eZ DFS File Handler please refer to the settings in file.ini [eZDFSClusteringSettings (page 1854)].

This setting is case sensitive.

Examples

DBPassword=mypassword

5

DBUser

Summary

This sets which username to use for your database when clustering with the eZDB File Handler.

Usage

```
DBUser=<name user>
```

Description

This sets which username to use for your database when clustering with the eZDB File Handler. Use the user name as value. In most cases this value will be the same as the "User" setting specified under the [DatabaseSettings] (page 2339) block of your "site.ini.append.php" configuration file.

When using an eZ DFS File Handler please refer to the settings in file.ini [eZDFSClusteringSettings (page 1854)].

This setting is case sensitive.

Examples

```
DBUser=root
```

By default this setting is set to "root".

5

DBName

Summary

This sets which database to use when clustering with the eZ DB File Handler.

Usage

```
DBName=<name database>
```

Description

This sets which database to use when clustering with the eZDB File Handler.

Use the actual database name as value. In most cases this value will be the same as the "Database" setting specified under the [DatabaseSettings] (page 2339) block of your "site.ini.append.php" configuration file.
When using an eZ DFS File Handler please refer to the settings in file.ini [eZDFSClusteringSettings (page 1854)].

By default this is set to "cluster".

This setting is case sensitive.

Examples

```
DBName=cluster
```

5

DBSocket

Summary

This sets which socket to use when clustering with the eZDB File Handler

Usage

```
DBSocket=<path to socket>
```

Description

This sets which socket to use when clustering with the eZDB File Handler.

Make sure that the "DBSocket" setting is correct. Leave it empty if you have "Socket=disabled" under the [DatabaseSettings (page 2339)] block in "site.ini.append.php".

When using an eZ DFS File Handler please refer to the settings in file.ini [eZDFSClusteringSettings (page 1854)].

This setting is case sensitive

Examples

```
DBSocket=/tmp/mysql.sock
```

5

DBPort

Summary

This setting sets which port to use when clustering with the eZDB File Handle

Usage

```
DBPort=<port number>
```

Description

This setting sets which port to use when clustering with the eZDB File Handler.

When using an eZ DFS File Handler please refer to the settings in file.ini [eZDFSClusteringSettings (page 1854)].

Examples

```
DBPort=3306
```

By default this setting is set to port 3306.

5

DBHost

Summary

Sets which database host to use when clustering with the eZDB File Handler.

Usage

```
DBHost=<name host OR Host IP address>
```

Description

Use the actual host name as value. In most cases this value will be the same as the "Server" setting specified under the [DatabaseSettings] (page 2339) block of your "site.ini.append.php" configuration file.

When using an eZ DFS File Handler please refer to the settings in file.ini [eZDFSClusteringSettings (page 1854)].

Note: It is recommended to use a distinct database server if you are clustering for a high traffic website.

By default this setting is set to "db".

This setting is case sensitive.

Examples

```
DBHost=localhost
```

```
DBHost=127.0.0.1
```

5

DBBackend

Summary

Sets which database back end to use when clustering with the eZDB File Handler.

Usage

```
DBBackend=<class name handler back-end>
```

Description

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].

Kzeep in mind that for this setting the previously used value "mysql" is deprecated.

Possible values are:

- "eZDBFileHandlerMysqlBackend" when using MySQL

- "eZDBFileHandlerOracleBackend" when using Oracle (this will require the ezoracle extension)

This setting is case sensitive.

Examples

```
DBBackend=eZDBFileHandlerMysqlBackend
DBBackend=eZDBFileHandlerOracleBackend
```

5

FileHandler

Summary

This setting sets the cluster file handler.

Usage

```
FileHandler=<class name file handler>
```

Description

This setting sets the cluster file handler.

For more information regarding the cluster file handlers visit chapter "cluster file handlers (page 297)" of the clusterdocumentation.

Possible file handlers are:

- eZFSFileHandler

- eZFS2FileHandler

- eZDBFileHandler

- eZDFSFileHandler

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 file.ini [eZDFSClusteringSettings (page 1854)].

This setting is case sensitive.

Examples

```
FileHandler=eZDBFileHandler
```

The eZ DB File Handler will be used for cluster filehandling.

5

NonExistentStaleCacheHandling

Summary

This setting defines what happens when a requested cache file is already being generated and no expired cache file exists.

Usage

```
NonExistentStaleChacheHandling[<cache type>]=wait|generate
```

Description

This is a new setting available since eZ Publish 4.1. NonExistentStaleCacheHandling 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). The second request can either wait for the first request to finish, or it can generate its own result to display without storing it.

Note that this is an advanced setting and should only be customized if it is specifically required.

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.

This setting is case sensitive.

Examples

```
NonExistantStaleCacheHandling[viewcache]=wait

NonExistantStaleCacheHandling[cacheblock]=wait

NonExistantStaleCacheHandling[misc]=wait
```

5.14.21   i18n.ini

The configuration blocks are documented in the following sections:

- [CharacterSettings] (page 1881)

5

[CharacterSettings]

Charset   (page 1882)
    Sets the character set that eZ Publish should use

5

Charset

Summary

Sets the character set that eZ Publish should use

Usage

```
Charset=character_set
```

Description

This directive can be used to set the internal charset for the site. The default value is "utf-8". Note that eZ Publish 4 requires UTF-8 to be used as the character set in the database, and thus you should never change this setting.

Examples

Example 1

```
Charset=utf-8
```

This will set the output character set of eZ Publish to UTF-8 (default).

5

5.14.22 icon.ini

The configuration blocks are documented in the following sections:

\begin{itemize} \item{} [FlagIcons]\refer{label4cc66c7585209e2a03b57dd7ab7507f0} \item{} [IconSettings]\refer{labelabfad951c9f4a17328518e23541b06f7} \item{} [\{type\}Icons]\refer{label298dcf01bf65a13c587068c0c58df29f} \end{itemize}

5

[FlagIcons]

Repository

Summary

Sets the location of the Flag icons repository.

Usage

```
Repository=<path_to_repository]
```

Description

This setting sets the location of the Flag icons repository. It tells your installation where to look for flag icons which are relevant to eZ Publish.

This setting overrides the default setting [IconSettings] Repository.

This setting is case sensitive.

Examples

```
Repository=share/icons
```

This is the default repository.

5

Theme

Summary

Sets which theme to use for the flag icons.

Usage

```
Theme=<theme_name>
```

Description

This setting sets which theme to use for the flag icons. This setting will override the default theme setting [IconSettings] Theme.This setting is case sensitive.

Examples

```
Theme=flags
```

This is the default theme.

5

IconFormat

Summary

Sets which suffix to use for flag icon files.

Usage

```
IconFormat=gif
```

Description

This setting sets which suffix to use for all flag icon files.This setting is case sensitive.

Examples

```
IconFormat=gif
```

This is the default setting. All flag icons will have the suffix "gif", for example eng-GB.gif

5

DefaultIcon

Summary

Sets the default icon to use.

Usage

```
DefaultIcon=<icon_name>
```

Description

This setting sets the default icon to use for flag icons. This setting is case sensitive.

Examples

```
#DefaultIcon=unknown
```

By default this setting is set to unknown and commented out.

5

[{type}Icons]

Theme

Summary

Sets which theme to use for the icon type.

Usage

```
Theme=<theme_name>
```

Description

This setting sets which theme to use for the icon type defined in by the configuration block. This setting will override the default theme setting [IconSettings] Theme.This setting is case sensitive.

Examples

```
[MimeIcons]
Theme=crystal
```

The default theme for MIME-icons is "crystal".

5

Size

Summary

Sets which size to use for the icon type.

Usage

```
Size=<size>
```

Description

This setting sets the size for the icon type defined in by the configuration block.   This setting will override the default size setting [IconSettings] Size.

This setting is case sensitive.

Examples

```
[ClassIcons]
Size=normal
```

The default size for Class icons is "normal".

5

{type}Map

Summary

Sets which map to use for the icon type.

Usage

<type>Map[<type>]=<icon_file>

Description

This setting is used to map specific icon types to their corresponding icon file. The entire file name of the icon file is used as value. It is possible to map an icon file to the group of the type instead of a specific type.
For example when using this setting in [MimeIcons] the syntax will be like this:

```
MimeMap[<MIME_type>]=<icon_file>
```

When using this setting in [ActionIcons] the syntax will be like this:

```
ActionMap[<Action_type>]=<icon_file>
```

This setting is case sensitive.

5

[IconSettings]

Repsitory   (page 1897)
> Sets the default location of the icon repository.

Size   (page 1895)
> Sets the default icon size.

Sizes   (page 1894)
> Sets override and new sizes.

Theme   (page 1896)
> Sets the default theme to use for all icon types.

5

Sizes

Summary

Sets override and new sizes.

Usage

```
Sizes[<identifier>]=<width>x<height>ORSizes[<identifier>]=<width>x<height>;
<subpath>
```

Description

This setting allows you to override default sizes or add new sizes. It is important to make sure that all used themes have the sizes listed here defined.The following syntax is used (with or without the subpath):

```
Sizes[<identifier>]=<width>x<height>;<subpath>
```

Each size refers to the name of the subdirectory of the icon theme. The value are two numbers, the width and height of the icon in number of pixels, with an "x" in between. It is possible, but not necessary to add a subpath after the width and height. This subpath will be used for fetching the icon.

This setting is commented out by default by adding a "#".

This setting is case sensitive.

Examples

Example 1

```
Sizes[normal]=32x32
```

For the icon type "normal" the icon size is defined at a width of 32 pixels and a height of 32 pixels. This setting will override the default icon size for the "normal" icons.**Example 2**

```
Sizes[normal]=32x32;normal-icons
```

For the icon type "normal" the icon size is defined at a width of 32 pixels and a height of 32 pixels. The subpath "normal-icons" will be used for fetching these icons. This setting will override the default icon size for the "normal" icons.

5

Size

Summary

Sets the default icon size.

Usage

```
Size=<string>
```

Description

This setting sets the default icon size. The value here is a string, not the actual hight and width. The actual sizes are defined with the setting [IconSettings] Sizes[]=.

This setting is case sensitive.

Examples

```
Size=normal
```

This is the default setting.

5

Theme

Summary

Sets the default theme to use for all icon types.

Usage

```
Theme=<theme_name>
```

Description

This setting sets which theme to use as current theme for all icon types. It is possible to set a different theme per icon type, by setting configuration block specifically for this type. For example the setting [MimeIcons] Theme will override the default theme for MIME icons.

This setting is case sensitive.

Examples

```
Theme=crystal
```

The default theme for icons is crystal.

5

Repsitory

Summary

Sets the default location of the icon repository.

Usage

```
Repository=<path_to_location>
```

Description

This setting sets the default location of the icon repository. It tells your installation where to look for icons which are relevant to eZ Publish.

This setting is case sensitive.

Examples

```
Repository=share/icons
```

This is the default folder containing eZ Publish related icons.

5

5.14.23 image.ini

The configuration blocks are documented in the following sections:

- [AliasSettings] (page 1911)
- [AnalyzerSettings] (page 2019)
- [AnimatedGIF] (page 1974)
- [EXIFAnalyzer] (page 2028)
- [FileSettings] (page 1901)
- [GDSettings] (page 1899)
- [GD] (page 1994)
- [GIFAnalyzer] (page 2025)
- [gray] (page 1960)
- [ImageConverterSettings] (page 1986)
- [ImageMagick] (page 2001)
- [large] (page 1945)
- [LayeredPhotoshop] (page 1981)
- [medium] (page 1940)
- [MIMETypeSettings] (page 1970)
- [MyCustomConverter] (page 1988)
- [original] (page 1920)
- [OutputSettings] (page 1909)
- [reference] (page 1925)
- [rssitem] (page 1955)
- [rss] (page 1950)
- [ShellSettings] (page 1984)
- [small] (page 1930)
- [tiny] (page 1935)
- [zoom] (page 1965)
- [{myalias}] (page 1914)

5

[GDSettings]

HasGD2   (page 1900)

    If GD2 is installed, this setting can allow your system to use the GD2's advanced functionality.

5

HasGD2

Summary

If GD2 is installed, this setting can allow your system to use the GD2's advanced functionality.

Usage

```
HasGD2=true| false
```

Description

This requires GD2 to be compiled in the PHP module. If set to "true" a more advanced functionality will be available which is present in GD2. For more information on the GD functionality please visit the following page http://be.php.net/manual/en/ref.image.phpThis setting is case sensitive.

Examples

```
HasGD2=false
```

This is the default setting.

5

[FileSettings]

DirPermissions   (page 1904)
> Sets the access permissions for the new image directories.

ImagePermissions   (page 1902)
> Sets the access permissions for the new images.

PublishedImages   (page 1907)
> Sets which sub-directory to use for published images.

TemporaryDir   (page 1908)
> Sets which sub-directory to use for temporary image files.

VersionedImages   (page 1906)
> Sets which sub-directory to use for unpublished images.

5

ImagePermissions

Summary

Sets the access permissions for the new images.

Usage

```
ImagePermissions= 0666 or 0660
```

Description

Sets the access permission for the new images.Possible values:

- 0666: setting it to 0666 means that anyone can read and write the files and can be considered a security risk.

- 0660: It's preferred to use 0660 and make sure the web server has correct user/group access.

As value chmod commands are used, which allow you to set certain permissions for files on your Web server. These commands allow you to specify permissions for the owner, the group and everyone else.

It can be useful to know the meaning of the numbers used:

- The location of the numbers has a specific meaning. The second number sets the permissions for the owner, the third sets the permissions for the group, and the last number sets the permissions for everyone else.

- The number itself also has a specific permission meaning:
    - "0" (zero) will permit nothing
    - "1" will permit executing only
    - "2" permits writing only
    - "3" permits writing and executing
    - "4" permits reading only
    - "5" permits reading and executing
    - "6" permits reading and writing
    - "7" permits reading, writing and executing.

And as an example, to allow everyone to read, write and execute use "0777".

Examples

5

```
ImagePermissions=0666
```

This is the default setting.

5

DirPermissions

Summary

Sets the access permissions for the new image directories.

Usage

```
DirPermissions= 0777 or 0770
```

Description

This setting allows you to set the permission for new image directories.

Possible values:

- 0777: Setting it to 0777 means that anyone can read and write the files, and as such anyone can be considered a security risk.

- 0770: It's preferred to use 0770 and make sure the web server has correct user/group access.

As value chmod commands are used, which allow you to set certain permissions for files on your Web server. These commands allow you to specify permissions for the owner, the group and everyone else.

It can be useful to know the meaning of the numbers used:

- The location of the numbers has a specific meaning. The second number sets the permissions for the owner, the third sets the permissions for the group, and the last number sets the permissions for everyone else.

- The number itself also has a specific permission meaning:
    - "0" (zero) will permit nothing
    - "1" will permit executing only
    - "2" permits writing only
    - "3" permits writing and executing
    - "4" permits reading only
    - "5" permits reading and executing
    - "6" permits reading and writing
    - "7" permits reading, writing and executing.

And as an example, to allow everyone to read, write and execute use "0777".

5

Examples

```
DirPermissions=0777
```

This is the default setting.

5

VersionedImages

Summary

Sets which sub-directory to use for unpublished images.

Usage

```
VersionedImages=<name sub-directory>
```

Description

Sets which sub-directory of var/storage to use for unpublished images, e.g. drafts, archived,.. This setting is case sensitive.

Examples

```
VersionedImages=images-versioned
```

5

PublishedImages

Summary

Sets which sub-directory to use for published images.

Usage

```
PublishedImages=<name sub-directory>
```

Description

Sets which sub-directory of var/storage to use for published images.

This setting is case sensitive.

Examples

```
PublishedImages=images
```

This is the default setting.

5

TemporaryDir

Summary

Sets which sub-directory to use for temporary image files.

Usage

```
TemporaryDir=<name sub-directory>
```

Description

Sets which sub-directory of var/cache to use for temporary image files.

This setting is case sensitive.

Examples

```
TemporaryDir=imagetmp
```

This is the default setting.

5

[OutputSettings]

AllowedOutputFormat   (page 1910)
    lists the MIME types which are allowed as output type.

5

AllowedOutputFormat

Summary

lists the MIME types which are allowed as output type.

Usage

```
AllowedOutputFormat[]=<MIME type>
```

Description

A list of MIME types that are allowed as output type. This setting determines which formats you want your web page to display.

MIME types not listed here will be converted according to the setting [MimeTypeSettings] ConversionRules.

This setting is case sensitive.

Examples

```
AllowedOutputFormat[]=image/jpeg
AllowedOutputFormat[]=image/png
AllowedOutputFormat[]=image/gif
```

The formats jpeg, png and gif are allowed by default.

5

[AliasSettings]

Contains all Image alias settings.

AliasList (page 1912)
  Sets the list of aliases.

5

AliasList

Summary

Sets the list of aliases.

Usage

```
AliasList[]=<alias name>
```

Description

Defines a list of aliases that are available to the template engine and other clients. The alias must be defined as a separate INI block.

Possible values can be:

- reference
- small
- tiny
- medium
- large
- rss
- rssitem
- gray
- zoom

This setting is case sensitive.

Examples

```
AliasList[]=reference

AliasList[]=small

AliasList[]=tiny

AliasList[]=medium
```

5

```
AliasList[]=large

AliasList[]=rss
```

These six settings are enabled by default.

5

[{myalias}]

Example of an image alias, the group is called the same as the alias name.

Filters

Summary

Sets a list of filters to run.

Usage

```
Filters[]=<filter>=<filter-parameters>
```

Description

A list of filters to run, each filter must be supported by the active image converters. The parameter to use is the pixel size of the image. So when you need to use the width, use the horizontal pixel size of the image, and when you need to use the height use the vertical pixel size of the image.

A list of filters:

- geometry/scale: scales the image to the size specified, but maintaining the aspect ratio. The parameters here will be:

```
Filter[]=geometry/scale=<width>;<height>
```

- geometry/scalewidth: scales the image to the widthsize specified, but maintaining the aspect ratio.
  The parameters here will be:

```
Filter[]=geometry/scalewidth=<width>
```

- geometry/scaleheight: scales the image to the heightsize specified, but maintaining the aspect ratio.
  The parameters here will be:

```
Filter[]=geometry/scaleheight=<height>
```

- geometry/scaledownonly: scales the image to the size specified, but maintaining the aspect ratio. The image will not be scaled if the source size is smaller than the destination size.
  The parameters here will be:

```
Filter[]=geometry/scaledownonly=<width>;<height>
```

- geometry/scalewidthdownonly: scales the image to the widthsize specified, but maintaining the aspect ratio. The image will not be scaled if the source size is smaller than the destination size.
  The parameters here will be:

5

```
Filter[]=geometry/scalewidthdownonly=<width>
```

- geometry/scaleheightdownonly: scales the image to the heightsize specified, but maintaining the aspect ratio. The image will not be scaled if the source size is smaller than the destination size.
  The parameters here will be:

```
Filter[]=geometry/scaleheightdownonly=<height>
```

- geometry/scaleexact: scales the image to the exact size specified, but without maintaining the aspect ratio.
  The parameters here will be:

```
Filter[]=geometry/scaleexact=<width>;<height>
```

- geometry/scalepercent: scales the image to the size specified, but without maintaining the aspect ratio.
  The parameters here will be:

```
Filter[]=geometry/scalepercent=<%>
```

- geometry/crop: crops the image to the crop size specified and according to the crop location specified.
  The parameters here will be:

```
Filter[]=geometry/crop=<width after cropping>;<height after  cropping>;
<width location of the cropped image in respect to the  original image>;
<height location of the cropped image in respect  to the original image>
```

- colorspace/gray

- luminance/gray: changes the colors of the image based on the luminance.

- luminance/sepia: changes the colors of the image based on the luminance

- color/monochrome: changes the colors of the image based on the threshold value. The luminance is calculated and if it is in the specified threshold range it will use the specified color for that range.

- border/color

- border/width

This setting is case sensitive.

Examples

```
Filters[]=geometry/scale=300;300
```

The image will be scaled to 300 pixels width and height.

```
Filters[]=geometry/crop=200;200;155;155
```

The image will be cropped to have a width and height of 200 pixels. The crop left upper starting point on the original image will be at 155 pixels.

5

MIMEType

Summary

Sets the MIME type of the output image.

Usage

```
MIMEType=<MIME type>
```

Description

The MIME type of the output image can be specified if you want a specific format for the alias, if not skip this setting and let the image system decide.

A useful application for this is when using screenshots, normally you don't want to use image/jpeg for screenshot but rather image/png or image/gif.

This setting is case sensitive.

Examples

```
MIMEType=image/png
```

5

Reference

Summary

This setting will tell the system which alias to use as reference image.

Usage

```
Reference=<alias>
```

Description

This setting will tell the system which alias to use as reference image. The reference image is the base of conversion and filtering.

This setting is case sensitive.

Examples

```
Reference=reference
```

5

[original]

Settings regarding the €original€ alias. By defining the €original€ alias and setting some filters or changing the MIME type you can force a conversion of the originally uploaded image. This is useful to save space or make sure the image is in a specific format.

Filters   (page 1922)

Sets filters to be used for the €original€ alias.

MIMEType   (page 1921)

Sets the MIME type of the output image.

5

MIMEType

Summary

Sets the MIME type of the output image.

Usage

```
MIMEType=<MIME type>
```

Description

The MIME type of the output image can be specified if you want a specific format for the alias, if not skip this setting and let the image system decide.
A useful application for this is when using screenshots, normally you don't want to use image/ jpeg for screenshot but rather image/png or image/gif.

This setting is case sensitive.

Examples

```
MIMEType=image/png
```

5

Filters

Summary

Sets filters to be used for the €original€ alias.

Usage

```
Filters[]=<filter>=<filter-parameters>
```

Description

A list of filters to run for the "original" alias, each filter must be supported by the active image converters. The parameter to use is the pixel size of the image. So when you need to use the width, use the horizontal pixel size of the image, and when you need to use the height use the vertical pixel size of the image.

A list of filters:

- geometry/scale: scales the image to the size specified, but maintaining the aspect ratio. The parameters here will be:

```
Filter[]=geometry/scale=<width>;<height>
```

- geometry/scalewidth: scales the image to the widthsize specified, but maintaining the aspect ratio.
  The parameters here will be:

```
Filter[]=geometry/scalewidth=<width>
```

- geometry/scaleheight: scales the image to the heightsize specified, but maintaining the aspect ratio.
  The parameters here will be:

```
Filter[]=geometry/scaleheight=<height>
```

- geometry/scaledownonly: scales the image to the size specified, but maintaining the aspect ratio. The image will not be scaled if the source size is smaller than the destination size.
  The parameters here will be:

```
Filter[]=geometry/scaledownonly=<width>;<height>
```

- geometry/scalewidthdownonly: scales the image to the widthsize specified, but maintaining the aspect ratio. The image will not be scaled if the source size is smaller than the destination size.
  The parameters here will be:

5

```
Filter[]=geometry/scalewidthdownonly=<width>
```

- geometry/scaleheightdownonly: scales the image to the heightsize specified, but maintaining the aspect ratio. The image will not be scaled if the source size is smaller than the destination size.
  The parameters here will be:

```
Filter[]=geometry/scaleheightdownonly=<height>
```

- geometry/scaleexact: scales the image to the exact size specified, but without maintaining the aspect ratio.
  The parameters here will be:

```
Filter[]=geometry/scaleexact=<width>;<height>
```

- geometry/scalepercent: scales the image to the size specified, but without maintaining the aspect ratio.
  The parameters here will be:

```
Filter[]=geometry/scalepercent=<%>
```

- geometry/crop: crops the image to the crop size specified and according to the crop location specified. The parameters here will be

```
Filter[]=geometry/crop=<width after cropping>;<height after  cropping>;
<width location of the cropped image in respect to the  original image>;
<height location of the cropped image in respect  to the original image>
```

- colorspace/gray

- luminance/gray: changes the colors of the image based on the luminance.

- luminance/sepia: changes the colors of the image based on the luminance

- color/monochrome: changes the colors of the image based on the threshold value. The luminance is calculated and if it is in the specified threshold range it will use the specified color for that range.

- border/color

- border/width

This setting is case sensitive.

Examples

```
Filters[]=filter/swirl=210
```

[reference]

This block contains settings regarding the "reference" alias.

Filters   (page 1927)
    Lists the filters to be used for the "reference" alias.

Reference   (page 1926)
    This setting will tell the system which alias to use as reference image for "reference" images.

5

Reference

Summary

This setting will tell the system which alias to use as reference image for "reference" images.

Usage

```
Reference=<alias>
```

Description

This setting will tell the system which alias to use as reference image for "reference" images. The reference image is the base of conversion and filtering.

This setting is case sensitive.

Examples

```
Reference=reference
```

5

Filters

Summary

Lists the filters to be used for the "reference" alias.

Usage

```
Filters[]=<filter>=<filter-parameters>
```

Description

A list of filters to run for the "reference alias", each filter must be supported by the active image converters. The parameter to use is the pixel size of the image. So when you need to use the width, use the horizontal pixel size of the image, and when you need to use the height use the vertical pixel size of the image.

A list of filters:

- geometry/scale: scales the image to the size specified, but maintaining the aspect ratio. The parameters here will be:

```
Filter[]=geometry/scale=<width>;<height>
```

- geometry/scalewidth: scales the image to the widthsize specified, but maintaining the aspect ratio.
  The parameters here will be:

```
Filter[]=geometry/scalewidth=<width>
```

- geometry/scaleheight: scales the image to the heightsize specified, but maintaining the aspect ratio.
  The parameters here will be:

```
Filter[]=geometry/scaleheight=<height>
```

- geometry/scaledownonly: scales the image to the size specified, but maintaining the aspect ratio. The image will not be scaled if the source size is smaller than the destination size.
  The parameters here will be:

```
Filter[]=geometry/scaledownonly=<width>;<height>
```

- geometry/scalewidthdownonly: scales the image to the widthsize specified, but maintaining the aspect ratio. The image will not be scaled if the source size is smaller than the destination size.
  The parameters here will be:

```
Filter[]=geometry/scalewidthdownonly=<width>
```

- geometry/scaleheightdownonly: scales the image to the heightsize specified, but maintaining the aspect ratio. The image will not be scaled if the source size is smaller than the destination size.
  The parameters here will be

```
Filter[]=geometry/scaleheightdownonly=<height>
```

- geometry/scaleexact: scales the image to the exact size specified, but without maintaining the aspect ratio.
  The parameters here will be:

```
Filter[]=geometry/scaleexact=<width>;<height>
```

- geometry/scalepercent: scales the image to the size specified, but without maintaining the aspect ratio.
  The parameters here will be:

```
Filter[]=geometry/scalepercent=<%>
```

- geometry/crop: crops the image to the crop size specified and according to the crop location specified. The parameters here will be:

```
Filter[]=geometry/crop=<width after cropping>;<height after  cropping>;
<width location of the cropped image in respect to the  original image>;
<height location of the cropped image in respect  to the original image>
```

- colorspace/gray

- luminance/gray: changes the colors of the image based on the luminance.

- luminance/sepia: changes the colors of the image based on the luminance

- color/monochrome: changes the colors of the image based on the threshold value. The luminance is calculated and if it is in the specified threshold range it will use the specified color for that range.

- border/color

- border/width

This setting is case sensitive.

Examples

```
Filters[]=geometry/scaledownonly=600;600
```

5

[small]

This block contains settings regarding the "small" alias.

Filters   (page 1931)
> Lists the filters to be used for the "small" alias.

Reference   (page 1934)
> This setting will tell the system which alias to use as reference image for "small" images.

5

Filters

Summary

Lists the filters to be used for the "small" alias.

Usage

```
Filters[]=<filter>=<filter-parameters>
```

Description

A list of filters to run for the "small" alias, each filter must be supported by the active image converters. The parameter to use is the pixel size of the image. So when you need to use the width, use the horizontal pixel size of the image, and when you need to use the height use the vertical pixel size of the image.

A list of filters:

- geometry/scale: scales the image to the size specified, but maintaining the aspect ratio. The parameters here will be:

```
Filter[]=geometry/scale=<width>;<height>
```

- geometry/scalewidth: scales the image to the widthsize specified, but maintaining the aspect ratio.
  The parameters here will be:

```
Filter[]=geometry/scalewidth=<width>
```

- geometry/scaleheight: scales the image to the heightsize specified, but maintaining the aspect ratio.
  The parameters here will be:

```
Filter[]=geometry/scaleheight=<height>
```

- geometry/scaledownonly: scales the image to the size specified, but maintaining the aspect ratio. The image will not be scaled if the source size is smaller than the destination size.
  The parameters here will be:

```
Filter[]=geometry/scaledownonly=<width>;<height>
```

- geometry/scalewidthdownonly: scales the image to the widthsize specified, but maintaining the aspect ratio. The image will not be scaled if the source size is smaller than the destination size.
  The parameters here will be:

```
Filter[]=geometry/scalewidthdownonly=<width>
```

- geometry/scaleheightdownonly: scales the image to the heightsize specified, but maintaining the aspect ratio. The image will not be scaled if the source size is smaller than the destination size.
  The parameters here will be:

```
Filter[]=geometry/scaleheightdownonly=<height>
```

- geometry/scaleexact: scales the image to the exact size specified, but without maintaining the aspect ratio.
  The parameters here will be:

```
Filter[]=geometry/scaleexact=<width>;<height>
```

- geometry/scalepercent: scales the image to the size specified, but without maintaining the aspect ratio.
  The parameters here will be:

```
Filter[]=geometry/scalepercent=<%>
```

- geometry/crop: crops the image to the crop size specified and according to the crop location specified. The parameters here will be:

```
Filter[]=geometry/crop=<width after cropping>;<height after  cropping>;
<width location of the cropped image in respect to the  original image>;
<height location of the cropped image in respect  to the original image>
```

- colorspace/gray

- luminance/gray: changes the colors of the image based on the luminance.

- luminance/sepia: changes the colors of the image based on the luminance

- color/monochrome: changes the colors of the image based on the threshold value. The luminance is calculated and if it is in the specified threshold range it will use the specified color for that range.

- border/color

- border/width

This setting is case sensitive.

Examples

```
Filters[]=geometry/scaledownonly=100;100
```

Reference

Summary

This setting will tell the system which alias to use as reference image for "small" images.

Usage

```
Reference=<alias>
```

Description

This setting will tell the system which alias to use as reference image for "small" images. The reference image is the base of conversion and filtering.

This setting is case sensitive.

Examples

```
Reference=reference
```

5

[tiny]

This block contains settings regarding the "tiny" alias.

Filters   (page 1936)
> Lists the filters for the "tiny" alias.

Reference   (page 1939)
> This setting will tell the system which alias to use as reference image for "tiny" images.

5

Filters

Summary

Lists the filters for the "tiny" alias.

Usage

```
Filters[]=<filter>=<filter-parameters>
```

Description

A list of filters to run by the "tiny" alias, each filter must be supported by the active image converters. The parameter to use is the pixel size of the image. So when you need to use the width, use the horizontal pixel size of the image, and when you need to use the height use the vertical pixel size of the image.

A list of filters:

- geometry/scale: scales the image to the size specified, but maintaining the aspect ratio. The parameters here will be:

```
Filter[]=geometry/scale=<width>;<height>
```

- geometry/scalewidth: scales the image to the widthsize specified, but maintaining the aspect ratio.
  The parameters here will be:

```
Filter[]=geometry/scalewidth=<width>
```

- geometry/scaleheight: scales the image to the heightsize specified, but maintaining the aspect ratio.
  The parameters here will be:

```
Filter[]=geometry/scaleheight=<height>
```

- geometry/scaledownonly: scales the image to the size specified, but maintaining the aspect ratio. The image will not be scaled if the source size is smaller than the destination size.
  The parameters here will be:

```
Filter[]=geometry/scaledownonly=<width>;<height>
```

- geometry/scalewidthdownonly: scales the image to the widthsize specified, but maintaining the aspect ratio. The image will not be scaled if the source size is smaller than the destination size.
  The parameters here will be

```
Filter[]=geometry/scalewidthdownonly=<width>
```

- geometry/scaleheightdownonly: scales the image to the heightsize specified, but maintaining the aspect ratio. The image will not be scaled if the source size is smaller than the destination size.
  The parameters here will be

```
Filter[]=geometry/scaleheightdownonly=<height>
```

- geometry/scaleexact: scales the image to the exact size specified, but without maintaining the aspect ratio.
  The parameters here will be

```
Filter[]=geometry/scaleexact=<width>;<height>
```

- geometry/scalepercent: scales the image to the size specified, but without maintaining the aspect ratio.
  The parameters here will be

```
Filter[]=geometry/scalepercent=<%>
```

- geometry/crop: crops the image to the crop size specified and according to the crop location specified. The parameters here will be

```
Filter[]=geometry/crop=<width after cropping>;<height after  cropping>;
<width location of the cropped image in respect to the  original image>;
<height location of the cropped image in respect  to the original image>
```

- colorspace/gray

- luminance/gray: changes the colors of the image based on the luminance.

- luminance/sepia: changes the colors of the image based on the luminance

- color/monochrome: changes the colors of the image based on the threshold value. The luminance is calculated and if it is in the specified threshold range it will use the specified color for that range.

- border/color

- border/width

This setting is case sensitive.

Examples

```
Filters[]=geometry/scaledownonly=30;30
```

Reference

Summary

This setting will tell the system which alias to use as reference image for "tiny" images.

Usage

```
Reference=<alias>
```

Description

This setting will tell the system which alias to use as reference image for "tiny" images. The reference image is the base of conversion and filtering.

This setting is case sensitive.

Examples

```
Reference=reference
```

5

[medium]

This block contains settings regarding the "medium" alias.

Filters   (page 1941)
>   Lists the filters to be used for the "medium" alias.

Reference   (page 1944)
>   This setting will tell the system which alias to use as reference image for "medium" images

5

Filters

Summary

Lists the filters to be used for the "medium" alias.

Usage

```
Filters[]=<filter>=<filter-parameters>
```

Description

A list of filters to run, each filter must be supported by the active image converters. The parameter to use is the pixel size of the image. So when you need to use the width, use the horizontal pixel size of the image, and when you need to use the height use the vertical pixel size of the image.

A list of filters:

- geometry/scale: scales the image to the size specified, but maintaining the aspect ratio. The parameters here will be:

```
Filter[]=geometry/scale=<width>;<height>
```

- geometry/scalewidth: scales the image to the widthsize specified, but maintaining the aspect ratio.
  The parameters here will be:

```
Filter[]=geometry/scalewidth=<width>
```

- geometry/scaleheight: scales the image to the heightsize specified, but maintaining the aspect ratio.
  The parameters here will be:

```
Filter[]=geometry/scaleheight=<height>
```

- geometry/scaledownonly: scales the image to the size specified, but maintaining the aspect ratio. The image will not be scaled if the source size is smaller than the destination size.
  The parameters here will be:

```
Filter[]=geometry/scaledownonly=<width>;<height>
```

- geometry/scalewidthdownonly: scales the image to the widthsize specified, but maintaining the aspect ratio. The image will not be scaled if the source size is smaller than the destination size.
  The parameters here will be:

5

```
Filter[]=geometry/scalewidthdownonly=<width>
```

- geometry/scaleheightdownonly: scales the image to the heightsize specified, but maintaining the aspect ratio. The image will not be scaled if the source size is smaller than the destination size.
  The parameters here will be:

```
Filter[]=geometry/scaleheightdownonly=<height>
```

- geometry/scaleexact: scales the image to the exact size specified, but without maintaining the aspect ratio.
  The parameters here will be:

```
Filter[]=geometry/scaleexact=<width>;<height>
```

- geometry/scalepercent: scales the image to the size specified, but without maintaining the aspect ratio.
  The parameters here will be:

```
Filter[]=geometry/scalepercent=<%>
```

- geometry/crop: crops the image to the crop size specified and according to the crop location specified. The parameters here will be:

```
Filter[]=geometry/crop=<width after cropping>;<height after  cropping>;
<width location of the cropped image in respect to the  original image>;
<height location of the cropped image in respect  to the original image>
```

- colorspace/gray

- luminance/gray: changes the colors of the image based on the luminance.

- luminance/sepia: changes the colors of the image based on the luminance

- color/monochrome: changes the colors of the image based on the threshold value. The luminance is calculated and if it is in the specified threshold range it will use the specified color for that range.

- border/color

- border/width

This setting is case sensitive.

Examples

```
Filters[]=geometry/scaledownonly=200;200
```

Reference

Summary

This setting will tell the system which alias to use as reference image for "medium" images

Usage

```
Reference=<alias>
```

Description

This setting will tell the system which alias to use as reference image for "medium" images. The reference image is the base of conversion and filtering.

This setting is case sensitive.

Examples

```
Reference=reference
```

5

[large]

This block contains settings regarding the "large" alias.

Filters   (page 1946)
    Lists the filters to be used for the "large" alias.

Reference   (page 1949)
    This setting will tell the system which alias to use as reference image for "large" images.

5

Filters

Summary

Lists the filters to be used for the "large" alias.

Usage

```
Filters[]=<filter>=<filter-parameters>
```

Description

A list of filters to run, each filter must be supported by the active image converters. The parameter to use is the pixel size of the image. So when you need to use the width, use the horizontal pixel size of the image, and when you need to use the height use the vertical pixel size of the image.

A list of filters:

- geometry/scale: scales the image to the size specified, but maintaining the aspect ratio. The parameters here will be:

```
Filter[]=geometry/scale=<width>;<height>
```

- geometry/scalewidth: scales the image to the widthsize specified, but maintaining the aspect ratio.
  The parameters here will be:

```
Filter[]=geometry/scalewidth=<width>
```

- geometry/scaleheight: scales the image to the heightsize specified, but maintaining the aspect ratio.
  The parameters here will be:

```
Filter[]=geometry/scaleheight=<height>
```

- geometry/scaledownonly: scales the image to the size specified, but maintaining the aspect ratio. The image will not be scaled if the source size is smaller than the destination size. The parameters here will be:

```
Filter[]=geometry/scaledownonly=<width>;<height>
```

- geometry/scalewidthdownonly: scales the image to the width size specified, but maintaining the aspect ratio. The image will not be scaled if the source size is smaller than the destination size.
  The parameters here will be:

5

```
Filter[]=geometry/scalewidthdownonly=<width>
```

- geometry/scaleheightdownonly: scales the image to the heightsize specified, but maintaining the aspect ratio. The image will not be scaled if the source size is smaller than the destination size.
  The parameters here will be:

```
Filter[]=geometry/scaleheightdownonly=<height>
```

- geometry/scaleexact: scales the image to the exact size specified, but without maintaining the aspect ratio.
  The parameters here will be:

```
Filter[]=geometry/scaleexact=<width>;<height>
```

- geometry/scalepercent: scales the image to the size specified, but without maintaining the aspect ratio.
  The parameters here will be:

```
Filter[]=geometry/scalepercent=<%>
```

- geometry/crop: crops the image to the crop size specified and according to the crop location specified. The parameters here will be:

```
Filter[]=geometry/crop=<width after cropping>;<height after  cropping>;
<width location of the cropped image in respect to the  original image>;
<height location of the cropped image in respect  to the original image>
```

- colorspace/gray

- luminance/gray: changes the colors of the image based on the luminance.

- luminance/sepia: changes the colors of the image based on the luminance

- color/monochrome: changes the colors of the image based on the threshold value. The luminance is calculated and if it is in the specified threshold range it will use the specified color for that range.

- border/color

- border/width

This setting is case sensitive.

Examples

```
Filters[]=geometry/scaledownonly=300;300
```

5

Reference

Summary

This setting will tell the system which alias to use as reference image for "large" images.

Usage

```
Reference=<alias>
```

Description

This setting will tell the system which alias to use as reference image for "large" images. The reference image is the base of conversion and filtering.

This setting is case sensitive.

Examples

```
Reference=reference
```

5

[rss]

This block contains settings regarding the "rss" alias.

Filters   (page 1951)

    Lists the filters for the "rss" alias.

Reference   (page 1954)

    This setting will tell the system which alias to use as reference image for "rss" images.

5

Filters

Summary

Lists the filters for the "rss" alias.

Usage

```
Filters[]=<filter>=<filter-parameters>
```

Description

A list of filters to run, each filter must be supported by the active image converters. The parameter to use is the pixel size of the image. So when you need to use the width, use the horizontal pixel size of the image, and when you need to use the height use the vertical pixel size of the image.

A list of filters:

- geometry/scale: scales the image to the size specified, but maintaining the aspect ratio. The parameters here will be:

```
Filter[]=geometry/scale=<width>;<height>
```

- geometry/scalewidth: scales the image to the widthsize specified, but maintaining the aspect ratio.
  The parameters here will be:

```
Filter[]=geometry/scalewidth=<width>
```

- geometry/scaleheight: scales the image to the heightsize specified, but maintaining the aspect ratio.
  The parameters here will be:

```
Filter[]=geometry/scaleheight=<height>
```

- geometry/scaledownonly: scales the image to the size specified, but maintaining the aspect ratio. The image will not be scaled if the source size is smaller than the destination size.
  The parameters here will be:

```
Filter[]=geometry/scaledownonly=<width>;<height>
```

- geometry/scalewidthdownonly: scales the image to the widthsize specified, but maintaining the aspect ratio. The image will not be scaled if the source size is smaller than the destination size.
  The parameters here will be:

5

```
Filter[]=geometry/scalewidthdownonly=<width>
```

- geometry/scaleheightdownonly: scales the image to the heightsize specified, but maintaining the aspect ratio. The image will not be scaled if the source size is smaller than the destination size.
  The parameters here will be:

```
Filter[]=geometry/scaleheightdownonly=<height>
```

- geometry/scaleexact: scales the image to the exact size specified, but without maintaining the aspect ratio.
  The parameters here will be:

```
Filter[]=geometry/scaleexact=<width>;<height>
```

- geometry/scalepercent: scales the image to the size specified, but without maintaining the aspect ratio.
  The parameters here will be:

```
Filter[]=geometry/scalepercent=<%>
```

- geometry/crop: crops the image to the crop size specified and according to the crop location specified. The parameters here will be:

```
Filter[]=geometry/crop=<width  after cropping>;<height after cropping>;
<width location of  the cropped image in respect to the original image>;
<height  location of the cropped image in respect to the original image>
```

- colorspace/gray

- luminance/gray: changes the colors of the image based on the luminance.

- luminance/sepia: changes the colors of the image based on the luminance

- color/monochrome: changes the colors of the image based on the threshold value. The luminance is calculated and if it is in the specified threshold range it will use the specified color for that range.

- border/color

- border/width

According to the RSS 1.0 / RSS 2.0 the following specification should be taken into account:

- maximum value for width is 144, default value is 88,

- maximum value for height is 400, default value is 31.

This setting is case sensitive.

Examples

```
Filters[]=geometry/scale=88;31
```

5

Reference

Summary

This setting will tell the system which alias to use as reference image for "rss" images.

Usage

```
Reference=<alias>
```

Description

This setting will tell the system which alias to use as reference image for "rss" images. The reference image is the base of conversion and filtering.

This setting is case sensitive.

Examples

```
Reference=reference
```

5

[rssitem]

This block contains settings regarding the "rssitem" alias.

Filters   (page 1956)

Lists the filters to be used for the "rssitem" alias.

Reference   (page 1959)

This setting will tell the system which alias to use as reference image for "rssitem" images.

5

Filters

Summary

Lists the filters to be used for the "rssitem" alias.

Usage

```
Filters[]=<filter>=<filter-parameters>
```

Description

A list of filters to run, each filter must be supported by the active image converters. The parameter to use is the pixel size of the image. So when you need to use the width, use the horizontal pixel size of the image, and when you need to use the height use the vertical pixel size of the image.

A list of filters:

- geometry/scale: scales the image to the size specified, but maintaining the aspect ratio. The parameters here will be:

```
Filter[]=geometry/scale=<width>;<height>
```

- geometry/scalewidth: scales the image to the widthsize specified, but maintaining the aspect ratio.
  The parameters here will be:

```
Filter[]=geometry/scalewidth=<width>
```

- geometry/scaleheight: scales the image to the heightsize specified, but maintaining the aspect ratio.
  The parameters here will be:

```
Filter[]=geometry/scaleheight=<height>
```

- geometry/scaledownonly: scales the image to the size specified, but maintaining the aspect ratio. The image will not be scaled if the source size is smaller than the destination size.
  The parameters here will be:

```
Filter[]=geometry/scaledownonly=<width>;<height>
```

- geometry/scalewidthdownonly: scales the image to the widthsize specified, but maintaining the aspect ratio. The image will not be scaled if the source size is smaller than the destination size.
  The parameters here will be:

5

```
Filter[]=geometry/scalewidthdownonly=<width>
```

- geometry/scaleheightdownonly: scales the image to the heightsize specified, but maintaining the aspect ratio. The image will not be scaled if the source size is smaller than the destination size.
  The parameters here will be:

```
Filter[]=geometry/scaleheightdownonly=<height>
```

- geometry/scaleexact: scales the image to the exact size specified, but without maintaining the aspect ratio.
  The parameters here will be:

```
Filter[]=geometry/scaleexact=<width>;<height>
```

- geometry/scalepercent: scales the image to the size specified, but without maintaining the aspect ratio.
  The parameters here will be:

```
Filter[]=geometry/scalepercent=<%>
```

- geometry/crop: crops the image to the crop size specified and according to the crop location specified. The parameters here will be/

```
Filter[]=geometry/crop=<width after cropping>;<height after  cropping>;
<width location of the cropped image in respect to the  original image>;
<height location of the cropped image in respect  to the original image>
```

- colorspace/gray

- luminance/gray: changes the colors of the image based on the luminance.

- luminance/sepia: changes the colors of the image based on the luminance

- color/monochrome: changes the colors of the image based on the threshold value. The luminance is calculated and if it is in the specified threshold range it will use the specified color for that range.

- border/color

- border/width

This setting is case sensitive.

Examples

```
Filters[]=geometry/scaledownonly=128;128
```

Reference

Summary

This setting will tell the system which alias to use as reference image for "rssitem" images.

Usage

```
Reference=<alias>
```

Description

This setting will tell the system which alias to use as reference image for "rssitem" images. The reference image is the base of conversion and filtering.

The optional image alias is used in rss feed items if you specify the image attribute as source for description tag, "rss" is used instead if this setting is disabled.

This setting is case sensitive.

Examples

```
reference=reference
```

5

[gray]

This is an example alias which converts the image to grayscale.

Filters (page 1961)
> Lists the filters for the "gray" alias.

Reference (page 1964)
> This setting will tell the system which alias to use as reference image for "gray" images.

5

Filters

Summary

Lists the filters for the "gray" alias.

Usage

```
Filters[]=<filter>=<filter-parameters>
```

Description

A list of filters to run, each filter must be supported by the active image converters. The parameter to use is the pixel size of the image. So when you need to use the width, use the horizontal pixel size of the image, and when you need to use the height use the vertical pixel size of the image.

A list of filters:

- geometry/scale: scales the image to the size specified, but maintaining the aspect ratio. The parameters here will be:

```
Filter[]=geometry/scale=<width>;<height>
```

- geometry/scalewidth: scales the image to the widthsize specified, but maintaining the aspect ratio.
  The parameters here will be:

```
Filter[]=geometry/scalewidth=<width>
```

- geometry/scaleheight: scales the image to the heightsize specified, but maintaining the aspect ratio.
  The parameters here will be:

```
Filter[]=geometry/scaleheight=<height>
```

- geometry/scaledownonly: scales the image to the size specified, but maintaining the aspect ratio. The image will not be scaled if the source size is smaller than the destination size.
  The parameters here will be:

```
Filter[]=geometry/scaledownonly=<width>;<height>
```

- geometry/scalewidthdownonly: scales the image to the widthsize specified, but maintaining the aspect ratio. The image will not be scaled if the source size is smaller than the destination size.
  The parameters here will be:

5

```
Filter[]=geometry/scalewidthdownonly=<width>
```

- geometry/scaleheightdownonly: scales the image to the heightsize specified, but maintaining the aspect ratio. The image will not be scaled if the source size is smaller than the destination size.
  The parameters here will be:

```
Filter[]=geometry/scaleheightdownonly=<height>
```

- geometry/scaleexact: scales the image to the exact size specified, but without maintaining the aspect ratio.
  The parameters here will be:

```
Filter[]=geometry/scaleexact=<width>;<height>
```

- geometry/scalepercent: scales the image to the size specified, but without maintaining the aspect ratio.
  The parameters here will be:

```
Filter[]=geometry/scalepercent=<%>
```

- geometry/crop: crops the image to the crop size specified and according to the crop location specified. The parameters here will be:

```
Filter[]=geometry/crop=<width after cropping>;<height after  cropping>;
<width location of the cropped image in respect to the  original image>;
<height location of the cropped image in respect  to the original image>
```

- colorspace/gray

- luminance/gray: changes the colors of the image based on the luminance.

- luminance/sepia: changes the colors of the image based on the luminance

- color/monochrome: changes the colors of the image based on the threshold value. The luminance is calculated and if it is in the specified threshold range it will use the specified color for that range.

- border/color

- border/width

This setting is case sensitive.

Examples

```
Filters[]=colorspace/gray
```

Reference

Summary

This setting will tell the system which alias to use as reference image for "gray" images.

Usage

```
Reference=<alias>
```

Description

This setting will tell the system which alias to use as reference image for "gray" images. The reference image is the base of conversion and filtering.

This setting is case sensitive.

Examples

```
Reference=large
```

5

[zoom]

This is an example alias which crops the image to specific size and adds a border.

Filters (page 1966)
　　Lists the filters for the "zoom" alias.

Reference (page 1969)
　　This setting will tell the system which alias to use as reference image for "zoom" images.

5

Filters

Summary

Lists the filters for the "zoom" alias.

Usage

```
Filters[]=<filter>=<filter-parameters>
```

Description

A list of filters to run, each filter must be supported by the active image converters. The parameter to use is the pixel size of the image. So when you need to use the width, use the horizontal pixel size of the image, and when you need to use the height use the vertical pixel size of the image.

A list of filters:

- geometry/scale: scales the image to the size specified, but maintaining the aspect ratio. The parameters here will be:

```
Filter[]=geometry/scale=<width>;<height>
```

- geometry/scalewidth: scales the image to the widthsize specified, but maintaining the aspect ratio.
  The parameters here will be:

```
Filter[]=geometry/scalewidth=<width>
```

- geometry/scaleheight: scales the image to the heightsize specified, but maintaining the aspect ratio.
  The parameters here will be:

```
Filter[]=geometry/scaleheight=<height>
```

- geometry/scaledownonly: scales the image to the size specified, but maintaining the aspect ratio. The image will not be scaled if the source size is smaller than the destination size.
  The parameters here will be:

```
Filter[]=geometry/scaledownonly=<width>;<height>
```

- geometry/scalewidthdownonly: scales the image to the widthsize specified, but maintaining the aspect ratio. The image will not be scaled if the source size is smaller than the destination size.
  The parameters here will be:

5

```
Filter[]=geometry/scalewidthdownonly=<width>
```

- geometry/scaleheightdownonly: scales the image to the heightsize specified, but maintaining the aspect ratio. The image will not be scaled if the source size is smaller than the destination size.
  The parameters here will be:

```
Filter[]=geometry/scaleheightdownonly=<height>
```

- geometry/scaleexact: scales the image to the exact size specified, but without maintaining the aspect ratio.
  The parameters here will be:

```
Filter[]=geometry/scaleexact=<width>;<height>
```

- geometry/scalepercent: scales the image to the size specified, but without maintaining the aspect ratio.
  The parameters here will be:

```
Filter[]=geometry/scalepercent=<%>
```

- geometry/crop: crops the image to the crop size specified and according to the crop location specified. The parameters here will be:

```
Filter[]=geometry/crop=<width after cropping>;<height after  cropping>;
<width location of the cropped image in respect to the  original image>;
<height location of the cropped image in respect  to the original image>
```

- colorspace/gray

- luminance/gray: changes the colors of the image based on the luminance.

- luminance/sepia: changes the colors of the image based on the luminance

- color/monochrome: changes the colors of the image based on the threshold value. The luminance is calculated and if it is in the specified threshold range it will use the specified color for that range.

- border/color

- border/width

This setting is case sensitive.

Examples

```
Filters[]=border/color=0;0;0
Filters[]=border=4;4
```

Reference

Summary

This setting will tell the system which alias to use as reference image for "zoom" images.

Usage

```
Reference=<alias>
```

Description

This setting will tell the system which alias to use as reference image for "zoom" images. The reference image is the base of conversion and filtering.

This setting is case sensitive.

Examples

```
Reference=reference
```

5

[MIMETypeSettings]

This block contains settings related to Image MIME type

ConversionRules   (page 1971)
> Sets the global conversion rules for formats that are not listed in [OutputSettings] Allowed-
> OutputFormats.

OverrideList   (page 1973)
> Sets the override settings for MIME types.

Quality   (page 1972)
> Sets the image quality value.

.

5

ConversionRules

Summary

Sets the global conversion rules for formats that are not listed in [OutputSettings] AllowedOutputFormats.

Usage

```
ConversionRules[]=image/<type_to_convert_from>;image/<type_to_convert_to>
```

Description

Sets the global conversion rules for formats that are not listed in [OutputSettings] AllowedOutputFormats. By default most types will be converted to JPEG.
Multiple conversion rules can be set by adding multiple ConversionRules entries for specific image types.

To convert all images to one type, an "*" (asterisk) can be used instead of "image/<type_to_convert_from>". When using an "*" (asterisk) every image type not specified with its own ConversionRules entry will be converted to the same type. See example 1 below.

The ConversionRules only applies to the image formats which are not allowed. Therefore when defining this setting, it is important to keep in mind which image types are defined in the setting AllowedOutputFormat in block [outputsettings], because these types will not be converted according to the conversion rules set here.

This setting is case sensitive.

Examples

Example1

```
ConversionRules[]=image/gif;image/png
ConversionRules[]=image/x-xpixmap;image/png
ConversionRules[]=*;image/jpeg
```

This is the default setting. The last ConversionRules entry defines that every type will be converted to jpeg, with the exception of the above mentioned gif and x-xpixmap images. Gif and x-xpixmap images will be converted to png.

Example2

```
ConversionRules[]=image/jpeg;image/png
```

This setting can be used as fallback if JPEG is disabled as output.

Quality

Summary

Sets the image quality value.

Usage

```
Quality[]=<MIME type>;<integer regarding quality>
```

Description

A list of MIME types and their image quality value. The value varies from MIME type to MIME type.Set JPEG quality from 0 (worst quality, smallest file) to 100 (best quality, biggest file).

This setting is case sensitive.

Examples

```
Quality[]=image/jpeg;75
```

5

OverrideList

Summary

Sets the override settings for MIME types.

Usage

```
OverrideList[]=<name setting>
```

Description

A list of settings which override certain MIME types. The overrides can define allowed and disallowed filters as well as extra filters.

This setting is case sensitive.

Examples

```
OverrideList[]=AnimatedGIF
```

```
OverrideList[]=LayeredPhotoshop
```

5

[AnimatedGIF]

Override for animated GIFs, this is needed to properly handle image/gif files with multiple image frames. It will turn off any geometry filters to leave the images untouched.

AllowedFilters   (page 1976)
    Sets the allowed filters.

DisallowedFilters   (page 1977)
    Sets the filters that are not allowed.

ExtraFilters   (page 1975)
    Lists the extra filters.

Match   (page 1979)
    Lists the matches needed to use this override.

MIMEType   (page 1980)
    Sets this overrides MIME Type.

OverrideMIMEType   (page 1978)
    Sets the override MIME type.

5

ExtraFilters

Summary

Lists the extra filters.

Usage

```
ExtraFilters[]=<filter>
```

Description

A list of extra filters to add for this image.

This setting is case sensitive.

5

AllowedFilters

Summary

Sets the allowed filters.

Usage

```
AllowedFilters[]=<filter>
```

Description

A list of filters which are allowed to be run on this image, wildcards can also be used.If this is supplied only filters that are in this list will be allowed.Note that resizing of animated GIFS is not supported when using GD2, at least not until PHP 5.3.

For more information please visit: http://www.libgd.org/FAQ_PHP

This setting is case sensitive.

Examples

```
AllowedFilters[]=*
```

5

DisallowedFilters

Summary

Sets the filters that are not allowed.

Usage

```
DisallowedFilters[]=<filter>
```

Description

A list of filters which are not allowed to be run on this image, wildcards can also be used.
This setting is case sensitive

Examples

```
DisallowedFilters[]=*
```

5

OverrideMIMEType

Summary

Sets the override MIME type.

Usage

```
OverrideMIMEType=<MIME type>
```

Description

Alternative MIME type for the image, but is currently not used.

This setting is case sensitive.

Examples

```
OverrideMIMEType=animation/gif
```

5

Match

Summary

Lists the matches needed to use this override.

Usage

```
Match[]=
```

Description

A list of matches that needs to be satisfied for this override to be used.

Examples

```
Match[is_animated]=1
```

5

MIMEType

Summary

Sets this overrides MIME Type.

Usage

```
MIMEType=<MIME type>
```

Description

Sets the MIME type which this override works on.

This setting is case sensitive.

Examples

```
MIMEType=image/gif
```

5

[LayeredPhotoshop]

ExtraFilters   (page 1982)
 Sets the extra filters.

MIMEType   (page 1983)
 Lists the MIME types.

5

ExtraFilters

Summary

Sets the extra filters.

Usage

```
ExtraFilters[]=<filter>
```

Description

Sets the extra filters, additional to the ones, for instance mentioned in [myalias] Filters= (page 1915).

This setting is case sensitive.

Examples

```
ExtraFilters[]=flatten
ExtraFilters[]=resize=200x
```

5

MIMEType

Summary

Lists the MIME types.

Usage

```
MIMEType=<MIME type>
```

Description

Lists the MIME types.

This setting is case sensitive.

Examples

```
MIMEType=application/x-photoshop
```

5

[ShellSettings]

This block contains image converter settings.

ConvertPath   (page 1985)
    Sets the path to the converter.

5

ConvertPath

Summary

Sets the path to the converter.

Usage

```
ConvertPath=<path to convert>
```

Description

Sets the path to the converter.

This setting is case sensitive.

5

[ImageConverterSettings]

This block contains Image converter settings.

ImageConverters   (page 1987)
    Lists the converters to use.

5

ImageConverters

Summary

Lists the converters to use.

Usage

```
ImageConverters[]=<name custom converter>
```

Description

Lists the converters to use. Each entry is the name of the converter which is defined in the ini block. You can use ImageMagick handlers and GD handlers by default.

The order of the converters is important since the first applicable converter will be used. The last handler is usually a fallback converter which is able to take care of most formats.

The list can also contain converters which cannot work on the installed server, the converter should check this and tell the imagesystem to not use it.

For any other converter make sure you have a INI section with the same name as your converter.

This setting is case sensitive.

Examples

```
ImageConverters[]=MyCustomConverter
```

5

[MyCustomConverter]

ExtensionList   (page 1991)
DEPRECATED (not needed or supported in 4.1 for autoload based handler loading)

ImageConverters   (page 1993)
Lists the converters to use.

ImageHandlerAlias   (page 1990)
DEPRECATED (not needed/supported in 4.1 for autoload based handler loading)

LockTimeout   (page 1989)
Sets the time lockout.

RepositoryList   (page 1992)
DEPRECATED (not needed or supported in 4.1 for autoload based handler loading)

5

LockTimeout

Summary

Sets the time lockout.

Usage

```
LockTimeout=<time in seconds>
```

Description

Sets the waiting time for processes in queue to get an exclusive writing lock for writing a specific image alias.

If a process spends more than the time set here waiting, the process holding the lock will be ended and the waiting process will continue, generating the image alias.

Examples

```
LockTimeout=60
```

5

ImageHandlerAlias

Summary

DEPRECATED (not needed/supported in 4.1 for autoload based handler loading)

Usage

```
ImageHandlerAlias[]=<alias>
```

5

ExtensionList

Summary

DEPRECATED (not needed or supported in 4.1 for autoload based handler loading)

Usage

```
ExtensionList[]=<name extension>
```

RepositoryList

Summary

DEPRECATED (not needed or supported in 4.1 for autoload based handler loading)

Usage

```
RepositoryList[]=<path to repository>
```

Examples

```
RepositoryList[]=lib/ezimage/classes
```

5

ImageConverters

Summary

Lists the converters to use.

Usage

```
ImageConverters[]=<name image converter>
```

Description

Lists the converters to use. Each entry is the name of the converter which is defined in the ini block. You can use ImageMagick handlers and GD handlers by default.The order of the converters is important since the first applicable converter will be used.

The last handler is usually a fallback converter which is able to take care of most formats.

The list can also contain converters which cannot work on the installed server, the converter should check this and tell the imagesystem to not use it.

For any other converter make sure you have a INI section with the same name as your converter.

This setting is case sensitive.

Examples

```
ImageConverters[]=ImageMagick
```

```
ImageConverters[]=GD
```

5

[GD]

Filters   (page 1995)
> Lists the filters to be used by the GD image converter.

Handler   (page 1998)
> Sets the PHP Handler to use.

IsEnabled   (page 1999)
> Sets if this Handler should be used.

Name   (page 2000)
> Sets the name of the converter.

5

Filters

Summary

Lists the filters to be used by the GD image converter.

Usage

```
Filters[]=<filter>=<filter-parameter>
```

Description

A list of filters to run, each filter must be supported by the active image converter, in this case GD.
The parameter to use is the pixel size of the image. So when you need to use the width, use the horizontal pixel size of the image, and when you need to use the height use the vertical pixel size of the image.
The following filters are supported by GD:

- geometry/scale: scales the image to the size specified, but maintaining the aspect ratio. The parameters here will be:

```
Filter[]=geometry/scale=<width>;<height>
```

- geometry/scalewidth: scales the image to the widthsize specified, but maintaining the aspect ratio.
  The parameters here will be:

```
Filter[]=geometry/scalewidth=<width>
```

- geometry/scaleheight: scales the image to the heightsize specified, but maintaining the aspect ratio.
  The parameters here will be:

```
Filter[]=geometry/scaleheight=<height>
```

- geometry/scaledownonly: scales the image to the size specified, but maintaining the aspect ratio. The image will not be scaled if the source size is smaller than the destination size.
  The parameters here will be:

```
Filter[]=geometry/scaledownonly=<width>;<height>
```

- geometry/scalewidthdownonly: scales the image to the widthsize specified, but maintaining the aspect ratio. The image will not be scaled if the source size is smaller than the destination size.
  The parameters here will be:

5

```
Filter[]=geometry/scalewidthdownonly=<width>
```

- geometry/scaleheightdownonly: scales the image to the heightsize specified, but maintaining the aspect ratio. The image will not be scaled if the source size is smaller than the destination size.
  The parameters here will be:

```
Filter[]=geometry/scaleheightdownonly=<height>
```

- geometry/scaleexact: scales the image to the exact size specified, but without maintaining the aspect ratio.
  The parameters here will be:

```
Filter[]=geometry/scaleexact=<width>;<height>
```

- geometry/scalepercent: scales the image to the size specified, but without maintaining the aspect ratio.
  The parameters here will be:

```
Filter[]=geometry/scalepercent=<%>
```

- geometry/crop: crops the image to the crop size specified and according to the crop location specified. The parameters here will be:

```
Filter[]=geometry/crop=<width after cropping>;<height after  cropping>;
<width location of the cropped image in respect to the  original image>;
<height location of the cropped image in respect  to the original image>
```

- colorspace/gray

- luminance/gray: changes the colors of the image based on the luminance.

- luminance/sepia: changes the colors of the image based on the luminance

- color/monochrome: changes the colors of the image based on the threshold value. The luminance is calculated and if it is in the specified threshold range it will use the specified color for that range.

- border/color

- border/width

This setting is case sensitive.

Examples

```
Filters[]=geometry/scale=300;300
```

Handler

Summary

Sets the PHP Handler to use.

Usage

```
Handler=<class name>
```

Description

Sets which PHP handler to use for the conversion, ezimage normally uses the ImageGD extension in PHP.

This setting is case sensitive.

Examples

```
Handler=eZImageGDFactory
```

5

IsEnabled

Summary

Sets if this Handler should be used.

Usage

```
IsEnabled= true | false
```

Description

If set to €true€ this handler will be used.

The setup wizard will turn this to false if GD is not found on the system.

This setting is case sensitive.

Examples

```
IsEnabled=true
```

This is the default setting.

5

Name

Summary

Sets the name of the converter.

Usage

```
Name=<converter name>
```

Description

Sets the converter name.

This setting is case sensitive.

Examples

```
Name=GD
```

5

[ImageMagick]

This block contains settings related to the image converter ImageMagick.

Executable   (page 2014)
>   Sets the name of the executable

ExecutablePath   (page 2015)
>   Sets the path to the executable.

Executable{OS}   (page 2013)
>   Sets the executable that must be used for a specific operating system.

Filters   (page 2003)
>   Sets the filters supported by ImageMagick.

FrameRangeParameters   (page 2006)
>   Set frame range parameters to handle PSD files with ImageMagick.

Handler   (page 2016)
>   Sets the PHP Handler that needs to be used.

InputMIMEList   (page 2009)
>   List the Input MIME types

IsEnabled   (page 2017)
>   Sets if the handler set in this configuration group should be used.

MIMETagMap   (page 2002)
>   Sets the MIME type tag.

Name   (page 2018)
>   Sets the name of the converter.

OutputMIMEList   (page 2008)
>   MIME types that the converter can generate

PostParameters   (page 2011)
>   List the parameters that must be run last.

PreParameters   (page 2012)
>   List the parameters that must be run first.

QualityParameters   (page 2007)
>   Sets the quality parameters of MIME Type

UseTypeTag   (page 2010)
>   Sets whether the destination name should be prefixed with a TAG name.

5

MIMETagMap

Summary

Sets the MIME type tag.

Usage

```
MIMETagMap[]=<MIME type>;<tag>
```

Description

Maps a MIME type to a tag name which is prepended to the destination name for ImageMagick. This setting is case sensitive.

Examples

```
MIMETagMap[]=image/x-xpixmap;XPM

MIMETagMap[]=image/png;PNG

MIMETagMap[]=image/jpeg;JPEG

MIMETagMap[]=image/bmp;BMP

MIMETagMap[]=image/x-portable-bitmap;PBM

MIMETagMap[]=image/svg+xml;SVG

MIMETagMap[]=text/plain;TEXT
```

5

Filters

Summary

Sets the filters supported by ImageMagick.

Usage

```
Filters[]=<filter>=<filter-parameter>
```

Description

A list of filters to run which ImageMagick supports. Add all that are needed.The same filter names are used as the filters for the aliases and MIME types, but optional parameters are added.

The parameter to use is the pixel size of the image. So when you need to use the width, use the horizontal pixel size of the image, and when you need to use the height use the vertical pixel size of the image.

The following filters are supported by ImageMagick by default:

- geometry/scale: scales the image to the size specified, but maintaining the aspect ratio.

- geometry/scalewidth: scales the image to the widthsize specified, but maintaining the aspect ratio.

- geometry/scaleheight: scales the image to the heightsize specified, but maintaining the aspect ratio.

- geometry/scaledownonly: scales the image to the size specified, but maintaining the aspect ratio. The image will not be scaled if the source size is smaller than the destination size.

- geometry/scalewidthdownonly: scales the image to the widthsize specified, but maintaining the aspect ratio. The image will not be scaled if the source size is smaller than the destination size.

- geometry/scaleheightdownonly: scales the image to the heightsize specified, but maintaining the aspect ratio. The image will not be scaled if the source size is smaller than the destination size.

- geometry/scaleexact: scales the image to the exact size specified, but without maintaining the aspect ratio.

- geometry/scalepercent: scales the image to the size specified, but without maintaining the aspect ratio.

- geometry/crop: crops the image to the crop size specified and according to the crop location specified.

5

- colorspace/gray

- luminance/gray: changes the colors of the image based on the luminance.

- luminance/sepia: changes the colors of the image based on the luminance

- color/monochrome: changes the colors of the image based on the threshold value. The luminance is calculated and if it is in the specified threshold range it will use the specified color for that range.

- border/color

- border/width

The parameters to use here are:

- (scale%) : the height and width will both be scaled by the specified percentage.

- (scale-x%)x(scale-y%) : the width and height are individually scaled by specified percentages. (Only one % symbol needed.)

- (width) : the width is given, and the height will automagically be selected to preserve aspect ratio.

- x(height) : the height is given, and the width will automagically be selected to preserve aspect ratio.

- (width)x(height) : the maximum values of the width and height are given, but the aspect ratio is preserved.

- (width)x(height)ˆ : the minimum values of the width and height are given, but the aspect ratio is preserved.

- (width)x(height)! : the width and height are emphatically given, and the original aspect ratio is ignored.

- (width)x(height)> : this will change the dimensions the same as "widthxheight", but only if both the image dimension exceeds the specified dimension.

- (width)x(height)< : this will change the dimensions the same as "widhtxheight", but only if both the image dimensions exceed the specified dimensions

- (area)@ : will resize the image to have the specified area in pixels. Aspect ratio is preserved.

This setting is case sensitive.

Examples

5

```
Filters[]=geometry/scale=-geometry  %1x%2

Filters[]=geometry/scalewidth=-geometry  %1

Filters[]=geometry/scaleheight=-geometry x%1

Filters[]=geometry/crop=-crop  %1x%2+%3+%4

Filters[]=filter/noise=-noise  %1

Filters[]=filter/swirl=-swirl %1

Filters[]=flatten=-flatten

Filters[]=resize=-resize  %1
```

5

FrameRangeParameters

Summary

Set frame range parameters to handle PSD files with ImageMagick.

Usage

```
FrameRangeParameters[]=<MIME type>;<option>
```

Description

This setting will provide the possibility to postpend options to the source file's filename. This behavior is needed to address a specific layer in PSD files.

This array needs a semicolon separated value which should be build like this:

```
<MIME type>;<option>
```

This setting is case sensitive.

Examples

```
FrameRangeParameters[]=application/x-photoshop;[0]
```

5

QualityParameters

Summary

Sets the quality parameters of MIME Type

Usage

```
QualityParameters[]=<MIME type>;<quality parameters>
```

Description

A list of MIME types and their quality parameters.

For more information on the ImageMagick quality parameters visit this link: http://www.imagemagick.org/script/command-line-options.php#quality

This setting is case sensitive.

Examples

```
QualityParameters[]=image/jpeg;-quality %1
```

5

OutputMIMEList

Summary

MIME types that the converter can generate

Usage

```
OutputMIMEList[]=<MIME type>
```

Description

MIME types that the converter can generate. Wildcards can be used as value.

This setting is case sensitive.

Examples

```
OutputMIMEList[]=*
```

5

InputMIMEList

Summary

List the Input MIME types

Usage

```
InputMIMEList[]=<MIME type>
```

Description

MIME types that the converter can get as input. Wildcards can be used as value.

This setting is case sensitive.

Examples

```
InputMIMEList[]=*
```

5

UseTypeTag

Summary

Sets whether the destination name should be prefixed with a TAG name.

Usage

```
UseTypeTag=<tag>
```

Description

Sets whether the destination name should be prefixed with a TAG name (see €MIMETagMap€). The variable contains the separator between the TAG and the file name, which is needed for ImageMagick to provide proper conversions of some formats.

Examples

```
UseTypeTag=:
```

In this case a ":" will be used.

5

PostParameters

Summary

List the parameters that must be run last.

Usage

```
PostParameters=<name parameter>
```

Description

Lists the parameters that must be run after all file names and filters.

This setting is case sensitive.

5

PreParameters

Summary

List the parameters that must be run first.

Usage

```
PreParameters=<name parameter>
```

Description

Lists the parameters that must be run before any file names and filters.

This setting is case sensitive.

Executable{OS}

Summary

Sets the executable that must be used for a specific operating system.

Usage

```
ExecutableWin32=convertim | convert

ExecutableMac=convertim | convert

ExecutableUnix=convertim | convert
```

Description

€ExecutableWin32€ sets the name of the executable for Windows. €ExecutableMac€ sets the name of the executable for the Mac specific converter. €ExecutableUnix€ sets the name of the executable for the Unix/Linux specific converter.

Possible values:

- convert

- convertim

This setting is case sensitive.

Examples

```
ExecutableWin32=convertim

ExecutableMac=convert

ExecutableUnix=convert
```

5

Executable

Summary

Sets the name of the executable

Usage

```
Executable=<name executable>
```

Description

Sets the name of the executable.

Options: convert or convertim

This setting is case sensitive.

Examples

```
Executable=convert
```

5

ExecutablePath

Summary

Sets the path to the executable.

Usage

```
ExecutablePath=<path to executable>
```

Description

Sets the path to the executable. This path can be left empty, then the global path will be used. This setting is case sensitive.

Handler

Summary

Sets the PHP Handler that needs to be used.

Usage

```
Handler=<name handler>
```

Description

Sets which PHP handler to use for the conversion. eZ imageshell normally uses the generic shell handler which creates a command line and executes it with the system.

This setting is case sensitive.

Examples

```
Handler=eZImageShellFactory
```

5

IsEnabled

Summary

Sets if the handler set in this configuration group should be used.

Usage

```
IsEnabled= true | false
```

Description

If set to €true€ the handler set in this configuration group will be used. The setup wizard will turn this to false, if ImageMagick is not found on the system.

This setting is case sensitive.

Examples

```
IsEnabled=true
```

This is the default setting.

5

Name

Summary

Sets the name of the converter.

Usage

```
Name=<name converter>
```

Description

Sets the name of the converter.

This setting is case sensitive.

Examples

```
Name=ImageMagick
```

[AnalyzerSettings]

This block contains image analysis settings.

AnalyzerMIMEList   (page 2020)
    Sets formats which need extra analysis.

ExtensionList   (page 2023)
    Sets the extensions which have analyzers.

ImageAnalyzerAlias   (page 2021)
    Lists aliases for the analyzer

ImageAnalyzers   (page 2022)
    Lists the active analyzers

RepositoryList   (page 2024)
    Sets where to look for analyzers.

5

AnalyzerMIMEList

Summary

Sets formats which need extra analysis.

Usage

```
AnalyzerMIMEList[]=<MIME type>
```

Description

Lists formats which need extra analysis.

This setting is case sensitive.

Examples

```
AnalyzerMIMEList[]=image/gif
```

This sets that GIF images must be analyzed to check for animated GIFs, since animated GIFs are hard to convert to other formats.

5

ImageAnalyzerAlias

Summary

Lists aliases for the analyzer

Usage

```
ImageAnalyzerAlias[]=<alias name>
```

Description

Lists aliases for the analyzer, which can be used to redirect certain analyzers to your own analyzer. This setting is case sensitive.

ImageAnalyzers

Summary

Lists the active analyzers

Usage

```
ImageAnalyzers[]=<tag active analyzer>
```

Description

Lists the active analyzers.

This setting is case sensitive.

Examples

```
ImageAnalyzers[]=GIF

ImageAnalyzers[]=EXIF
```

5

ExtensionList

Summary

Sets the extensions which have analyzers.

Usage

```
ExtensionList[]=<extension name>
```

Description

A list of extensions which have analyzers. In the extension directory the system will look for analyzers in the image analyzer sub directory defined in the setting €RepositoryList=€.

This setting is case sensitive.

5

RepositoryList

Summary

Sets where to look for analyzers.

Usage

```
RepositoryList[]=<path to analyzers>
```

Description

Sets where to look for analyzers.

This setting is case sensitive.

Examples

```
RepositoryList[]=lib/ezimage/classes
```

[GIFAnalyzer]

GIF analyzer, figures out details about GIF images.This block is primarily used to determine GIF animation.

Handler   (page 2027)
> Sets the handler to use.

MIMEList   (page 2026)
> Lists the MIME types.

5

MIMEList

Summary

Lists the MIME types.

Usage

```
MIMEList[]=<MIME type>
```

Description

Lists the MIME types.

This setting is case sensitive.

Examples

```
MIMEList[]=image/gif
```

5

Handler

Summary

Sets the handler to use.

Usage

Handler=<name handler>

Description

Sets the handler to use.

This setting is case sensitive.

Examples

Handler=ezgif

5

[EXIFAnalyzer]

This block contains settings related to the analyzer for EXIF data, currently supports JPEG and TIFF.

Handler (page 2030)
    Sets the handler to use

MIMEList (page 2029)
    Lists the MIME types

5

MIMEList

Summary

Lists the MIME types

Usage

```
MIMEList[]=<MIME type>
```

Description

Lists the MIME types.

This setting is case sensitive.

Examples

```
MIMEList[]=image/jpeg
```

```
MIMEList[]=image/tiff
```

5

Handler

Summary

Sets the handler to use

Usage

Handler=<name handler>

Description

Sets the handler to use.

This setting is case sensitive.

Examples

Handler=ezexif

5

5.14.24   layout.ini

5

5.14.25   ldap.ini

The configuration blocks are documented in the following sections:

\begin{itemize}   \item{}   [LDAPSettings]\refer{labelaeb5b1aaff20baa1a5a0a451b25c74f4}
\end{itemize}

5

[LDAPSettings]

KeepGroupAssignment (page 2067)
    Sets if LDAP users will automatically moving back to the group they are configured to be
    placed in at login.

LDAPBaseDn (page 2041)
    Sets the Base DN for the directory

LDAPBindPassword (page 2043)
    Sets the password for LDAP server.

LDAPBindUser (page 2042)
    Sets the user name for LDAP server.

LDAPCreateMissingGroups (page 2068)
    Sets whether or not groups that are not found in eZ Publish should be created.

LDAPDebugTrace (page 2035)
    This setting enables tracing the LDAP login.

LDAPEmailAttribute (page 2064)
    Sets the LDAP attribute for email.

LDAPEmailEmptyAttributeSuffix (page 2065)
    Sets suffix to be used to create email address.

LDAPEnabled (page 2038)
    Sets the use of the LDAP server.

LDAPEqualSign (page 2045)
    Sets replacement for equal sign.

LDAPFirstNameAttribute (page 2061)
    Sets the LDAP attribute for the first name.

LDAPFirstNameIsCommonName (page 2062)
    Permits the extraction on the First Name from the common name.

LDAPFollowReferrals (page 2037)
    Sets if the LDAP library will automatically follow referrals returned by LDAP servers.

LDAPGroupBaseDn (page 2053)
    Sets the Base LDAP DN.

LDAPGroupClass (page 2054)
    Sets the class of LDAP group objects.

LDAPGroupDescriptionAttribute (page 2057)
    Sets the attribute of the LDAP group that contains the description of the group.

5

5

LDAPDebugTrace

Summary

This setting enables tracing the LDAP login.

Usage

```
LDAPDebugTrace= disabled | enabled
```

Description

This setting enables tracing the LDAP login, which outputs extensive debug information which can be used during setup.
Keep in mind that login name and passwords will be logged to logfiles or outputted if DebugOutput settings are enabled in site.ini. For this reason it is best not keep this setting enabled on production setup.
This setting is case sensitive.

Examples

```
LDAPDebugTrace=disabled
```

No debug information will be shown during setup.

5

LDAPVersion

Summary

Set LDAP version number.

Usage

```
LDAPVersion=<integer>
```

Description

Set LDAP version number. This should be 2 or 3.

Examples

```
LDAPVersion=2
```

This is the default setting.

5

LDAPFollowReferrals

Summary

Sets if the LDAP library will automatically follow referrals returned by LDAP servers.

Usage

```
LDAPFollowReferrals= 0 | 1 (boolean value)
```

Description

This setting determines if the LDAP library will automatically follow referrals returned by LDAP servers.
Boolean values are used: 0 (zero) to disable and 1 to enable.

This setting is available since eZ Publish 4.1 to enhance the LDAP functionality.

Examples

```
LDAPFollowReferrals=0
```

The LDAP library will not automatically follow the referrals returned by LDAP servers.
This is the default setting.

5

LDAPEnabled

Summary

Sets the use of the LDAP server.

Usage

```
LDAPEnabled= true | false
```

Description

Allows you to enable/disable the use of the LDAP server. The LDAP server can be used as a repository for user and group information. If LDAP is enabled the LDAP ligin handler should be set here:

site.ini [UserSettings] LoginHandler.

For more information on the workings of LDAP please visit http://ez.no/doc/ez_publish/ technical_manual/4_x/features/ldap_login_handler. (page 408)

Possible settings are true to enable the use of the LDAP server and false to disable the use of the LDAP server.

This setting is case sensitive.

Examples

```
LDAPEnabled=true
```

The LDAP server will be used. This is the default setting.

5

LDAPServer

Summary

Sets the LDAPhost

Usage

```
LDAPServer=<LDAP host>
```

Description

This setting sets the LDAP host. If the setting LDAPEnabled is set to true, the LDAP server set here will be used as a repository for user and group information.

Values can be the name of the LDAP host or its IP-address.

This setting is case sensitive.

Examples

```
LDAPServer=ldaphost
```

Instead of using the name of the LDAP server, as used here, the IP address could also be used.

5

LDAPPort

Summary

Sets the portnumber for LDAP

Usage

```
LDAPPort=<port number>
```

Description

Sets the portnumber for LDAP. The default port is 389.

Examples

```
LDAPPort=389
```

This is the default port.

5

LDAPBaseDn

Summary

Sets the Base DN for the directory

Usage

```
LDAPBaseDn=<name LDAP base DN>
```

Description

This setting specifies the base DN for the directory. The more specific your base DN is, the less data the handler has to search through before finding it.

Please note that all DNs 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 set in setting LDAPEqualSign in block [LDAPSettings], which set by default to '--'.
This setting is left empty by default.

This setting is case sensitive.

5

LDAPBindUser

Summary

Sets the user name for LDAP server.

Usage

```
LDAPBindUser=<user name>
```

Description

If the server does not allow anonymous bind, specify the username for the bind here. A bind is a LDAP term for authenticating against a server, an anonymous bind is a bind without username, password or DN.
Note that the bind user has nothing to do with the credentials of the user logging in. Instead, it is the username that must be supplied in order to read the LDAP user object we want to authenticate against.

This setting is case sensitive.

Examples

```
LDAPBindUser=admin
```

5

LDAPBindPassword

Summary

Sets the password for LDAP server.

Usage

LDAPBindPassword=<password>

Description

If the server does not allow anonymous bind, specify the actual password for the bind here. A bind is a LDAP term for authenticating against a server, an anonymous bind is a bind without username, password or DN.
Note that the Bind password has nothing to do with the credentials of the user logging in. Instead, it is the password that must be supplied in order to read the LDAP user object we want to authenticate against.

This setting is case sensitive.

Examples

LDAPBindPassword=LDAP

5

LDAPSearchScope

Summary

Sets the type of search performed by the LDAP handler.

Usage

```
LDAPSearchScope= sub | one | base
```

Description

The LDAP handler performs a search for the user object on the LDAP server to verify the login
and password provided by the user.
There are three kinds of searches: sub, one or base. This setting defines the type of search to use.

- sub: This is the default search, it will search the whole subtree below the base DN

- one: This will search one level only, the level immediately below the base DN

- base: This will search for only one object, which means that the base DN must equal the
  DN of the object you want. This 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.

This setting is case sensitive.

Examples

```
LDAPSearchScope=sub
```

The LDAP handler will search the whole subtree below the base DN for the user object on the
LDAP server to verify the login and password provided by the user.
This is the default setting.

5

LDAPEqualSign

Summary

Sets replacement for equal sign.

Usage

```
LDAPEqualSign=<replacement for "=">
```

Description

Due to a limitation in the ini file format, equal signs can not be used in values in ldap.ini. This setting sets the equal sign which is used to replace "=" when specifing such things as LDAPBaseDn or LDAPSearchFilters in block [LDAPSettings].

Examples

```
LDAPEqualSign=--
```

This is the default setting.

5

LDAPSearchFilters

Summary

Sets extra search filters.

Usage

```
LDAPSearchFilters[]=<filter>
```

Description

This setting allows you to add an extra search requirement, it allows you to limit the set of data to search through. It is recommended to uncomment it if you don't need it.

This setting permits you to search for a particular class using for example 'objectClass--inetOrgPerson', or a particular attribute using 'myattribute--myvalue'. Keep in mind that ini files do not allow for "=", a replacement for the equal sign must be set in the setting LDAPEqualSign in the block [LDAPSettings]. By default "--" is used.

It is possible to configure multiple filters, which will be combined and used together.

The filter for the user who is logging in (for example 'uid--johndoe') will automatically be added by the login handler.

This setting is case sensitive.

Examples

```
LDAPSearchFilters[]=objectClass--inetOrgPerson
```

The LDAP search will be done in this particular class.

5

LDAPLoginAttribute

Summary

Sets the LDAP attribute for login.

Usage

```
LDAPLoginAttribute=<string value
```

Description

Sets the LDAP attribute for login. This is the attribute of the LDAP servers user object definition, that holds the data needed to use as login (most likely the username) in eZ Publish.
This implies that the attribute must hold a string that uniquely identifies a user, most commonly the user ID.
Any string value is possible, depending on the LDAP server setup, but "uid" (User ID) is recommended.

This setting is case sensitive.

Examples

```
LDAPLoginAttribute=uid
```

The User ID is used to identify the user at login.
This is the default setting.

5

LDAPUserGroupType

Summary

Sets the LDAP user group type.

Usage

```
LDAPUserGroupType=id | name
```

Description

Sets the LDAP user group type.

Possible values are id or name
name: the name of the user group will be used as value in the setting LDAPUserGroup in block
[LDAPSettings].
id: the numeric ID of the user group will be used as value in the setting LDAPUserGroup in block
[LDAPSettings].

This setting is case sensitive.

Examples

```
LDAPUserGroupType=id
```

The numeric ID of the user group should be used as value in the setting LDAPUserGroup in block
[LDAPSettings].

5

LDAPUserGroup

Summary

Sets the default location to store LDAP users.

Usage

```
LDAPUserGroup[]=<content object id or group name>
```

Description

Sets the default location to store LDAP users. Depending on the type defined in the LDAPUser-GroupType setting in block [LDAPSettings] the value can be the content object id or the group name for LDAP user group.
This setting is case sensitive.

Examples

```
LDAPUserGroup[]=354
```

The ID number 354 can be used if LDAPUserGroupType setting in block [LDAPSettings] is set to id.

5

LDAPGroupRootNodeId

Summary

This sets the group mapping settings.

Usage

```
LDAPGroupRootNodeId=<node id>
```

Description

This sets the group mapping settings. This setting sets the location of groups created by the LDAP login handler. The location must be the node ID of the group node that should be the parent of all created groups.
If this setting is left blank the root node ID where the LDAP groups are created, node id 5, will be used.
This setting is case sensitive.

Examples

```
LDAPGroupRootNodeId=5
```

The groups created by the LDAP login handler will be located at node 5.
This is the default setting.

5

LDAPGroupMappingType

Summary

Sets the LDAP group mapping type.

Usage

```
LDAPGroupMappingType= UseGroupAttribute | SimpleMapping | GetGroupsTree
```

Description

This setting sets the LDAP group mapping type.
Possible values are: UseGroupAttribute, SimpleMapping an GetGroupsTree

- UseGroupAttribute: this is the old style group assignment which uses the LDAPUser-GroupAttribute setting.
  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 using this value also check the setting LDAPCreateMissingGroups in block [LDAPSettings].
  When using this value in combination with the LDAPUserGroupAttributeType setting set to 'dn', you should also check the setting LDAPGroupNameAttribute in block [LDAPSettings]

- SimpleMapping: this uses the LDAPUserGroupMap array for name-to-name group mapping.
  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 also set LDAPGroupClass, LDAPGroupNameAttribute, LDAPGroupMemberAttribute, and LDAPUserGroupMap in block [LDAPSettings]

- GetGroupsTree: this uses the automatic name-to-name group mapping, which creates groups as needed.
  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 in block [LDAPSettings].

For more information and examples on LDAP group mapping type please visit the chapter LDAP group mapping type (page 411)

This setting is case sensitive

Examples

5

```
LDAPGroupMappingType=UseGroupAttribute
```

This is the default setting.

5

LDAPGroupBaseDn

Summary

Sets the Base LDAP DN.

Usage

```
LDAPGroupBaseDn=<base DN>
```

Description

Sets the Base LDAP DN which should be used to fetch user group objects from LDAP.
By setting this to the base DN of your groups tree, you will limit the amount of data to search
through.

This setting is case sensitive.

5

LDAPGroupClass

Summary

Sets the class of LDAP group objects.

Usage

```
LDAPGroupClass=<name of the class of LDAP group objects>
```

Description

Sets the class of LDAP group objects.
This setting is case sensitive.

Examples

```
LDAPGroupClass=exampleGroupDAClassName
```

This is the default setting.

5

LDAPGroupNameAttribute

Summary

Sets the attribute of the LDAP group that contains the name of the group.

Usage

```
LDAPGroupNameAttribute=<attribute that contains the name of the group>
```

Description

Sets the attribute of the LDAP group that contains the name of the group.
Keep in mind that this setting is required when the LDAPGroupMappingType setting in block [LDAPSettings] is set to GetGroupsTree or SimpleMapping. This is also the case when the setting is set to UseGroupAttribute and the LDAPUserGroupAttributeType setting in the same block is set to dn.
This setting is case sensitive.

Possible values are

- cn: common name

- fn: first name

- sn: last name

Examples

```
LDAPGroupNameAttribute=cn
```

This is the default setting.

5

LDAPGroupMemberAttribute

Summary

Sets the attribute of the LDAP group that contains the names of the users that are members.

Usage

```
LDAPGroupMemberAttribute=<the attribute of the LDAP group that contains the
names of the users that are members>
```

Description

This sets the attribute of the LDAP group that contains the names of the users that are members. It is an attribute of the LDAP user that is used to obtain the groups of which the user(group) belongs to.
This setting is required when 'LDAPGroupMappingType' is set to 'GetGroupsTree' or 'SimpleMapping'.
This setting is case sensitive.

Examples

```
LDAPGroupMemberAttribute=member
```

This is the default setting.

5

LDAPGroupDescriptionAttribute

Summary

Sets the attribute of the LDAP group that contains the description of the group.

Usage

```
LDAPGroupDescriptionAttribute=<attribute with group description>
```

Description

This is the attribute of the LDAP servers user group object definition, that holds a textual, human readable description of the user group. This is an optional setting, it 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, yet it allows the eZ Publish groups to have different names than the corresponding LDAP groups.
Any string value is possible, depending on the LDAP server setup. If the server has no such attribute, it is recommended to leave the setting empty and the attribute will be left empty in eZ Publish.
This setting is case sensitive.

Examples

```
LDAPGroupDescriptionAttribute=groupdescription
```

This is the default setting.

5

LDAPUserGroupMap

Summary

Maps group names from the LDAP group to eZ Publish usergroups.

Usage

```
LDAPUserGroupMap[<LDAP group name>]=<eZ group name>
```

Description

This sets the group names map, from the LDAP group to eZ Publish usergroups. This setting is used when the setting LDAPGroupMappingType in block [LDAPSettings] is set to SimpleMapping. By default this setting is left empty.
This setting is case sensitive.

Examples

```
LDAPUserGroupMap[myldapgroup]=myezgroup
```

This will map the LDAP group "myldapgroup" to the eZ Publish group "myezgroup".

5

LDAPUserGroupAttributeType:

Summary

Sets the LDAP attribute type for user groups.

Usage

```
LDAPUserGroupAttributeType= name | id | dn
```

Description

This setting sets the LDAP attribute type for user groups. This setting is used in combination with the setting LDAPUserGroupAttribute in block [LDAPSettings].
Possible values are:

- name: the name of the group will be used. If 'name' is used, then the LDAPUserGroupAttribute attribute must contain the name of an existing eZ Publish group.

- id: the numeric ID of the group will be used. If 'id' is used, then the LDAPUserGroupAttribute attribute must contain an ID, for example 42, that will match an existing eZ Publish group with the name 'LDAP 42'.

- dn: Starting with eZ Publish 4.3, the DN of the group will be used. If 'DN' is used, then the LDAPUserGroupAttribute attribute must contain a DN referring to the group the user belongs to.
  Note that when the 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 instead of containing multiple DNs (like how LDAP attributes are normally used).

This setting is case sensitive.

Examples

```
LDAPUserGroupAttributeType=name
```

The value used in the LDAPUserGroupAttribute setting must contain the name of an existing eZ Publish group.
This is the default setting.

5

LDAPUserGroupAttribute

Summary

Sets the LDAP attribute for user group.

Usage

```
LDAPUserGroupAttribute=<the name, ID or DN of the user group>
```

Description

Sets the LDAP attribute for user group. If specified, LDAP users will be saved under the same group as in LDAP server.
This value depends on the value set in LDAPUserGroupAttributeType in block [LDAPSettings]. If LDAPUserGroupAttributeType is set to:
name: the attribute holding name of the user group must be used as value for LDAPUserGroupAttribute.
id: the attribute holding nummeric ID of the user group must be used as value for LDAPUserGroupAttribute.
dn: the attribute holding DN of the user group must be used as value for LDAPUserGroupAttribute.
This setting is case sensitive.

Examples

Example 1

```
LDAPUserGroupAttribute=employeetype
```

Employeetype can be used as value if LDAPUserGroupAttributeType is set to name.

Example 2

```
LDAPUserGroupAttribute=42
```

42 can be used if LDAPUserGroupAttributeType is set to id.

LDAPFirstNameAttribute

Summary

Sets the LDAP attribute for the first name.

Usage

```
LDAPFirstNameAttribute=fn | givename
```

Description

When the group assignment is complete, the user object is stored in eZ Publish.
This setting defines the LDAP attribute for the first name. By default it is "givenname" which implies that the first name is stored directly in LDAP.
If the first name is not stored directly in LDAP, but it can be extracted from the common name (cn) you should set LDAPFirstNameAttribute to 'cn' and set LDAPFirstNameIsCommonName in block [LDAPSettings] to true.
The first name will then be extracted from the common name, minus the last name. (This feature was added in version 4.1 beta 1.)
This setting is case sensitive.

Examples

```
LDAPFirstNameAttribute=givename
```

This is the default setting.

5

LDAPFirstNameIsCommonName

Summary

Permits the extraction on the First Name from the common name.

Usage

```
LDAPFirstNameIsCommonName=true | false
```

Description

This setting allows you to use the common name (cn) for the first name, by removing the last name (sn) from common name.
This setting is useful if the first name is not stored directly in LDAP, but it can be extracted from the common name (cn). If you enable this setting by setting it to true, you need to set LDAPFirstNameAttribute in block [LDAPSettings] to 'cn'.
The first name will then be extracted from the common name. (This feature was added in version 4.1 beta 1.)
You can disable this setting, by using the value false.
This setting is case sensitive.

Examples

```
LDAPFirstNameIsCommonName=false
```

The first name will not be extracted from the common name.

5

LDAPLastNameAttribute

Summary

Sets the LDAP attribute for the last name.

Usage

```
LDAPLastNameAttribute= sn | givename
```

Description

When the group assignment is complete, the user object is stored in eZ Publish.
This setting defines the LDAP attribute for the last name. By default it is "sn".
This setting is case sensitive.

Examples

```
LDAPLastNameAttribute=sn
```

This is the default value.

5

LDAPEmailAttribute

Summary

Sets the LDAP attribute for email.

Usage

```
LDAPEmailAttribute=<LDAP mail attribute>
```

Description

This setting sets the LDAP attribute for email. By default the value is set to mail.
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 LDAPEmailEmptyAttribute-Suffix.
This setting is case sensitive.

Examples

```
LDAPEmailAttribute=mail
```

This is the default setting.

5

LDAPEmailEmptyAttributeSuffix

Summary

Sets suffix to be used to create email address.

Usage

```
LDAPEmailEmptyAttributeSuffix=<email suffix>
```

Description

This setting can be used when the LDAP server does not return mail. It will create one using the login name and the email suffix defined here.

This setting is available since eZ Publish 4.1 to enhance the LDAP functionality.

By default this setting is left empty.

This setting is case sensitive.

Examples

```
LDAPEmailEmptyAttributeSuffix=@ez.no
```

When '@ez.no' is set here, then a user with user ID 'joepublish' will get the email 'joepublish@ez.no'.

Utf8Encoding

Summary

Sets the use of LDAP encoding.

Usage

```
Utf8Encoding=true | false
```

Description

This setting allows you to enable or disable if LDAP encoding is utf-8 or not.

Possible values are true and false. True sets LDAP encoding to UTF-8.

This setting is case sensitive.

Examples

```
Utf8Encoding=false
```

UTF-8 LDAP encoding will not be used.

5

KeepGroupAssignment

Summary

Sets if LDAP users will automatically moving back to the group they are configured to be placed in at login.

Usage

```
KeepGroupAssignment=enabled | disabled
```

Description

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 prefer the eZ Publish group assignments to stay the same, set KeepGroupAssignment to enabled. This setting will allow you to move LDAP users to a different group without them automatically moving back to the group they are configured to be placed in, when the user logs in again.
This setting is case sensitive.

Examples

```
KeepGroupAssignment=disabled
```

This setting will atomatically move LDAP users back to the group they are configured to be placed in, when the user logs in again.

5

LDAPCreateMissingGroups

Summary

Sets whether or not groups that are not found in eZ Publish should be created.

Usage

```
LDAPCreateMissingGroups= disabled | enabled
```

Description

When the setting LDAPGroupMappingType in block [LDAPSettings] is set to value UseGroupAttribute, LDAPCreateMissingGroups sets whether or not groups that are not found in eZ Publish, should be created.
Enabling it will allow groups to be created, disabling is will prevent groups to be greated.

This setting is case sensitive.

Examples

```
LDAPCreateMissingGroups=disabled
```

None existing groups will not automatically be created.
This is the default setting.

5

5.14.26   logfile.ini

The configuration blocks are documented in the following sections:

- [AccessLogFileSettings] (page 2070)

5

[AccessLogFileSettings]

LogFileName   (page 2072)
     Sets the name of the Apache log file.

SitePrefix   (page 2071)
     Sets which prefixes that are used in the Apache log of requested URLs.

StorageDir   (page 2073)
     Sets the directory where eZ Publish will look for the Apache log file.

5

SitePrefix

Summary

Sets which prefixes that are used in the Apache log of requested URLs.

Usage

```
SitePrefix[]=siteaccess1
SitePrefix[]=siteaccess2
...
```

Description

If the "updateviewcount.php" cronjob script is run periodically, it will analyze the requested URLs stored in the Apache log file. When the URI access method is used, the requested URLs will contain the name of the target siteaccess. For example, the following URL will tell eZ Publish to use the "example_admin" siteaccess: http://www.mysite.com/example_admin. If you wish these prefixes to be removed before analyzing the log file, specify the names of the siteaccesses using the "SitePrefix" directive.

5

LogFileName

Summary

Sets the name of the Apache log file.

Usage

```
LogFileName=file_name
```

Description

The node view count (number of page views) in the database isn't updated by the module/view system when the different nodes are being shown. To make the view counter work, you must periodically run the "updateviewcount.php" cronjob script. The script will extract information from the Apache log file and store it in the database. Please note that this directive can only be used to set the name of the log file. The path to the file must be specified using the StorageDir (page 2073) directive.

Examples

Example 1

```
StorageDir=/var/log/httpd/
LogFileName=access_log
```

This configuration will tell eZ Publish to use the "access_log" file inside the "/var/log/httpd/" directory.

5

StorageDir

Summary

Sets the directory where eZ Publish will look for the Apache log file.

Usage

```
StorageDir=directory_name
```

Description

The node view count (number of page views) in the database isn't updated by the module/view system when the different nodes are being shown. To make the view counter work, you must periodically run the "updateviewcount.php" cronjob script. The script will extract information from the Apache log file and store it in the database. Please note that this directive can only be used to specify the path to the log file. Use the "LogFileName" (page 2072) directive to set the actual name of the file.

Examples

Example 1

```
StorageDir=/var/log/httpd/
LogFileName=access_log
```

This configuration will tell eZ Publish to use the "access_log" file inside the "/var/log/httpd/" directory.

5

5.14.27 menu.ini

The configuration blocks are documented in the following sections:

\begin{itemize}   \item{}   [DoubleTop]\refer{label13f215d10c942f3b9cba026659e1b48b}
\item{}   [LeftMenuSettings]\refer{label641333180ec38fef77a713716c77b2a8}
\item{}   [Leftmenu_content]\refer{labelf4a4d1a68b484e4a6def2998f2470cda}
\item{}   [Leftmenu_design]\refer{label348f7ed90a69fb7790c74ffe127775d1}
\item{}   [Leftmenu_media]\refer{labelf59a159a625ed64dfe1f854c96269014}
\item{}   [Leftmenu_my]\refer{label29ad5be6b97a10c84ff879d6f5f8993d}
\item{}   [Leftmenu_setup]\refer{labelb9ed6069afc22806a5d7369a01fdccf7}
\item{}   [Leftmenu_shop]\refer{label61fc1bd544a4135bbdae47b9fae9c0f8}
\item{}   [Leftmenu_user]\refer{labeled61a72fd8528b171ad9a76451b1fad1}
\item{}   [LeftOnly]\refer{labele0b1c22b11bcfbbc8faacebaa07dea14}   \item{}
[LeftTop]\refer{labeld8fbbb58d905549e316079a2a04c2f83}   \item{}
[MenuContentSettings]\refer{label2e0012ee495f0c32846cf519d47d5094}
\item{}   [MenuSettings]\refer{labeldef073b93f32d75d96c985928209aefa}
\item{}   [NavigationPart]\refer{labela82e25618a5b49cc4ac55e3d0d6f3f8a}
\item{}   [SelectedMenu]\refer{labelfd91f0df9faa86feb38786ff3d4f9c19}   \item{}
[TopAdminMenu]\refer{label2e3bc44e8451f957465f5703ad5ebecb}   \item{}
[Topmenu_content]\refer{labeld7dc3e7cfc27319df3ddc7a7ed231693}   \item{}
[Topmenu_dashboard]\refer{label90274b9e00574abcec55958b06f9639e}   \item{}
[Topmenu_design]\refer{label09380b73257ec44a6472574d24ffdec1}   \item{}
[Topmenu_media]\refer{label4cbc7f117b09d7b67fdc8014bce476f4}   \item{}
[Topmenu_setup]\refer{label0586a04f2306efce8ee07c96e7fba975}   \item{}
[Topmenu_shop]\refer{labelb131c9e70178e57edc41d56603e8d911}   \item{}
[Topmenu_users]\refer{labelf996be54ca48c217d4d03517db6374c6}   \item{}
[TopOnly]\refer{labelb1e66f68f2a307e0463f824e7e273945} \end{itemize}

5

[LeftMenuSettings]

This block controls the display of the left menu for the administration interface.

MenuWidth   (page 2076)
     This setting allows you to define the width setting used for the different menu sizes.

5

MenuWidth

Summary

This setting allows you to define the width setting used for the different menu sizes.

Usage

```
MenuWidth[<menu size>]=<width in em>
```

Description

This setting allows you to define the width setting, which can be used to setup the widths for the different menu sizes: small, medium and large. The measurement unit is em.
This setting is case sensitive.

Examples

```
MenuWidth[small]=13
MenuWidth[medium]=19
MenuWidth[large]=25
```

This is the default setting.

5

[Leftmenu_user]

LinkNames   (page 2079)
    Sets link name.

Links   (page 2078)
    List of link URLs and index.

Name   (page 2081)
    Sets the translated name value for this menu

PolicyList_{Links_index}   (page 2080)
    Sets the policy list per link, using the index from the "Links" array as part of the name.

5

Links

Summary

List of link URLs and index.

Usage

```
Links[<Links_index>]=<path>
```

Description

List of link URLs and index. The index is used for LinkNames and PolicyList settings.
This setting is case sensitive.

Examples

```
Links[roles_and_policies]=role/list
```

This is the default setting.

LinkNames

Summary

Sets link name.

Usage

```
LinkNames[<Links_index>]=<link name>
```

Description

Sets link name.
If left empty the default will be used. Keep in mind that if you wish to have the name translatable you will need to add a string hidden in a template somewhere with a i18n() call or a dummy PHP file with the following in it:

```
ezi18n( 'design/admin/parts/example/
menu', 'Example menu', 'Leftmenu  name for example menu' )
```

This setting is case sensitive.

5

PolicyList_{Links_index}

Summary

Sets the policy list per link, using the index from the "Links" array as part of the name.

Usage

```
PolicyList_{Links_index}[]=<node_id> | <module>/<function>
```

Description

Sets the policy list per link, using the index from the "Links" array as part of the name.
The possible values are either the "<node_ID>" or according to the syntax "<module>/<function>".
If this policy returns false, the menu item will be disabled, which means no link will be shown.
This setting is case sensitive.

Examples

```
PolicyList[]=role/list
```

This is the default setting.

5

Name

Summary

Sets the translated name value for this menu

Usage

```
Name=<name>
```

Description

Sets the translated name value for this menu, therefore it is recommended to use eng-GB here.
If left empty the default will be used. Keep in mind that if you wish to have the name translatable
you will need to add a string hidden in a template somewhere with a i18n() call or a dummy
PHP file with the following in it:

```
ezi18n( 'design/admin/parts/example/
menu', 'Example menu', 'Leftmenu  name for example menu' )
```

This will cause the ezlupdate to include it in the .ts file.
Note: For custom menu's on custom tabs you can also pass a hash of translations to ini_menu.tpl.
It will then use strings in that, if name of menu or link index exists in the hash. For more
information please visit "design/admin2/templates/parts/setup/menu.tpl" as example.
This setting is case sensitive.

Examples

```
Name=access_controll
```

This is the default setting.

5

[Leftmenu_shop]

LinkNames   (page)
  Sets link name.

Links   (page)
  List of link URLs and index.

Name   (page)
  Sets the translated name value for this menu

5

Links

Summary

List of link URLs and index.

Usage

```
Links[<Links_index>]=<path>
```

Description

List of link URLs and index. The index is used for LinkNames and PolicyList settings.
This setting is case sensitive.

Examples

```
Links[customers]=shop/customerlist
Links[discounts]=shop/discountgroup
Links[orders]=shop/orderlist
Links[archive]=shop/archivelist
Links[order_status]=shop/status
Links[product_statistics]=shop/statistics
Links[vat_types]=shop/vattype
Links[vat_rules]=shop/vatrules
Links[product_categories]=shop/productcategories
Links[currencies]=shop/currencylist
Links[preferred_currency]=shop/preferredcurrency
Links[products_overview]=shop/productsoverview
```

These are the default settings.

5

LinkNames

Summary

Sets link name.

Usage

```
LinkNames[<Links_index>]=<link name>
```

Description

Sets link name.
If left empty the default will be used. Keep in mind that if you wish to have the name translatable you will need to add a string hidden in a template somewhere with a i18n() call or a dummy PHP file with the following in it:

```
ezi18n( 'design/admin/parts/example/
menu', 'Example menu', 'Leftmenu  name for example menu' )
```

This setting is case sensitive.

5

Name

Summary

Sets the translated name value for this menu

Usage

```
Name=<name>
```

Description

Sets the translated name value for this menu, therefore it is recommended to use eng-GB here.
If left empty the default will be used. Keep in mind that if you wish to have the name translatable
you will need to add a string hidden in a template somewhere with a i18n() call or a dummy
PHP file with the following in it:

```
ezi18n( 'design/admin/parts/example/
menu', 'Example menu', 'Leftmenu  name for example menu' )
```

This will cause the ezlupdate to include it in the .ts file.
Note: For custom menu's on custom tabs you can also pass a hash of translations to ini_menu.tpl.
It will then use strings in that, if name of menu or link index exists in the hash. For more
information please visit "design/admin2/templates/parts/setup/menu.tpl" as example.
This setting is case sensitive.

Examples

```
Name=shop
```

"shop" is the default value for this setting.

5

[Leftmenu_setup]

LinkNames   (page 2088)
>   Sets link name.

Links   (page 2089)
>   List of link URLs and index.

Name   (page 2090)
>   Sets the translated name value for this menu

PolicyList_{Links_index}   (page 2087)
>   Sets the policy list per link, using the index from the "Links" array as part of the name.

5

PolicyList_{Links_index}

Summary

Sets the policy list per link, using the index from the "Links" array as part of the name.

Usage

```
PolicyList_{Links_index}[]=<node_id> | <module>/<function>
```

Description

Sets the policy list per link, using the index from the "Links" array as part of the name.
The possible values are either the "<node_ID>" or according to the syntax "<module>/
<function>".
If this policy returns false, the menu item will be disabled, which means no link will be shown.
This setting is case sensitive.

Examples

```
PolicyList_cache[]=setup/managecache
PolicyList_classes[]=class/grouplist
PolicyList_collected[]=infocollector/read
PolicyList_extensions[]=setup/setup
PolicyList_ini[]=settings/view
PolicyList_languages[]=content/translations
PolicyList_menu_management[]=visual/menuconfig
PolicyList_pdf_export[]=pdf/edit
PolicyList_packages[]=package/list
PolicyList_rad[]=setup/setup
PolicyList_rss[]=rss/edit
PolicyList_search_statistics[]=search/stats
PolicyList_sessions[]=setup/administrate
PolicyList_states[]=state/administrate
PolicyList_system_information[]=setup/system_info
PolicyList_upgrade_check[]=setup/setup
PolicyList_triggers[]=trigger/list
PolicyList_url_management[]=url/list
PolicyList_url_translator[]=content/urltranslator
PolicyList_url_wildcards[]=content/urltranslator
PolicyList_workflows[]=workflow/grouplist
PolicyList_workflow_processes[]=workflow/processlist
```

These are the default settings.

5

LinkNames

Summary

Sets link name.

Usage

```
LinkNames[<Links_index>]=<link name>
```

Description

Sets link name.
If left empty the default will be used. Keep in mind that if you wish to have the name translatable you will need to add a string hidden in a template somewhere with a i18n() call or a dummy PHP file with the following in it:

```
ezi18n( 'design/admin/parts/example/
menu', 'Example menu', 'Leftmenu  name for example menu' )
```

This setting is case sensitive.

5

Links

Summary

List of link URLs and index.

Usage

```
Links[<Links_index>]=<path>
```

Description

List of link URLs and index. The index is used for LinkNames and PolicyList settings.
This setting is case sensitive.

Examples

```
Links[cache]=setup/cache
Links[classes]=class/grouplist
Links[collected]=infocollector/overview
Links[extensions]=setup/extensions
Links[ini]=settings/view
Links[languages]=content/translations
Links[menu_management]=visual/menuconfig
Links[pdf_export]=pdf/list
Links[packages]=package/list
Links[rad]=setup/rad
Links[rss]=rss/list
Links[search_statistics]=search/stats
Links[sections]=section/list
Links[sessions]=setup/session
Links[states]=state/groups
Links[system_information]=setup/info
Links[upgrade_check]=setup/systemupgrade
Links[triggers]=trigger/list
Links[url_management]=url/list
Links[url_translator]=content/urltranslator
Links[url_wildcards]=content/urlwildcards
Links[workflows]=workflow/grouplist
Links[workflow_processes]=workflow/processlist
```

These are the default settings.

5

Name

Summary

Sets the translated name value for this menu

Usage

```
Name=<name>
```

Description

Sets the translated name value for this menu, therefore it is recommended to use eng-GB here.
If left empty the default will be used. Keep in mind that if you wish to have the name translatable
you will need to add a string hidden in a template somewhere with a i18n() call or a dummy
PHP file with the following in it:

```
ezi18n( 'design/admin/parts/example/
menu', 'Example menu', 'Leftmenu  name for example menu' )
```

This will cause the ezlupdate to include it in the .ts file.
Note: For custom menu's on custom tabs you can also pass a hash of translations to ini_menu.tpl.
It will then use strings in that, if name of menu or link index exists in the hash. For more
information please visit "design/admin2/templates/parts/setup/menu.tpl" as example.
This setting is case sensitive.

Examples

```
Name=setup
```

"setup" is the default value for this setting.

5

[Leftmenu_my]

LinkNames   (page 2093)
   Sets link name.

Links   (page 2094)
   List of link URLs and index.

Name   (page 2095)
   Sets the translated name value for this menu

PolicyList_{Links_index}   (page 2092)
   Sets the policy list per link, using the index from the "Links" array as part of the name.

5

PolicyList_{Links_index}

Summary

Sets the policy list per link, using the index from the "Links" array as part of the name.

Usage

```
PolicyList_{Links_index}[]=<node_id> | <module>/<function>
```

Description

Sets the policy list per link, using the index from the "Links" array as part of the name.
The possible values are either the "<node_ID>" or according to the syntax "<module>/<function>".
If this policy returns false, the menu item will be disabled, which means no link will be shown.
This setting is case sensitive.

Examples

```
PolicyList_change_password[]=user/password
PolicyList_collaboration[]=collaboration/view
PolicyList_edit_profile[]=user/selfedit
PolicyList_my_bookmarks[]=content/bookmark
PolicyList_my_drafts[]=content/edit
PolicyList_my_notifications[]=notification/use
PolicyList_my_pending[]=content/pendinglist
```

These are the default settings.

5

LinkNames

Summary

Sets link name.

Usage

```
LinkNames[<Links_index>]=<link name>
```

Description

Sets link name.
If left empty the default will be used. Keep in mind that if you wish to have the name translatable
you will need to add a string hidden in a template somewhere with a i18n() call or a dummy
PHP file with the following in it:

```
ezi18n( 'design/admin/parts/example/
menu', 'Example menu', 'Leftmenu  name for example menu' )
```

This setting is case sensitive.

5

Links

Summary

List of link URLs and index.

Usage

```
Links[<Links_index>]=<path>
```

Description

List of link URLs and index. The index is used for LinkNames and PolicyList settings.
This setting is case sensitive.

Examples

```
Links[change_password]=user/password
Links[collaboration]=collaboration/view/summary
Links[dashboard]=content/dashboard
Links[edit_profile]=user/edit/(action)/edit
Links[my_bookmarks]=content/bookmark
Links[my_drafts]=content/draft
Links[my_notifications]=notification/settings
Links[my_pending]=content/pendinglist
```

This is the default setting.

5

Name

Summary

Sets the translated name value for this menu

Usage

```
Name=<name>
```

Description

Sets the translated name value for this menu, therefore it is recommended to use eng-GB here.
If left empty the default will be used. Keep in mind that if you wish to have the name translatable
you will need to add a string hidden in a template somewhere with a i18n() call or a dummy
PHP file with the following in it:

```
ezi18n( 'design/admin/parts/example/
menu', 'Example menu', 'Leftmenu  name for example menu' )
```

This will cause the ezlupdate to include it in the .ts file.
Note: For custom menu's on custom tabs you can also pass a hash of translations to ini_menu.tpl.
It will then use strings in that, if name of menu or link index exists in the hash. For more
information please visit "design/admin2/templates/parts/setup/menu.tpl" as example.
This setting is case sensitive.

Examples

```
Name=dashboard
```

This is the default setting.

5

[Leftmenu_media]

LinkNames (page 2097)
   Sets link name.

Links (page 2098)
   List of link URLs and index.

Name (page 2099)
   Sets the translated name value for this menu.

5

LinkNames

Summary

Sets link name.

Usage

```
LinkNames[<Links_index>]=<link name>
```

Description

Sets link name.
If left empty the default will be used. Keep in mind that if you wish to have the name translatable
you will need to add a string hidden in a template somewhere with a i18n() call or a dummy
PHP file with the following in it:

```
ezi18n( 'design/admin/parts/example/
menu', 'Example menu', 'Leftmenu  name for example menu' )
```

This setting is case sensitive.

5

Links

Summary

List of link URLs and index.

Usage

Links[<Links_index>]=<path>

Description

List of link URLs and index. The index is used for LinkNames and PolicyList settings.
This setting is case sensitive.

Examples

Links[]=

5

Name

Summary

Sets the translated name value for this menu.

Usage

```
Name=<name>
```

Description

Sets the translated name value for this menu, therefore it is recommended to use eng-GB here.
If left empty the default will be used. Keep in mind that if you wish to have the name translatable
you will need to add a string hidden in a template somewhere with a i18n() call or a dummy
PHP file with the following in it:

```
ezi18n( 'design/admin/parts/example/
menu', 'Example menu', 'Leftmenu  name for example menu' )
```

This will cause the ezlupdate to include it in the .ts file.
Note: For custom menu's on custom tabs you can also pass a hash of translations to ini_menu.tpl.
It will then use strings in that, if name of menu or link index exists in the hash. For more
information please visit "design/admin2/templates/parts/setup/menu.tpl" as example.
This setting is case sensitive.

Examples

```
Name=Media Menu
```

"Media Menu" is the default setting.

5

[Leftmenu_design]

\begin{description} \item[LinkNames]\refer{labela0bd9ff052ec02d8ebecf42c93e424b4}\ \\ Sets link name.

\item[Links]\refer{labelad441d3eca58d7d5eb04382c2e5c98b6}\ \\ List of link URLs and index.

\item[Name]\refer{labela4dfad67141a639895159070074f9cf6}\ \\ Sets the translated name value for this menu.

\item[PolicyList_\{Links_index\}]\refer{label60bf8a1ca6b81deb690b4b76cf175c2e}\ \\ Sets the policy list per link, using the index from the "Links" array as part of the name.

\end{description}

5

Name

Summary

Sets the translated name value for this menu.

Usage

```
Name=<name>
```

Description

Sets the translated name value for this menu, therefore it is recommended to use eng-GB here. If left empty the default will be used. Keep in mind that if you wish to have the name translatable you will need to add a string hidden in a template somewhere with a i18n() call or a dummy PHP file with the following in it:

```
ezi18n( 'design/admin/parts/example/
menu', 'Example menu', 'Leftmenu  name for example menu' )
```

This will cause the ezlupdate to include it in the .ts file.

Note: For custom menu's on custom tabs you can also pass a hash of translations to ini_menu.tpl. It will then use strings in that, if name of menu or link index exists in the hash. For more information please visit "design/admin2/templates/parts/setup/menu.tpl" as example.

This setting is case sensitive.

Examples

```
Name=design
```

This is the default setting.

5

Links

Summary

List of link URLs and index.

Usage

```
Links[<Links_index>]=<path>
```

Description

List of link URLs and index. The index is used for LinkNames and PolicyList settings.

This setting is case sensitive.

Examples

```
Links[look_and_feel]=content/edit/54
Links[toolbar_management]=visual/toolbarlist
```

This is the default setting.

5

LinkNames

Summary

Sets link name.

Usage

```
LinkNames[<Links_index>]=<link name>
```

Description

Sets link name.
If left empty the default will be used. Keep in mind that if you wish to have the name translatable you will need to add a string hidden in a template somewhere with a i18n() call or a dummy PHP file with the following in it:

```
ezi18n( 'design/admin/parts/example/
menu', 'Example menu', 'Leftmenu  name for example menu' )
```

This setting is case sensitive.

Examples

```
LinkNames[]=
```

Left empty by default.

5

PolicyList_{Links_index}

Summary

Sets the policy list per link, using the index from the "Links" array as part of the name.

Usage

```
PolicyList_{Links_index}[]=<node_id> | <module>/<function>
```

Description

Sets the policy list per link, using the index from the "Links" array as part of the name.
The possible values are either the "<node_ID>" or according to the syntax "<module>/
<function>".
If this policy returns false, the menu item will be disabled, which means no link will be shown.
This setting is case sensitive.

Examples

```
PolicyList_look_and_feel[]=56
PolicyList_toolbar_management[]=visual/toolbarlist
```

This is the default setting.

5

[Leftmenu_content]

Adding this block will add a left menu to content area.

LinkNames   (page 2107)
  Sets link name.

Links   (page 2108)
  List of link URLs and index

Name   (page 2109)
  Sets the translated name value for this menu.

PolicyList_{Links_index}   (page 2106)
  Sets the policy list per link, using the index from the "Links" array as part of the name.

5

PolicyList_{Links_index}

Summary

Sets the policy list per link, using the index from the "Links" array as part of the name.

Usage

```
PolicyList_{Links_index}[]=<node_id> | <module>/<function>
```

Description

Sets the policy list per link, using the index from the "Links" array as part of the name.
The possible values are either the "<node_ID>" or according to the syntax "<module>/<function>".
If this policy returns false, the menu item will be disabled, which means no link will be shown.
This setting is case sensitive.

Examples

```
PolicyList_languages[]=content/translations
```

This is the default setting.

5

LinkNames

Summary

Sets link name.

Usage

```
LinkNames[<Links_index>]=<link name>
```

Description

If left empty the default will be used. Keep in mind that if you wish to have the name translatable you will need to add a string hidden in a template somewhere with a i18n() call or a dummy PHP file with the following in it:

```
ezi18n( 'design/admin/parts/example/
menu', 'Example menu', 'Leftmenu  name for example menu'
```

This setting is case sensitive.

Examples

```
LinkNames[languages]=Languages
```

This is the default setting.

5

Links

Summary

List of link URLs and index

Usage

Links[<Links_index>]=<path>

Description

List of link URLs and index. The index is used for LinkNames and PolicyList settings.
This setting is case sensitive.

Examples

Links[languages]=content/translations

This is the default setting.

5

Name

Summary

Sets the translated name value for this menu.

Usage

```
Name=<name>
```

Description

Sets the translated name value for this menu, therefore it is recommended to use eng-GB here.
If left empty the default will be used. Keep in mind that if you wish to have the name translatable
you will need to add a string hidden in a template somewhere with a i18n() call or a dummy
PHP file with the following in it:

```
ezi18n( 'design/admin/parts/example/
menu', 'Example menu', 'Leftmenu  name for example menu' )
```

This will cause the ezlupdate to include it in the .ts file.
Note: For custom menu's on custom tabs you can also pass a hash of translations to ini_menu.tpl.
It will then use strings in that, if name of menu or link index exists in the hash. For more
information please visit "design/admin2/templates/parts/setup/menu.tpl" as example.
This setting is case sensitive.

Examples

```
Name=Content Menu
```

This is the default setting.

5

[Topmenu_dashboard]

Enabled   (page 2112)
    Sets whether or not the menu item is clickable in current UI context.

Name   (page 2115)
    Sets which name to show on the menu item.

NavigationPartIdentifier   (page 2116)
    Sets the navigation part of the identifier of the menu item.

Shown   (page 2111)
    Sets whether or not the menu item is shown in current UI context.

Tooltip   (page 2114)
    Sets the tooltip which is shown for the menu item when the cursor hovers over the menu item.

URL   (page 2113)
    Sets the view which an item points to in the different context.

5

Shown

Summary

Sets whether or not the menu item is shown in current UI context.

Usage

```
Shown[<ui_context>]=true | false
```

Description

Sets whether or not the menu item is shown in current UI context
The User Interface context can be default, browse or edit.

True enables showing the menu item and false disables showing the menu item.

This setting is case sensitive.

Examples

```
Shown[default]=true
Shown[navigation]=true
Shown[browse]=true
```

This is the default setting. It means that this menu item is shown in the default mode, navigation mode and browse mode.

5

Enabled

Summary

Sets whether or not the menu item is clickable in current UI context.

Usage

```
Enabled[<ui_context>]=true | false
```

Description

Sets whether or not the menu item is clickable in current UI context.
The User Interface context can be "default", "browse", "edit",....
This setting is case sensitive.

Examples

```
Enabled[default]=true
 Enabled[browse]=false
 Enabled[edit]=false
```

This is the default setting. It implies that this menu item is not clickable in Edit- or Browse mode.

5

URL

Summary

Sets the view which an item points to in the different context.

Usage

URL[<ui_context>]=<path>

Description

Sets the view which an item points to in the different context.
The User Interface context can be default, browse or edit.
This setting is case sensitive.

Examples

URL[default]=content/dashboard

This is the default setting.

5

Tooltip

Summary

Sets the tooltip which is shown for the menu item when the cursor hovers over the menu item.

Usage

```
Tooltip=<message that appears as tooltip for this menu>
```

Description

Sets the tooltip which is shown for the menu item when the cursor hovers over the menu item. If the setting is left empty the default value is used. But keep in mind the when you don't use the default value the tooltip can be translated.
This setting is case sensitive.

Examples

```
Tooltip=Manage items and settings that belong to your account.
```

This is the default text.

5

Name

Summary

Sets which name to show on the menu item.

Usage

```
Name=<name>
```

Description

Sets which name to show on the menu item. If this setting is left empty, the default value will be used. But keep in mind that if you don't use the default value, the menu item can be translated.

This setting is case sensitive.

Examples

```
Name=My account
```

This is the default setting

5

NavigationPartIdentifier

Summary

Sets the navigation part of the identifier of the menu item.

Usage

```
NavigationPartIdentifier=<navigation part identifier>
```

Description

Sets the navigation part of the identifier of the menu item.
This setting is case sensitive.

Examples

```
NavigationPartIdentifier=ezmynavigationpart
```

This is the default setting.

5

[Topmenu_setup]

Enabled   (page 2120)
    Sets whether or not the menu item is clickable in current UI context.

Name   (page 2123)
    Sets which name to show on the menu item.

NavigationPartIdentifier   (page 2124)
    Sets the navigation part of the identifier of the menu item.

PolicyList   (page 2118)
    Lists the policies for the menu's.

Shown   (page 2119)
    Sets the view which an item points to in the different context.

Tooltip   (page 2122)
    Sets the tooltip which is shown for the menu item when the cursor hovers over the menu
    item.

URL   (page 2121)
    Sets the view which an item points to in the different context.

5

PolicyList

Summary

Lists the policies for the menu's.

Usage

```
PolicyList[]=<node_ID> | <module>/<function>
```

Description

Lists the policies for the menu's. The possible values are either the "<node_ID>" or according to the syntax "<module>/<function>".
In case the node ID is used the content/read is checked!
This setting is case sensitive.

Examples

```
PolicyList[]=setup/managecache
```

This is the default setting.

5

Shown

Summary

Sets the view which an item points to in the different context.

Usage

```
Shown[<ui_context>]=true | false
```

Description

Sets the view which an item points to in the different context.
The User Interface context can be default, browse or edit.
This setting is case sensitive.

Examples

```
Shown[default]=true
Shown[navigation]=true
Shown[browse]=true
```

This is the default setting.

5

Enabled

Summary

Sets whether or not the menu item is clickable in current UI context.

Usage

```
Enabled[<ui_context>]=true | false
```

Description

Sets whether or not the menu item is clickable in current UI context.
The User Interface context can be default, browse or edit.
This setting is case sensitive.

Examples

```
Enabled[default]=true
Enabled[browse]=false
Enabled[edit]=false
```

This is the default setting and it means that the menu will be enabled in the default mode, but
disabled in browse and edit mode.

5

URL

Summary

Sets the view which an item points to in the different context.

Usage

URL[<ui_context>]=<path>

Description

Sets the view which an item points to in the different context.
The User Interface context can be default, browse or edit.
This setting is case sensitive.

Examples

URL[default]=setup/cache

This is the default setting.

5

Tooltip

Summary

Sets the tooltip which is shown for the menu item when the cursor hovers over the menu item.

Usage

```
Tooltip=<message that appears as tooltip for this menu>
```

Description

Sets the tooltip which is shown for the menu item when the cursor hovers over the menu item. If the setting is left empty the default value is used. But keep in mind the when you don't use the default value the tooltip can be translated.
This setting is case sensitive.

Examples

```
Tooltip=Configure settings and manage advanced functionality.
```

This is the default tooltip.

5

Name

Summary

Sets which name to show on the menu item.

Usage

```
Name=<name>
```

Description

Sets which name to show on the menu item. If this setting is left empty, the default value will be used. But keep in mind that if you don't use the default value, the menu item can be translated. This setting is case sensitive.

Examples

```
Name=Setup
```

This is the default setting.

5

NavigationPartIdentifier

Summary

Sets the navigation part of the identifier of the menu item.

Usage

```
NavigationPartIdentifier=<navigation part identifier>
```

Description

Sets the navigation part of the identifier of the menu item.
This setting is case sensitive.

Examples

```
NavigationPartIdentifier=ezsetupnavigationpart
```

This is the default setting.

5

[Topmenu_design]

Enabled   (page 2127)
> Sets whether or not the menu item is clickable in current UI context.

Name   (page 2130)
> Sets which name to show on the menu item.

NavigationPartIdentifier   (page 2131)
> Sets the navigation part of the identifier of the menu item.

Shown   (page 2126)
> Sets whether or not the menu item is shown in current UI context

Tooltip   (page 2129)
> Sets the tooltip which is shown for the menu item when the cursor hovers over the menu item.

URL   (page 2128)
> Sets the view which an item points to in the different context.

5

Shown

Summary

Sets whether or not the menu item is shown in current UI context

Usage

```
Shown[<ui_context>]=true | false
```

Description

Sets whether or not the menu item is shown in current UI context
The User Interface context can be "default", "browse", "edit",....
This setting is case sensitive.

Examples

```
Shown[navigation]=true
 Shown[default]=true
 Shown[browse]=true
```

This is the default setting.

5

Enabled

Summary

Sets whether or not the menu item is clickable in current UI context.

Usage

```
Enabled[<ui_context>]=true | false
```

Description

Sets whether or not the menu item is clickable in current UI context.
The User Interface context can be default, browse or edit.
This setting is case sensitive.

Examples

```
Enabled[default]=true
 Enabled[browse]=false
 Enabled[edit]=false
```

This is the default setting.

5

URL

Summary

Sets the view which an item points to in the different context.

Usage

URL[<ui_context>]=<path>

Description

Sets the view which an item points to in the different context.
The User Interface context can be "default", "browse", "edit",....
This setting is case sensitive.

Examples

URL[default]=visual/toolbarlist

This is the default setting.

5

Tooltip

Summary

Sets the tooltip which is shown for the menu item when the cursor hovers over the menu item.

Usage

```
Tooltip=<message that appears as tooltip for this menu>
```

Description

Sets the tooltip which is shown for the menu item when the cursor hovers over the menu item.
If the setting is left empty the default value is used. But keep in mind the when you don't use the
default value the tooltip can be translated.
This setting is case sensitive.

Examples

```
Tooltip=Manage templates, menus, toolbars and other things related to
appearance.
```

This is the default text.

5

Name

Summary

Sets which name to show on the menu item.

Usage

```
Name=<name>
```

Description

Sets which name to show on the menu item. If this setting is left empty, the default value will be used. But keep in mind that if you don't use the default value, the menu item can be translated. This setting is case sensitive.

Examples

```
Name=Design
```

This is the default setting.

5

NavigationPartIdentifier

Summary

Sets the navigation part of the identifier of the menu item.

Usage

```
NavigationPartIdentifier=<navigation part identifier>
```

Description

Sets the navigation part of the identifier of the menu item.
This setting is case sensitive.

Examples

```
NavigationPartIdentifier=ezvisualnavigationpart
```

This is the default setting.

5

[Topmenu_shop]

Enabled   (page 2135)
  Sets whether or not the menu item is clickable in current UI context.

Name   (page 2138)
  Sets which name to show on the menu item.

NavigationPartIdentifier   (page 2139)
  Sets the navigation part of the identifier of the menu item.

PolicyList   (page 2133)
  Lists the policies for the menu's.

Shown   (page 2134)
  Sets whether or not the menu item is shown in current UI context

Tooltip   (page 2137)
  Sets the tooltip which is shown for the menu item when the cursor hovers over the menu item.

URL   (page 2136)
  Sets the view which an item points to in the different context.

5

PolicyList

Summary

Lists the policies for the menu's.

Usage

```
PolicyList[]=<node_ID> | <module>/<function>
```

Description

Lists the policies for the menu's. The possible values are either the "<node_ID>" or according to the syntax "<module>/<function>".
In case the node ID is used the content/read is checked!
This setting is case sensitive.

Examples

```
PolicyList[]=shop/administrate
```

This is the default setting.

5

Shown

Summary

Sets whether or not the menu item is shown in current UI context

Usage

```
Shown[<ui_context>]=true | false
```

Description

Sets whether or not the menu item is shown in current UI context
The User Interface context can be default, browse or edit.
This setting is case sensitive.

Examples

```
Shown[]
 Shown[navigation]=true
 Shown[default]=false
 Shown[browse]=true
```

This is the default setting. It implies that this menu item will not be shown the default view.

5

Enabled

Summary

Sets whether or not the menu item is clickable in current UI context.

Usage

```
Enabled[<ui_context>]=true | false
```

Description

Sets whether or not the menu item is clickable in current UI context.
The User Interface context can be default, browse or edit.
This setting is case sensitive.

Examples

```
Enabled[default]=true
 Enabled[browse]=false
 Enabled[edit]=false
```

This is the default setting.

5

URL

Summary

Sets the view which an item points to in the different context.

Usage

```
URL[<ui_context>]=<path>
```

Description

Sets the view which an item points to in the different context.
The User Interface context can be default, browse or edit.
This setting is case sensitive.

Examples

```
URL[default]=shop/orderlist
```

This is the default setting.

5

Tooltip

Summary

Sets the tooltip which is shown for the menu item when the cursor hovers over the menu item.

Usage

```
Tooltip=<message that appears as tooltip for this menu>
```

Description

Sets the tooltip which is shown for the menu item when the cursor hovers over the menu item. If the setting is left empty the default value is used. But keep in mind the when you don't use the default value the tooltip can be translated.
This setting is case sensitive.

Examples

```
Tooltip=Manage customers, orders, discounts and VAT types;
view sales  statistics.
```

This is the default setting.

5

Name

Summary

Sets which name to show on the menu item.

Usage

```
Name=<name>
```

Description

Sets which name to show on the menu item. If this setting is left empty, the default value will be used. But keep in mind that if you don't use the default value, the menu item can be translated. This setting is case sensitive.

Examples

```
Name=Webshop
```

This is the default setting.

5

NavigationPartIdentifier

Summary

Sets the navigation part of the identifier of the menu item.

Usage

```
NavigationPartIdentifier=<navigation part identifier>
```

Description

Sets the navigation part of the identifier of the menu item.
This setting is case sensitive.

Examples

```
NavigationPartIdentifier=ezshopnavigationpart
```

This is the default setting.

5

[Topmenu_users]

Enabled   (page 2143)
> Sets whether or not the menu item is clickable in current UI context.

Name   (page 2146)
> Sets which name to show on the menu item.

NavigationPartIdentifier   (page 2147)
> Sets the navigation part of the identifier of the menu item.

PolicyList   (page 2141)
> Lists the policies for the menu's.

Shown   (page 2142)
> Sets whether or not the menu item is shown in current UI context

Tooltip   (page 2145)
> Sets the tooltip which is shown for the menu item.

URL   (page 2144)
> Sets the view which an item points to in the different context.

5

PolicyList

Summary

Lists the policies for the menu's.

Usage

```
PolicyList[]=<node_ID> | <module>/<function>
```

Description

Lists the policies for the menu's. The possible values are either the "<node_ID>" or according to the syntax "<module>/<function>".
In case the node ID is used the content/read is checked!
This setting is case sensitive.

Examples

```
PolicyList[]=5
```

This is the default setting.

5

Shown

Summary

Sets whether or not the menu item is shown in current UI context

Usage

```
Shown[<ui_context>]=true | false
```

Description

Sets whether or not the menu item is shown in current UI context
The User Interface context can be default, browse or edit.
This setting is case sensitive.

Examples

```
Shown[default]=true
Shown[navigation]=true
Shown[browse]=true
```

This is the default setting.

5

Enabled

Summary

Sets whether or not the menu item is clickable in current UI context.

Usage

```
Enabled[<ui_context>]=true | false
```

Description

Sets whether or not the menu item is clickable in current UI context.
The User Interface context can be default, browse or edit.
This setting is case sensitive.

Examples

```
Enabled[default]=true
Enabled[browse]=true
Enabled[edit]=false
```

This is the default setting.

5

URL

Summary

Sets the view which an item points to in the different context.

Usage

URL[<ui_context>]=<path>

Description

Sets the view which an item points to in the different context.
The User Interface context can be default, browse or edit.
This setting is case sensitive.

Examples

URL[default]=content/view/full/5
 URL[browse]=content/browse/5

This is the default setting.

5

Tooltip

Summary

Sets the tooltip which is shown for the menu item.

Usage

```
Tooltip=<message that appears as tooltip for this menu>
```

Description

Sets the tooltip which is shown for the menu item when the cursor hovers over the menu item.
If the setting is left empty the default value is used. But keep in mind the when you don't use the
default value the tooltip can be translated.
This setting is case sensitive.

Examples

```
Tooltip=Manage users, user groups and permission settings.
```

This is the default setting.

5

Name

Summary

Sets which name to show on the menu item.

Usage

```
Name=<name>
```

Description

Sets which name to show on the menu item. If this setting is left empty, the default value will be used. But keep in mind that if you don't use the default value, the menu item can be translated. This setting is case sensitive.

Examples

```
Name= User accounts
```

This is the default setting.

5

NavigationPartIdentifier

Summary

Sets the navigation part of the identifier of the menu item.

Usage

```
NavigationPartIdentifier=<navigation part identifier>
```

Description

Sets the navigation part of the identifier of the menu item.
This setting is case sensitive.

Examples

```
NavigationPartIdentifier=ezusernavigationpart
```

This is the default setting.

5

[Topmenu_media]

Enabled   (page 2151)
    Sets whether or not the menu item is clickable in current UI context.

Name   (page 2154)
    Sets which name to show on the menu item.

NavigationPartIdentifier   (page 2155)
    Sets the navigation part of the identifier of the menu item.

PolicyList   (page 2149)
    Lists the policies for the menu's.

Shown   (page 2150)
    Sets whether or not the menu item is shown in current UI context.

Tooltip   (page 2153)
    Sets the tooltip which is shown for the menu item when the cursor hovers over the menu item.

URL   (page 2152)
    Sets the view which an item points to in the different context.

5

PolicyList

Summary

Lists the policies for the menu's.

Usage

```
PolicyList[]=<node_ID> | <module>/<function>
```

Description

Lists the policies for the menu's. The possible values are either the "<node_ID>" or according to the syntax "<module>/<function>".
In case the node ID is used the content/read is checked!
This setting is case sensitive.

Examples

```
PolicyList[]=43
```

This is the default setting.

5

Shown

Summary

Sets whether or not the menu item is shown in current UI context.

Usage

```
Shown[<ui_context>]=true | false
```

Description

Sets whether or not the menu item is shown in current UI context
The User Interface context can be default, browse or edit.
This setting is case sensitive.

Examples

```
Shown[default]=true
 Shown[navigation]=true
 Shown[browse]=true
```

This is the default setting.

5

Enabled

Summary

Sets whether or not the menu item is clickable in current UI context.

Usage

```
Enabled[<ui_context>]=true | false
```

Description

Sets whether or not the menu item is clickable in current UI context.
The User Interface context can be default, browse or edit.
This setting is case sensitive.

Examples

```
Enabled[default]=true
Enabled[browse]=true
Enabled[edit]=false
```

This is the default setting.

5

URL

Summary

Sets the view which an item points to in the different context.

Usage

```
URL[<ui_context>]=<path>
```

Description

Sets the view which an item points to in the different context.
The User Interface context can be default, browse or edit.
This setting is case sensitive.

Examples

```
URL[default]=content/view/full/43
 URL[browse]=content/browse/43
```

This is the default setting.

5

Tooltip

Summary

Sets the tooltip which is shown for the menu item when the cursor hovers over the menu item.

Usage

```
Tooltip=<tooltip text>
```

Description

Sets the tooltip which is shown for the menu item when the cursor hovers over the menu item. If the setting is left empty the default value is used. But keep in mind the when you don't use the default value the tooltip can be translated.
This setting is case sensitive.

Examples

```
Tooltip=Manage images, files, documents, etc.
```

This is the default setting.

5

Name

Summary

Sets which name to show on the menu item.

Usage

```
Name=<name>
```

Description

Sets which name to show on the menu item. If this setting is left empty, the default value will be used. But keep in mind that if you don't use the default value, the menu item can be translated. This setting is case sensitive.

Examples

```
Name=Media library
```

This is the default setting.

5

NavigationPartIdentifier

Summary

Sets the navigation part of the identifier of the menu item.

Usage

```
NavigationPartIdentifier=<navigation part identifier>
```

Description

Sets the navigation part of the identifier of the menu item.
This setting is case sensitive.

Examples

```
NavigationPartIdentifier=ezmedianavigationpart
```

The is the default setting.

5

[Topmenu_content]

Enabled   (page 2159)
>   Sets whether or not the menu item is clickable.

Name   (page 2161)
>   Sets which name to show on the menu item.

NavigationPartIdentifier   (page 2162)
>   Sets the navigation part of the identifier of the menu item.

PolicyList   (page 2157)
>   Lists the policies.

Shown   (page 2158)
>   Sets whether or not the menu item is shown.

Tooltip   (page 2160)
>   Sets the tooltip which is shown for the menu item when the cursor hovers over the menu
>   item.

URL   (page 2163)
>   Sets the view which an item points to in the different context.

5

PolicyList

Summary

Lists the policies.

Usage

```
PolicyList[]=<node_ID> | <module>/<function>
```

Description

Lists the policies for the menu's.
The possible values are either the "<node ID>" or according to the syntax "<module>/
<function>".
In case the node ID is used the content/read is checked!
This setting is case sensitive.

Examples

```
PolicyList[]=2
```

The is the default setting.

5

Shown

Summary

Sets whether or not the menu item is shown.

Usage

```
Shown[<uit_context>]=true|false
```

Description

Sets whether or not the menu item is shown in current UI context.
The User Interface context can be default, browse or edit.
This setting is case sensitive.

Examples

```
Shown[default]=true
 Shown[navigation]=true
 Shown[browse]=true
```

The is the default setting.

5

Enabled

Summary

Sets whether or not the menu item is clickable.

Usage

```
Enabled[<ui_context>]=true|false
```

Description

Sets whether or not the menu item is clickable in current UI context.
The User Interface context can be default, browse or edit.
The value "true" enables and "false" disables.
This setting is case sensitive.

Examples

```
Enabled[default]=true
 Enabled[browse]=true
 Enabled[edit]=false
```

The is the default setting. It implies that this menu item is not clickable in Edit-mode.

5

Tooltip

Summary

Sets the tooltip which is shown for the menu item when the cursor hovers over the menu item.

Usage

```
Tooltip=<tooltip text>
```

Description

Sets the tooltip which is shown for the menu item when the cursor hovers over the menu item. If the setting is left empty the default value is used. But keep in mind the when you don't use the default value the tooltip can be translated.
This setting is case sensitive.

Examples

```
Tooltip=Manage the main content structure of the site.
```

The is the default setting.

5

Name

Summary

Sets which name to show on the menu item.

Usage

```
Name=<name menu item>
```

Description

Sets which name to show on the menu item. If this setting is left empty, the default value will be used. But keep in mind that if you don't use the default value, the menu item can be translated. This setting is case sensitive.

Examples

```
Name=Content structure
```

The is the default setting.

5

NavigationPartIdentifier

Summary

Sets the navigation part of the identifier of the menu item.

Usage

```
NavigationPartIdentifier=<name navigation part identifier>
```

Description

Sets the navigation part of the identifier of the menu item.
This setting is case sensitive.

Examples

```
NavigationPartIdentifier=ezcontentnavigationpart
```

The is the default setting.

5

URL

Summary

Sets the view which an item points to in the different context.

Usage

URL[<ui_context>]=<path>

Description

Sets the view which an item points to in the different context.
The User Interface context can be default, browse or edit.
This setting is case sensitive.

Examples

```
URL[default]=content/view/full/2
 URL[browse]=content/browse/2
```

The default setting.

5

[TopAdminMenu]

Tabs   (page)
      Lists the menu items of the top menu in administrator interface.

5

Tabs

Summary

Lists the menu items of the top menu in administrator interface.

Usage

```
Tabs[]=<menu name>
```

Description

This list contains menu items (or tabs) of the top menu in administrator interface. This setting is
case sensitive.

Note: As of eZ Publish 4.3 the tabs "shop" and "design" are hidden by default. To enable the
"shop" and "design" follow the example below.

Examples

```
Tabs[]=dashboard
 Tabs[]=content
 Tabs[]=media
 Tabs[]=users
 Tabs[]=setup
```

The default setting is that the dashboard, content, media, users and setup tabs are displayed.

You enable the eZ Webshop tab "shop" and the design tab "design" by entering the following
settings:

```
Tabs[]=shop
Tabs[]=design
```

5

[NavigationPart]

Part   (page 2167)
  A list of the current navigation parts and their names.

5

Part

Summary

A list of the current navigation parts and their names.

Usage

```
Part[<identifier navigation part>]=<name navigation part>
```

Description

A list of the current navigation parts and their names. Each entry consists of the identifier as key and the name as value.
Note: If you wish to have the name translatable you will need to create a dummy PHP file with the following in it:

```
ezi18n( 'kernel/navigationpart', 'Custom navigation', 'Navigation part'  );
```

This will cause the ezlupdate to include it in the .ts file.
This setting is case sensitive.

Examples

```
Part[ezcontentnavigationpart]=Content structure
Part[ezmedianavigationpart]=Media library
Part[ezusernavigationpart]=User accounts
Part[ezshopnavigationpart]=Webshop
Part[ezvisualnavigationpart]=Design
Part[ezsetupnavigationpart]=Setup
Part[ezmynavigationpart]=My account
```

5

[MenuContentSettings]

LeftIdentifierList   (page 2169)

>   Lists the identifiers of the classses that are allowed to be shown in left menu.

TopIdentifierList   (page 2170)

>   Lists the identifiers of the classses that are allowed to be shown in top menu's.

5

LeftIdentifierList

Summary

Lists the identifiers of the classses that are allowed to be shown in left menu.

Usage

```
LeftIdentifierList[]=<class identifier>
```

Description

This list contains the identifiers of the classses that are allowed to be shown in left menu.
This setting is case sensitive.

Examples

```
LeftIdentifierList[]=folder
 LeftIdentifierList[]=feedback_form
 LeftIdentifierList[]=gallery
 LeftIdentifierList[]=forum
```

This is the default value used for this setting.

5

TopIdentifierList

Summary

Lists the identifiers of the classses that are allowed to be shown in top menu's.

Usage

```
TopIdentifierList[]=<class identifier>
```

Description

This list contains the identifiers of the classses that are allowed to be shown in top menu's.

Examples

```
TopIdentifierList[]=folder
TopIdentifierList[]=feedback_form
TopIdentifierList[]=gallery
TopIdentifierList[]=forum
```

This is the default value used for this setting.

5

[LeftTop]

LeftMenu   (page 2172)
     Sets the left menu to use for the "LeftTop" menu array.

MenuThumbnail   (page 2174)
     Sets the menu thumbnail to use for the "LeftTop" menu array.

TitleText   (page 2175)
     Sets the title text to use for the "LeftTop" menu array.

TopMenu   (page 2173)
     Sets the top menu to use for the "LeftTop" menu array.

5

LeftMenu

Summary

Sets the left menu to use for the "LeftTop" menu array.

Usage

```
LeftMenu=<name menu>
```

Description

Sets the left menu to use for the "LeftTop" menu array.
This setting is case sensitive.

Examples

```
LeftMenu=sub_left
```

"sub_left" is the default value used for this setting.

5

TopMenu

Summary

Sets the top menu to use for the "LeftTop" menu array.

Usage

```
TopMenu=<name menu>
```

Description

Sets the title text to use for the "LeftTop" menu array.
This setting is case sensitive.

Examples

```
TopMenu=flat_top
```

"flat_top" is the default value used for this setting.

5

MenuThumbnail

Summary

Sets the menu thumbnail to use for the "LeftTop" menu array.

Usage

MenuThumbnail=<name thumbnail>

Description

Sets the menu thumbnail to use for the "LeftTop" menu array.
This setting is case sensitive.

Examples

MenuThumbnail=menu/left_top.jpg

This is the default value used for this setting.

5

TitleText

Summary

Sets the title text to use for the "LeftTop" menu array.

Usage

```
TitleText=<text>
```

Description

Sets the title text to use for the "LeftTop" menu array.
This setting is case sensitive.

Examples

```
TitleText=Left and top
```

"Left and top" is the default value used for this setting.

5

[DoubleTop]

LeftMenu   (page 2177)
      Sets the left menu to use for the ”DoubleTop” menu array.

MenuThumbnail   (page 2179)
      Sets the menu thumbnail to use for the ”DoubleTop” menu array.

TitleText   (page 2180)
      Sets the title text to use for the ”DoubleTop” menu array.

TopMenu   (page 2178)
      Sets the top menu to use for the ”DoubleTop” menu array.

5

LeftMenu

Summary

Sets the left menu to use for the "DoubleTop" menu array.

Usage

```
LeftMenu=<name left menu>
```

Description

Sets the menu thumbnail to use for the "DoubleTop" menu array.
This setting is case sensitive.

Examples

```
LeftMenu=
```

By default this value is left empty.

5

TopMenu

Summary

Sets the top menu to use for the "DoubleTop" menu array.

Usage

```
TopMenu=<name menu>
```

Description

Sets the top menu to use for the "DoubleTop" menu array.
This setting is case sensitive.

Examples

```
TopMenu=double_top
```

"double_top" is the default value used for this setting.

5

MenuThumbnail

Summary

Sets the menu thumbnail to use for the "DoubleTop" menu array.

Usage

```
MenuThumbnail=<name thumbnail>
```

Description

Sets the menu thumbnail to use for the "DoubleTop" menu array.
This setting is case sensitive.

Examples

```
MenuThumbnail=menu/double_top.jpg
```

This is the default value used for this setting.

5

TitleText

Summary

Sets the title text to use for the "DoubleTop" menu array.

Usage

```
TitleText=<text>
```

Description

Sets the title text to use for the "DoubleTop" menu array.
This setting is case sensitive.

Examples

```
TitleText=Double top menu
```

"Double top menu" is the default value used for this setting.

5

[LeftOnly]

LeftMenu   (page)
    Sets the left menu to use for the "LeftOnly" menu array.

MenuThumbnail   (page)
    Sets the menu thumbnail to use for the "LeftOnly" menu array.

TitleText   (page)
    Sets the title text to use for the "LeftOnly" menu array.

TopMenu   (page)
    Sets the top menu to use for the "LeftOnly" menu array.

5

LeftMenu

Summary

Sets the left menu to use for the "LeftOnly" menu array.

Usage

```
LeftMenu=<name left menu>
```

Description

Sets the left menu to use for the "LeftOnly" menu array.
This setting is case sensitive.

Examples

```
LeftMenu=flat_left
```

"flat_left" is the default value used for this setting.

5

TopMenu

Summary

Sets the top menu to use for the "LeftOnly" menu array.

Usage

```
TopMenu=<menu>
```

Description

Sets the top menu to use for the "LeftOnly" menu array.
This setting is case sensitive.

Examples

```
TopMenu=
```

This setting is left empty by default.

5

MenuThumbnail

Summary

Sets the menu thumbnail to use for the "LeftOnly" menu array.

Usage

```
MenuThumbnail=<name thumbnail>
```

Description

Sets the menu thumbnail to use for the "LeftOnly" menu array.
This setting is case sensitive.

Examples

```
MenuThumbnail=menu/left_only.jpg
```

This is the default value for this setting.

5

TitleText

Summary

Sets the title text to use for the "LeftOnly" menu array.

Usage

```
TitleText=<text>
```

Description

Sets the title text to use for the "LeftOnly" menu array.
This setting is case sensitive.

Examples

```
TitleText=Left menu
```

"Left menu" is the default value used for this setting.

5

[TopOnly]

LeftMenu   (page 2187)
> Sets the left menu to use for the "TopOnly" menu array.

MenuThumbnail   (page 2189)
> Sets the thumbnail to be used for the "TopOnly" menu array.

TitleText   (page 2190)
> Sets the title text for the "TopOnly" menu array.

TopMenu   (page 2188)
> Sets the top menu to use for the "TopOnly" menu array.

5

LeftMenu

Summary

Sets the left menu to use for the "TopOnly" menu array.

Usage

```
LeftMenu=<name left menu>
```

Description

Sets the left menu to use for the "TopOnly" menu array.
This setting is case sensitive.

Examples

```
LeftMenu=
```

By default this value is left empty.

5

TopMenu

Summary

Sets the top menu to use for the "TopOnly" menu array.

Usage

```
TopMenu=<name top menu>
```

Description

Sets the top menu to use for the "TopOnly" menu array.
This setting is case sensitive.

Examples

```
TopMenu=flat_top
```

"flat_top" is the default value used for this setting.

5

MenuThumbnail

Summary

Sets the thumbnail to be used for the "TopOnly" menu array.

Usage

```
MenuThumbnail=<thumbnail>
```

Description

Sets the thumbnail to be used for the "TopOnly" menu array.
This setting is case sensitive.

Examples

```
MenuThumbnail=menu/top_only.jpg
```

This is the default value used for this setting.

5

TitleText

Summary

Sets the title text for the "TopOnly" menu array.

Usage

```
TitleText=<text>
```

Description

Sets the title text for the "TopOnly" menu array.
This setting is case sensitive.

Examples

```
TitleText=Only top menu
```

"Only top menu" is the default value used for this setting.

5

[SelectedMenu]

CurrentMenu   (page 2194)
    Sets the current menu.

LeftMenu   (page 2192)
    Sets the left menu.

TopMenu   (page 2193)
    Sets the top menu

5

LeftMenu

Summary

Sets the left menu.

Usage

```
LeftMenu=<name menu>
```

Description

Sets the left menu. This setting is case sensitive.

Examples

```
LeftMenu=
```

By default this setting is left empty.

5

TopMenu

Summary

Sets the top menu

Usage

```
TopMenu=<name menu>
```

Description

Sets the top menu.
This setting is case sensitive.

Examples

```
TopMenu=flat_top
```

"flat_top" is the default value used for this setting.

5

CurrentMenu

Summary

Sets the current menu.

Usage

```
CurrentMenu=<name menu>
```

Description

Sets the current menu. The value used here must be set in "[MenuSetting] AvailableMenuArray[]=". Also the value must be defined in its own block. For example, if "CurrentMenu=TopOnly" is set, the block [TopOnly] must exist.
This setting is case sensitive.

Examples

```
CurrentMenu=TopOnly
```

"TopOnly" is the default value used for this setting.

5

[MenuSettings]

AvailableMenuArray   (page)
    Sets the available array's for your menu.

5

AvailableMenuArray

Summary

Sets the available array's for your menu.

Usage

```
AvailableMenuArray[]=<name menu array>
```

Description

Sets the available array's for your menu.
The values set here must also be defined in their own block. For example, when "AvailableMenuArray[]=TopOnly" is set, the block [TopOnly] should also exist.
This setting is case sensitive.

Examples

```
AvailableMenuArray[]
 AvailableMenuArray[]=TopOnly
 AvailableMenuArray[]=LeftOnly
 AvailableMenuArray[]=DoubleTop
 AvailableMenuArray[]=LeftTop
```

This is the default setting.

5

5.14.28   module.ini

The configuration blocks are documented in the following sections:

- [ModuleOverrides] (page 2198)
- [ModuleSettings] (page 2200)

5

[ModuleOverrides]

NavigationPart   (page)
    Controls the override of navigation parts.

5

NavigationPart

Summary

Controls the override of navigation parts.

Usage

```
NavigationPart[<module_name>]=<name_navigation_part>
NavigationPart[<module_name>/<view>]=<name_navigation_part>
```

Description

This setting controls the override of navigation parts defined in module.php or by a dynamic code.
As key the module name can be used alone or together with a view.
By default this setting is left empty and commented out.
This setting is case sensitive.

Examples

```
NavigationPart[content/trash]=ezsetupnavigationpart
```

This setting will allow the trash view of the content module to appear in the setup tab.

5

[ModuleSettings]

ExtensionRepositories   (page)
    Lists the extensions which have modules.

ModuleList   (page)
    Lists the known modules in the system.

ModuleRepositories   (page)
    Sets the path to the module repository.

5

ModuleList

Summary

Lists the known modules in the system.

Usage

```
ModuleList[]=<module_name>
```

Description

This setting lists the known modules in the system. Extensions with modules should also be listed here. This setting is used to avoid URL aliases overriding existing module names.
Since eZ Publish 4.4 the system will first read this ModuleList setting to see if the module exists before it reads the Rules setting in site.ini [SiteAccessRules], to see if the module view is enabled or disabled. This is done simply because normally an extension will not be disabled if it does not exist.
This setting is case sensitive.

Examples

```
ModuleList[]
ModuleList[]=class
ModuleList[]=collaboration
ModuleList[]=content
ModuleList[]=error
ModuleList[]=ezinfo
ModuleList[]=infocollector
ModuleList[]=layout
ModuleList[]=notification
ModuleList[]=package
ModuleList[]=pdf
ModuleList[]=role
ModuleList[]=rss
ModuleList[]=search
ModuleList[]=section
ModuleList[]=settings
ModuleList[]=setup
ModuleList[]=shop
ModuleList[]=state
ModuleList[]=trigger
ModuleList[]=url
ModuleList[]=user
```

5

```
ModuleList[]=visual
ModuleList[]=workflow
ModuleList[]=switchlanguage
```

These are the default settings.

5

ExtensionRepositories

Summary

Lists the extensions which have modules.

Usage

```
ExtensionRepositories[]=<name_extension>
```

Description

This setting lists the extensions which have modules.
It is recommended to create a module.ini.append.php file within these extensions, at the following location:
(...root of your eZ Publish installation)/extension/<extension>/settings/module.ini.append
By adding the extension name to that file, the modules from the extension will automatically be loaded when the extension is turned on.
This setting is case sensitive.
By default this setting is left empty.

5

ModuleRepositories

Summary

Sets the path to the module repository.

Usage

```
ModuleRepositories[]=<path_to_repository
```

Description

This setting sets the path to the module repository.
This setting is case sensitive.

Examples

```
ModuleRepositories[]=kernel
ModuleRepositories[]=kernel/private/modules
```

These are the default locations of the repository.

5

5.14.29   notification.ini

The configuration blocks are documented in the following sections:

\begin{itemize}   \item{}   [MailSettings]\refer{labelbf61cd820673d5436e1265b4b33c941e} \item{}   [NotificationEventHandlerSettings]\refer{labeld5ae2224f3ccc7e79ee3e68bfb25621a} \item{}   [NotificationEventTypeSettings]\refer{labeld0d4f0ae262349ac00801410f16b23ff} \item{}   [RuleSettings]\refer{labelf17e4153497c2b725325e71b61328e23}   \item{} [TransportSettings]\refer{label5defa2104bfd113c728214bd27d4699c} \end{itemize}

5

[RuleSettings]

Alias   (page 2208)
    DEPRECATED

ExtensionDirectories   (page 2209)
    DEPRECATED

LimitDeleteElements   (page 2207)
    Limits the number of elements to be deleted when sending notifications in digest mode.

RepositoryDirectories   (page 2210)
    DEPRECATED

5

LimitDeleteElements

Summary

Limits the number of elements to be deleted when sending notifications in digest mode.

Usage

```
LimitDeleteElements =<integer>
```

Description

This sets the maximum number of notification digest elements that are to be removed in one query. This is needed to prevent exceeding the DBMS limits (maximum query length, maximum number of elements).

Examples

```
LimitDeleteElements=50
```

This is the default setting.

5

Alias

Summary

DEPRECATED

Usage

```
Alias[]=<alias>
```

Examples

```
Alias[advanced]=ezadvanced
Alias[general]=ezgeneral
Alias[keyword]=ezkeyword
```

5

ExtensionDirectories

Summary

DEPRECATED

Usage

```
ExtensionDirectories[]=
```

Examples

```
ExtensionDirectories[]=
```

5

RepositoryDirectories

Summary

DEPRECATED

Usage

```
RepositoryDirectories[]=<path to repository>
```

Examples

```
RepositoryDirectories[]=kernel/notification/rules
```

5

[NotificationEventTypeSettings]

AvailableNotificationEventTypes   (page 2212)
> Defines the list of available notification event types.

ExtensionDirectories   (page 2213)
> Defines in which extension eZ Publish must look for notification event types.

RepositoryDirectories   (page 2214)
> Defines the default location where notification event type are stored.

5

AvailableNotificationEventTypes

Summary

Defines the list of available notification event types.

Usage

```
AvailableNotificationEventTypes[] =<notification event type name>
```

Description

eZ Publish will search for a notification event type class in each type defined.
Each type must be stored in

```
(path to notifications types)/mycustomcollaboration/
mycustomcollaborationtype.php
```

Keep in mind that you must replace "(path to notifications types)" with the actual path to the
collaboration type.

Examples

```
AvailableNotificationEventTypes[]=ezpublish
AvailableNotificationEventTypes[]=eznewcontent
AvailableNotificationEventTypes[]=ezcurrenttime
AvailableNotificationEventTypes[]=ezcollaboration
AvailableNotificationEventTypes[]=mycustomcollaboration
```

5

ExtensionDirectories

Summary

Defines in which extension eZ Publish must look for notification event types.

Usage

```
ExtensionDirectories[]=<myextension>
```

Description

Defines in which extension eZ Publish must look for notification event types.
In an extension, these new types must be stored under the 'notificationtypes' directory.

Examples

```
ExtensionDirectories[]=ezsurvey
```

5

RepositoryDirectories

Summary

Defines the default location where notification event type are stored.

Usage

```
RepositoryDirectories[] =<path to location>
```

Description

Defines the default location where notification event type are stored.

Examples

```
RepositoryDirectories[]=kernel/classes/notification/event/
```

5

[NotificationEventHandlerSettings]

Note: These settings are similar to the ones in [NotificationEventTypes] except the suffix must be "handler.php"

AvailableNotificationEventTypes   (page 2216)
  eZ Publish will search for a notification event handler class in each type defined.

ExtensionDirectories   (page 2217)
  Defines in which extension eZ Publish must look for notification event handlers.

RepositoryDirectories   (page 2218)
  Defines the default location where the notification event handlers are stored.

5

AvailableNotificationEventTypes

Summary

eZ Publish will search for a notification event handler class in each type defined.

Usage

```
AvailableNotificationEventTypes[]=<type>
```

Description

eZ Publish will search for a notification event handler class in each type defined.

Each type must be stored in

```
(path to notificationstypes)/mycustomcollaboration/
mycustomcollaborationhandler.php
```

Keep in mind that you must replace "(path to notifications types)" with the actual path and folder containing the collaboration type.

Examples

```
AvailableNotificationEventTypes[]=ezgeneraldigest

AvailableNotificationEventTypes[]=ezcollaborationnotification

AvailableNotificationEventTypes[]=ezsubtree

AvailableNotificationEventTypes[]=ezsimpledigest
```

5

ExtensionDirectories

Summary

Defines in which extension eZ Publish must look for notification event handlers.

Usage

```
ExtensionDirectories[]=<name extension>
```

Description

Defines in which extension eZ Publish must look for notification event handlers.

5

RepositoryDirectories

Summary

Defines the default location where the notification event handlers are stored.

Usage

```
RepositoryDirectories[] =<path to location handler>
```

Description

Defines the default location where notification event handlers are stored.

Examples

```
RepositoryDirectories[]=kernel/classes/notification/handler/
```

5

[TransportSettings]

DefaultTransport   (page 2221)
    Defines the default mail transport for notifications.

TransportPluginPath   (page 2220)
    Defines a new mail transport system for notifications.

5

TransportPluginPath

Summary

Defines a new mail transport system for notifications.

Usage

```
TransportPluginPath[]=<path_to_plugin>
```

Description

Defines a new mail transport system for notifications. An array of paths to the transport plugins. The value used here must be

```
<path to>/<customnotificationtransport.php>
```

Note that the file name (in the previous example "custom") must be appended with "notification-transport.php" and the class must extend "eZNotificationTransport".

5

DefaultTransport

Summary

Defines the default mail transport for notifications.

Usage

```
DefaultTransport=<transport name>
```

Description

Defines the default mail transport for notifications.

Examples

```
DefaultTransport=mail
```

This will use the default mail transport for the notification system.

5

[MailSettings]

EmailSender   (page)
Defines the default sender e-mail address for mails sent from eZ Publish.

5

EmailSender

Summary

Defines the default sender e-mail address for mails sent from eZ Publish.

Usage

```
EmailSender=<email@host.ext>
```

Description

The specified e-mail address will be used as the default in the from field when sending notification e-mails from eZ Publish. By default this setting is left empty, in this case the e-mail address defined in the configuration file "site.ini", block "[MailSettings] AdminEmail=" will be used instead.

Examples

```
EmailSender=nospam@ez.no
```

5

5.14.30   override.ini

Please refer to the following pages:

- The template override system (page 226)
- Template override example (page 228)
- Template override conditions (page 1444)

5.14.31 package.ini

The configuration blocks are documented in the following sections:

\begin{itemize} \item{} [CreationSettings]\refer{labelfe2dc01ea25a688193a55d6a263fdc0c}
\item{}   [InstallerSettings]\refer{label663bc8a48cbac82da51cc817cd90dbf8}
\item{}   [MaintainerSettings]\refer{label42775ccf98c6c0377aea1da09c08815e}
\item{}   [PackageSettings]\refer{labelad426cac6350cfead1f3b90ddf135e0f}   \item{}
[RepositorySettings]\refer{labelc00778c515731363ec4ad4e873b0cf67} \end{itemize}

5

[RepositorySettings]

\begin{description} \item[RepositoryDirectory]\refer{label7234241755a309cfd176cf25c4f2037f}\
\\ This setting sets the repository directory.

\end{description}

5

RepositoryDirectory

Summary

This setting sets the repository directory.

Usage

```
RepositoryDirectory=<name_repository_directory>
```

Description

This setting sets the default repository directory.

This setting is case sensitive.

Examples

```
RepositoryDirectory=packages
```

This is the default setting.

5

[RepositorySettings]

[PackageSettings]

\begin{description} \item[ExtensionDirectories]\refer{label3199e4fe49f25394f4d40b53a86380b8}\ \\ Lists the extensions which support package handlers.

\item[HandlerAlias]\refer{label321027522a6cae31b07ec790b8c19a41}\ \\ Maps handlers to their aliases.

\item[RepositoryDirectories]\refer{label99be0e362300d75da59940765f1b1a60}\ \\ Sets the default package repository.

\item[StateList]\refer{label5fad7dd4000abb76389118ad51a80d6d}\ \\ This setting lists the possible states of the packages

\item[TypeList]\refer{labela1cc9cfe60a4858c86d80d094ea5865a}\ \\ Sets the package types for specific package ID's or identifiers.

\end{description}

5

RepositoryDirectories

Summary

Sets the default package repository.

Usage

```
RepositoryDirectories[]=<path_to_repository>
```

Description

This setting sets the default repository where packages are located.

This setting is case sensitive.

Examples

```
RepositoryDirectories[]=kernel/classes
```

This is the default setting.

5

ExtensionDirectories

Summary

Lists the extensions which support package handlers.

Usage

```
ExtensionDirectories[]=<name_extension>
```

Description

This setting lists the extensions which support package handlers. For these extensions the package handlers must be placed in the subdirectory "packagehandlers" located here:...(root of your eZ Publish installation)/extension/<EXTENSION_NAME>/packagehandlers

By default this setting is left empty.

This setting is case sensitive and left empty by default.

5

HandlerAlias

Summary

Maps handlers to their aliases.

Usage

```
HandlerAlias[<alias>]=<handler_name>
```

Description

This setting will map handlers to their aliases. Aliases provide a more people friendly name for handlers and can also be used as a shortcut for doing specific items.

This setting is case sensitive.

Examples

```
HandlerAlias[file]=ezfile
HandlerAlias[ini]=ezfile
HandlerAlias[design]=ezfile
HandlerAlias[template]=ezfile
HandlerAlias[thumbnail]=ezfile
```

This are set by default. And "file", "ini", "design", "template" and "thumbnail" are the aliases that can be used for the package handler "ezfile".

5

TypeList

Summary

Sets the package types for specific package ID's or identifiers.

Usage

```
TypeList[<ID>]=<package type>
TypeList[<Identifier>]=<package type>
```

Description

This setting list the package types and matches them to ID's and Identifiers.

This setting is case sensitive.

Examples

```
TypeList[]
 TypeList[contentobject]=Content object
 TypeList[contentclass]=Content class
 TypeList[sitestyle]=Site style
 TypeList[site]=Site
```

These are the default settings.

5

StateList

Summary

This setting lists the possible states of the packages

Usage

```
StateList[]=
```

Description

This setting lists the state of packages.

This setting is case sensitive.

Examples

```
StateList[]
 StateList[mature]=Mature
 StateList[stable]=Stable
 StateList[beta]=Beta
 StateList[alpha]=Alpha
 StateList[prealpha]=Pre-Alpha
```

These are the default settings.

5

[MaintainerSettings]

\begin{description}   \item[RoleList]\refer{labeld79a51da90d037d625876a43896d3cfe}\   \\
Lists the roles of maintainers.

\end{description}

5

RoleList

Summary

Lists the roles of maintainers.

Usage

```
RoleList[]=<name_role>
```

Description

This setting lists the roles of maintainers.

This setting is case sensitive.

Examples

```
RoleList[]
 RoleList[]=lead
 RoleList[]=developer
 RoleList[]=designer
 RoleList[]=contributor
 RoleList[]=tester
```

These are the default settings.

5

[CreationSettings]

\begin{description} \item[HandlerAlias]\refer{label039f79fd613473a145421ed09dcee4ef}\ \\
Maps the package creation handlers to their aliases.

\item[HandlerList]\refer{labele004817bb614230bed7ca1fdcb71ae6a}\ \\ Lists the package
creation handlers.

\end{description}

5

HandlerList

Summary

Lists the package creation handlers.

Usage

```
HandlerList[]=<name_handler>
```

Description

This setting lists the package creation handlers. Each extension which has package creation handlers support must place their package creation handlers in the sub-directory package creators, located here:

```
...(root of your eZ Publish installation)/extension/<extension_name>/
packagecreators
```

This setting is case sensitive.

Examples

```
HandlerList[]
 HandlerList[]=ezcontentclass
 HandlerList[]=ezstyle
 HandlerList[]=ezcontentobject
```

These are the default settings.

5

HandlerAlias

Summary

Maps the package creation handlers to their aliases.

Usage

```
HandlerAlias[<alias>]=<handler_name>
```

Description

This setting maps the package creation handlers to their aliases. Aliases provide a more people friendly name for handlers and can also be used as a shortcut for doing specific items.

This setting is case sensitive.

Examples

```
HandlerAlias[ezcontentclass]=eZContentClassPackageCreator
HandlerAlias[ezstyle]=eZStylePackageCreator
HandlerAlias[ezcontentobject]=eZContentObjectPackageCreator
HandlerAlias[ezextension]=eZExtensionPackageCreator
```

5

[InstallerSettings]

\begin{description} \item[HandlerAlias]\refer{label18d4bd76ebdb7c8d1c76dd9aab926e45}\ \\ Maps the package installation handlers to their aliases.

\item[HandlerList]\refer{label448965fc1d3bff074fb53d8d530a50bc}\ \\ Lists the package installation handlers.

\end{description}

5

HandlerList

Summary

Lists the package installation handlers.

Usage

```
HandlerList[]=<name_handler>
```

Description

This setting lists the package installation handlers. Each extension which has package instal-lation handlers support must place their package installation handlers in the sub-directory package installers located here:

```
...(root of your eZ Publish installation)/extension/<extension_name>/
packageinstallers
```

This setting is case sensitive.

Examples

```
HandlerList[]
HandlerList[]=ezcontentobject
```

These are the default settings.

5

HandlerAlias

Summary

Maps the package installation handlers to their aliases.

Usage

```
HandlerAlias[<alias>]=<handler_name>
```

Description

This setting maps the package installation handlers to their aliases. Aliases provide a more people friendly name for handlers and can also be used as a shortcut for doing specific items.

This setting is case sensitive.

Examples

```
HandlerAlias[ezcontentobject]=eZContentObjectPackageInstaller
HandlerAlias[ezinstallscript]=eZInstallScriptPackageInstaller
```

5

5.14.32   paymentgateways.ini

The configuration blocks are documented in the following sections:

\begin{itemize} \item{} [GatewaysSettings]\refer{label8ad7d45fcc63b1b3ff8749e8233a78e9} \end{itemize}

5

[GatewaysSettings]

Available gateways

Summary

Sets the gateways to use for payment.

Usage

```
AvailableGateways[]=<gateway_name>
```

Description

This setting lists the gateways to use for payment. The gateways listed here should be placed in a gateway directory listed with the setting GatewaysDirectories.

This setting is case sensitive and left empty by default.

5

Gateways directories

Summary

Sets the gateway directories.

Usage

```
GatewaysDirectories[]=<path_to_gateway>
```

Description

This setting sets the location of the directories which have gateways.

This setting is case sensitive.

Examples

```
GatewaysDirectories[]<strong>=</strong>kernel/shop/paymentgateways
```

This is the default setting.

5

5.14.33 setup.ini

The configuration blocks are documented in the following sections:

- [curl_extension] (page 2311)
- [DatabaseSettings] (page 2254)
- [database_all_extensions] (page 2284)
- [database_extensions] (page 2287)
- [DemoSettings] (page 2252)
- [directory_permissions] (page 2296)
- [dom_extension] (page 2272)
- [execution_time] (page 2298)
- [ezcversion] (page 2314)
- [file_upload] (page 2290)
- [iconv_extension] (page 2308)
- [imagemagick_program] (page 2265)
- [image_conversion] (page 2281)
- [memory_limit] (page 2300)
- [open_basedir] (page 2292)
- [phpversion] (page 2305)
- [php_session] (page 2302)
- [settings_permission] (page 2294)
- [SiteTemplates] (page 2248)
- [texttoimage_functions] (page 2278)
- [zlib_extension] (page 2275)

5

[SiteTemplates]

Name   (page 2249)
    Sets the name of the site template.

ThumbnailBase   (page 2251)
    Sets the site template for the thumbnail.

ThumbnailExtension   (page 2250)
    Sets the extension for the site template for the thumbnail.

5

Name

Summary

Sets the name of the site template.

Usage

```
Name[]=<name site template>
```

Description

Sets the name of the site template.
This setting is case sensitive.

Examples

```
Name[]=corporate
Name[]=forum
Name[]=magazine
```

These value's are set by default.

5

ThumbnailExtension

Summary

Sets the extension for the site template for the thumbnail.

Usage

```
ThumbnailExtension=<extension thumbnail>
```

Description

Sets the extension for the site template for the thumbnail.
This setting is case sensitive.

Examples

```
ThumbnailExtension=png
```

This is the default setting.

5

ThumbnailBase

Summary

Sets the site template for the thumbnail.

Usage

ThumbnailBase=<name thumbnail>

Description

Sets the site template for the thumbnail.
This setting is case sensitive.

Examples

ThumbnailBase=eZ_setup_template

This is the default setting

5

[DemoSettings]

DataFile   (page)
   Sets the file name for the Demo settings datafile.

DataFile

Summary

Sets the file name for the Demo settings datafile.

Usage

```
DataFile=<name file>
```

Description

Sets the file name for the Demo settings datafile.
This setting is case sensitive.

Examples

```
DataFile=var.tgz
```

This is the default setting

5

[DatabaseSettings]

CleanSQL   (page 2259)
> Sets the file name for the CleanSQL file.

CleanSQLData   (page 2256)
> Sets the file name for the CleanSQLData file.

DefaultName   (page 2262)
> Sets the default database name.

DefaultPassword   (page 2260)
> Sets the default database password.

DefaultPort   (page 2263)
> Sets the default port name.

DefaultServer   (page 2264)
> Sets the default server name.

DefaultUser   (page 2261)
> Sets the default database user name.

DemoSQL   (page 2258)
> Sets the file name for the DemoSQL file.

DemoSQLData   (page 2255)
> Sets the file name for the DemoSQLData file.

SQLSchema   (page 2257)
> Sets the file name for the SQLSchema file.

5

DemoSQLData

Summary

Sets the file name for the DemoSQLData file.

Usage

```
DemoSQLData=<demodata sql file name>
```

Description

Sets the file name for the DemoSQLData file.
This setting is case sensitive.

Examples

```
DemoSQLData=demodata.sql
```

This is the default setting.

5

CleanSQLData

Summary

Sets the file name for the CleanSQLData file.

Usage

```
CleanSQLData=<cleandata sql file name>
```

Description

Sets the file name for the CleanSQLDate file.
This setting is case sensitive.

Examples

```
CleanSQLData=cleandata.sql
```

This is the default setting

5

SQLSchema

Summary

Sets the file name for the SQLSchema file.

Usage

```
SQLSchema=<SQL schema file name>
```

Description

Sets the file name for the SQLSchema file.
This setting is case sensitive.

Examples

```
SQLSchema=kernel_schema.sql
```

This is the default setting.

5

DemoSQL

Summary

Sets the file name for the DemoSQL file.

Usage

```
DemoSQL=<DemoSql file name>
```

Description

Sets the file name for the DemoSQL file.
This setting is case sensitive.

Examples

```
DemoSQL=demokernel.sql
```

This is the default setting.

5

CleanSQL

Summary

Sets the file name for the CleanSQL file.

Usage

```
CleanSQL=<CleanSQL file name>
```

Description

Sets the file name for the CleanSQL file.
This setting is case sensitive.

Examples

```
CleanSQL=kernel_clean.sql
```

This is the default setting.

5

DefaultPassword

Summary

Sets the default database password.

Usage

```
DefaultPassword=<password>
```

Description

The actual password needs to be used as value.
This setting is case sensitive.

Examples

```
DefaultPassword=mypassword
```

5

DefaultUser

Summary

Sets the default database user name.

Usage

```
DefaultUser=<name user>
```

Description

Sets the default database user name.
This setting is case sensitive.

Examples

```
DefaultUser=root
```

this is the default setting

5

DefaultName

Summary

Sets the default database name.

Usage

```
DefaultName=<name>
```

Description

Sets the default database name.
This setting is case sensitive.

Examples

```
DefaultName=ezpublish3
```

This is the default setting.

5

DefaultPort

Summary

Sets the default port name.

Usage

```
DefaultPort=<port number>
```

Description

Sets the default port name.
This setting is left empty by default.
This setting is case sensitive.

Examples

```
DefaultPort=3306
```

5

DefaultServer

Summary

Sets the default server name.

Usage

```
DefaultServer=<name server>
```

Description

Sets the default server name.
This setting is case sensitive.

Examples

```
DefaultServer=localhost
```

This is the default setting

5

[imagemagick program]

unix ExcludePaths   (page)
     Sets the exclude paths for Unix.

unix Executable   (page)
     Sets the name for the image executable for Unix.

unix SearchPaths   (page)
     Sets the search path for Unix.

win32 ExcludePaths   (page)
     Sets the exclude path for win32.

win32 Executable   (page)
     Sets the name for the image executable for win32.

win32 SearchPaths   (page)
     Sets the search path for win32.

5

win32_ExcludePaths

Summary

Sets the exclude path for win32.

Usage

```
win32_ExcludePaths=<path>
```

Description

Sets the exclude path for win32. If there is more than one path, use ";" to separate them. An example of syntax in such a case is:

```
win32_ExcludePaths=<path>;<path>;<path>
```

This setting is case sensitive.

Examples

```
win32_ExcludePaths=convert.exe@Windows/system32
```

This is the default setting.

5

win32_SearchPaths

Summary

Sets the search path for win32.

Usage

```
win32_SearchPaths=<path>
```

Description

Sets the search path for win 32. If there is more than one path, use ";" to separate them. An example of syntax in such a case is:

```
win32_SearchPaths=<path>;<path>;<path>
```

By default this setting is left empty.
This setting is case sensitive.

5

win32_Executable

Summary

Sets the name for the image executable for win32.

Usage

```
win32_Executable=<name executable>
```

Description

Sets the name for the image executable for win32.
If there is more than one executable, use ";" to separate them. An example of syntax in such a case is:

```
unix_Executable=<executable>;<executable>
```

This setting is case sensitive.

Examples

```
win32_Executable=convertim.exe;convert.exe
```

This is the default setting.

5

unix_ExcludePaths

Summary

Sets the exclude paths for Unix.

Usage

```
unix_ExcludePaths=<path>
```

Description

Sets the exclude path for Unix. If there is more than one path, use ";" to separate them. An example of syntax in such a case is:

```
unix_ExcludePaths=<path>;<path>;<path>
```

By default this setting is left empty.
This setting is case sensitive.

5

unix_SearchPaths

Summary

Sets the search path for Unix.

Usage

```
unix_SearchPaths=<path>
```

Description

Sets the search path for for Unix. If there is more than one path, use ";" to separate them. An example of syntax in such a case is:

```
unix_SearchPaths=<path>;<path>;<path>
```

This setting is case sensitive.

Examples

```
unix_SearchPaths=/bin;/sbin;/usr/bin;/usr/sbin;/usr/local/bin;/usr/local/sbin
```

This is the default setting.

5

unix_Executable

Summary

Sets the name for the image executable for Unix.

Usage

```
unix_Executable=<name executable>
```

Description

Sets the name for the image executable for Unix.
Possible values are convert or convertim
This setting is case sensitive.

Examples

```
unix_Executable=convert
```

This is the default setting.

5

[dom_extension]

This block contains settings regarding the DOM extension tests. The DOM extension allows you to operate on XML documents through the DOM API with PHP 5.

For more information please visit http://be.php.net/dom

Extensions   (page 2274)
> Sets the required DOM extension that need to be tested during setup.

Require   (page 2273)
> sets the test requirements.

5

Require

Summary

sets the test requirements.

Usage

```
Require=one | all
```

Description

Some DOM extension are required for eZ Publish to work correctly. These extensions are listed in the setting "Extensions" in the same block. During the setup, tests are preformed to check if the extensions listed exist. This setting defines whether the existence of only one of these extensions is required for the test to succeed, or whether they all should exist.
Possible values are:

- one: only one of the extensions listed must exist for the test to be successful.

- all: all of the extensions listed must exist for the test to be successful.

This setting is case sensitive.

Examples

```
Require=all
```

All of the listed extensions in the setting "Extensions" in the same block are required for the setup test to be a success.
This is the default setting

5

Extensions

Summary

Sets the required DOM extension that need to be tested during setup.

Usage

```
Extensions=<name DOM extension>
```

Description

Some DOM extension are required for eZ Publish to work correctly. During setup it is tested if these extensions exist. This setting lists the extensions that need to be tested. The setting "Require" in the same block defines whether only one of these extensions must exist, or whether all of them should.
This setting is case sensitive.

Examples

```
Extensions=dom
```

This is the default setting.

5

[zlib_extension]

This block contains settings regarding the Zlib extension tests. The Zlib module enables you to transparently read and write gzip (.gz) compressed files, through versions of most of the file system functions which work with gzip-compressed files (and uncompressed files, too, but not with sockets).

For more information please visit http://be2.php.net/zlib

Extensions   (page 2277)
   Sets the required zlib extensions that need to be tested during setup.

Require   (page 2276)
   sets the test requirements.

5

Require

Summary

sets the test requirements.

Usage

```
Require=one | all
```

Description

Some zlib extensions are required for eZ Publish to work correctly. These extensions are listed in the setting "Extensions" in the same block. During the setup, tests are preformed to check if the extensions listed exist. This setting defines whether the existence of only one of these extensions is required for the test to succeed, or whether they all should exist.
Possible values are:

- one: only one of the extensions listed must exist for the test to be successful.

- all: all of the extensions listed must exist for the test to be successful.

This setting is case sensitive.

Examples

```
Require=all
```

All of the listed extensions in the setting "Extensions" in the same block are required for the setup test to be a success.
This is the default setting.

5

Extensions

Summary

Sets the required zlib extensions that need to be tested during setup.

Usage

```
Extensions=<name zlib extension>
```

Description

Some zlib extensions are required for eZ Publish to work correctly. During setup it is tested if these extensions exist. This setting lists the extensions that need to be tested. The setting "Require" in the same block defines whether only one of these extensions must exist, or whether all of them should.
This setting is case sensitive.

Examples

```
Extensions=zlib
```

This is the default setting.

5

[texttoimage_functions]

Functions   (page 2280)
    Sets the required texttoimage functions that need to be tested during setup.

Require   (page 2279)
    Sets the test requirements.

5

Require

Summary

Sets the test requirements.

Usage

```
Require=one | all
```

Description

Some texttoimage functions are required for eZ Publish to work correctly. These functions are listed in the setting "Functions" in the same block. During the setup, tests are preformed to check if the functions listed exist. This setting defines whether the existence of only one of these functions is required for the test to succeed, or whether they all should exist.
Possible values are:

- one: only one of the functions listed must exist for the test to be successful.

- all: all of the functions listed must exist for the test to be successful.

This setting is case sensitive.

Examples

```
Require=all
```

All of the listed functions in the setting "Functions" in the same block are required for the setup test to be a success.

This is the default setting.

5

Functions

Summary

Sets the required texttoimage functions that need to be tested during setup.

Usage

```
Extensions=<name texttoimage functions>
```

Description

Some texttoimage functions are required for eZ Publish to work correctly. During setup it is tested if these functions exist. This setting lists the function that need to be tested. The setting "Require" in the same block defines whether only one of these functions must exist, or whether all of them should.
This setting is case sensitive.

Examples

```
Functions=imagettftext;imagettfbbox
```

This is the default setting.

5

[image_conversion]

Require   (page 2282)
  sets the test requirements.

TestList   (page 2283)
  This setting lists the tests that need be run during setup regarding image conversion.

5

Require

Summary

sets the test requirements.

Usage

```
Require=one | all
```

Description

Some image_conversion extensions are required for eZ Publish to work correctly. These extensions are listed in the setting "TestList" in the same block. During the setup, tests are preformed to check if the extensions listed exist. This setting defines whether the existence of only one of these extensions is required for the test to succeed, or whether they all should exist.
Possible values are:

- one: only one of the extensions listed must exist for the test to be successful.

- all: all of the extensions listed must exist for the test to be successful.

This setting is case sensitive.

Examples

```
Require=one
```

Only one of the listed test in the setting "TestList" in the same block are required for the setup test to be a success.
This is the default setting.

5

TestList

Summary

This setting lists the tests that need be run during setup regarding image conversion.

Usage

```
TestList=<name test>
```

Description

This setting lists the tests that need be run during setup regarding image conversion.

Examples

```
TestList=imagegd_extension;imagemagick_program
```

This is the default setting.

5

[database_all_extensions]

Extensions   (page 2286)
> Sets the required database extensions that need to be tested during setup.

Require   (page 2285)
> Sets the test requirements.

5

Require

Summary

Sets the test requirements.

Usage

```
Require=one | all
```

Description

Some database extensions are required for eZ Publish to work correctly. These extensions are listed in the setting "Extensions" in the same block. During the setup, tests are preformed to check if the extensions listed exist. This setting defines whether the existence of only one of these extensions is required for the test to succeed, or whether they all should exist.
Possible values are:

- one: only one of the extensions listed must exist for the test to be successful.

- all: all of the extensions listed must exist for the test to be successful.

This setting is case sensitive.

Examples

```
Require=all
```

All of the listed extensions in the setting "Extensions" in the same block are required for the setup test to be a success.
This is the default setting

5

Extensions

Summary

Sets the required database extensions that need to be tested during setup.

Usage

```
Extensions=<name database extension>
```

Description

Some database extensions are required for eZ Publish to work correctly. During setup it is tested
if these extensions exist. This setting lists the extensions that need to be tested. The setting
"Require" in the same block defines whether only one of these extensions must exist, or whether
all of them should.
This setting is case sensitive.

Examples

```
Extensions=mysql;pgsql;mysqli
```

This is the default setting

5

[database_extensions]

Extensions (page 2289)
> Sets the required database extensions that need to be tested during setup.

Require (page 2288)
> Sets the test requirements.

5

Require

Summary

Sets the test requirements.

Usage

```
Require=one | all
```

Description

Some database extensions are required for eZ Publish to work correctly. These extensions are
listed in the setting "Extensions" in the same block. During the setup, tests are preformed to
check if the extensions listed exist. This setting defines whether the existence of only one of
these extensions is required for the test to succeed, or whether they all should exist.
Possible values are:

- one: only one of the extensions listed must exist for the test to be successful.

- all: all of the extensions listed must exist for the test to be successful.

This setting is case sensitive.

Examples

```
Require=one
```

Only one of the listed extensions in the setting "Extensions" in the same block are required for
the setup test to be a success.
This is the default setting

5

Extensions

Summary

Sets the required database extensions that need to be tested during setup.

Usage

```
Extensions=<name database extension>
```

Description

Some database extensions are required for eZ Publish to work correctly. During setup tests are run to check if these extensions exist. This setting lists these extensions that need to be tested. The setting "Require" in the same block defines whether only one of these extensions must exist, or whether all of them should.
This setting is case sensitive.

Examples

```
Extensions=mysql;pgsql;mysqli
```

This is the default setting

5

[file_upload]

This block contains information regarding the file upload test that need to be run during setup.

Require   (page)
    Sets the test requirements.

5

Require

Summary

Sets the test requirements.

Usage

```
Require=one | all
```

Description

Some file upload test need to be run during setup. This setting defines whether all requirements need to be available or only one of them, for the test to be a success.
Possible values are:

- one: only one requirement must be met for the test to be successful.

- all: all of the requirements must be met for the test to be successful.

This setting is case sensitive.

Examples

```
Require=all
```

This is the default setting.

5

[open_basedir]

Require   (page)
    Sets the test requirements.

5

Require

Summary

Sets the test requirements.

Usage

```
Require=one | all
```

Description

Some file upload test need to be run during setup. This setting defines whether all requirements need to be available or only one of them, for the test to be a success.
Possible values are:

- one: only one requirement must be met for the test to be successful.

- all: all of the requirements must be met for the test to be successful.

This setting is case sensitive.

5

[settings_permission]

CheckList   (page)
   Sets where to check for settings permissions during the setup.

5

CheckList

Summary

Sets where to check for settings permissions during the setup.

Usage

```
CheckList=<folders>
```

Description

Sets which folders to check for settings permissions during the setup.
This setting is case sensitive.

Examples

```
CheckList=settings
```

This is the default setting.

5

[directory_permissions]

CheckList   (page 2297)
    Sets where to check for directory permissions during the setup.

5

CheckList

Summary

Sets where to check for directory permissions during the setup.

Usage

```
CheckList=<folders>
```

Description

Sets which folders need to be checked for directory permissions during the setup.
This setting is case sensitive.

Examples

```
CheckList=design;extension;settings;settings/override;settings/siteaccess;
settings/siteaccess/admin;var;var/cache;var/cache/codepages;var/cache/content;
var/cache/ini;var/cache/override;var/cache/template;var/cache/template/process;
var/cache/template/tree;var/cache/texttoimage;var/cache/translation;var/log;
var/storage;var/storage/original;var/storage/reference;var/storage/variations;
var/autoload
```

This is the default setting.

5

[execution_time]

MinExecutionTime   (page)
Sets the minimum execution time required for the setup.

5

MinExecutionTime

Summary

Sets the minimum execution time required for the setup.

Usage

```
MinExecutionTime=<integer>
```

Description

Sets the minimum execution time required for the setup.

Examples

```
MinExecutionTime=30
```

"30" is the default value for this setting.

5

[memory_limit]

Defines the memory limit required during setup.

MinMemoryLimit   (page 2301)
   Sets the minimum memory limit required for the setup.

5

MinMemoryLimit

Summary

Sets the minimum memory limit required for the setup.

Usage

```
MinMemoryLimit=<memoryamount set in Megabytes>
```

Description

Sets the minimum memory limit required for the setup.
This setting is case sensitive.

Examples

```
MinMemoryLimit=64M
```

"64M" is the default value for this setting.

5

[php_session]

This block contains settings that regard to the required PHP sessions.

Extensions   (page 2304)
    Sets the PHP required session extensions.

Require   (page 2303)
    Sets the level of requirement for the PHP extensions.

Require

Summary

Sets the level of requirement for the PHP extensions.

Usage

```
Require=one | all
```

Description

Some PHP extensions are required for eZ Publish to work correctly. These extensions are listed in the setting "Extensions" in the same block. During setup tests are preformed to check if the extensions listed exist. This setting defines whether the existence of only one of these extensions is required for the test to succeed, or whether they all should exist.
Possible values are:

- one: only one of the extensions listed must exist for the test to be successful.

- all: all of the extensions listed must exist for the test to be successful.

This setting is case sensitive.

Examples

```
Require=one
```

"one" is the default value for this setting.

5

Extensions

Summary

Sets the PHP required session extensions.

Usage

```
Extensions=<name PHP extension>
```

Description

Some PHP extensions are required for eZ Publish to work correctly. During setup it is tested if these extensions exist. This setting lists the extensions that need to be tested. The setting "Require" in the same block defines whether only one of these extensions must exist, or whether all of them should.
This setting is case sensitive.

Examples

```
Extensions=session
```

"session" is the default value for this setting.

5

[phpversion]

Setting regarding the required PHP version.

MinimumVersion   (page 2307)
     Sets the minimum PHP version required.

UnstableVersions   (page 2306)
     Sets the amount of unstable PHP versions.

5

UnstableVersions

Summary

Sets the amount of unstable PHP versions.

Usage

```
UnstableVersions=<integer>
```

Description

Sets the amount of unstable PHP versions.

Examples

```
UnstableVersions=6
```

5

MinimumVersion

Summary

Sets the minimum PHP version required.

Usage

```
MinimumVersion=<version number>
```

Description

This setting defines the minimum PHP version required for using this eZ Publish installation. For more information regarding PHP visit http://php.net

Examples

```
MinimumVersion=5.1
```

This means that the minimum version of PHP required for this eZ Publish installation in 5.1

5

[iconv_extension]

This block contains the settings regarding the iconv extension setup test. For more information please visit http://be.php.net/iconv

Extensions   (page 2310)
    Sets the required iconv extensions that need to be tested during setup.

Require   (page 2309)
    sets the test requirements.

5

Require

Summary

sets the test requirements.

Usage

```
Require=one | all
```

Description

Some extensions are required for eZ Publish to work correctly. These extensions are listed in
the setting "Extensions" in the same block. During the setup, tests are preformed to check if the
extensions listed exist. This setting defines whether the existence of only one of these extensions
is required for the test to succeed, or whether they all should exist.
Possible values are:

- one: only one of the extensions listed must exist for the test to be successful.

- all: all of the extensions listed must exist for the test to be successful.

This setting is case sensitive.

Examples

```
Require=all
```

All of the listed extensions in the setting "Extensions" in the same block are required for the setup
test to be a success.
This is the default setting.

5

Extensions

Summary

Sets the required iconv extensions that need to be tested during setup.

Usage

```
Extensions=<name PHP extension>
```

Description

Some PHP extensions are required for eZ Publish to work correctly. During setup it is tested if these extensions exist. This setting lists the extensions that need to be tested. The setting "Require" in the same block defines whether only one of these extensions must exist, or whether all of them should.
This setting is case sensitive.

Examples

```
Extensions=iconv
```

This is the default setting

5

[curl_extension]

This block contains the settings regarding the setup test regarding Curl. For more information please visit http://be.php.net/curl

Extensions   (page 2313)
> Sets the required Curl extensions that need to be tested during setup.

Require   (page 2312)
> sets the test requirements.

5

Require

Summary

sets the test requirements.

Usage

```
Require=one | all
```

Description

Some extensions are required for eZ Publish to work correctly. These extensions are listed in the setting "Extensions" in the same block. During the setup, tests are preformed to check if the extensions listed exist. This setting defines whether the existence of only one of these extensions is required for the test to succeed, or whether they all should exist.
Possible values are:

- one: only one of the extensions listed must exist for the test to be successful.

- all: all of the extensions listed must exist for the test to be successful.

This setting is case sensitive.

Examples

```
Require=all
```

All of the listed extensions in the setting "Extensions" in the same block are required for the setup test to be a success.
This is the default setting.

5

Extensions

Summary

Sets the required Curl extensions that need to be tested during setup.

Usage

```
Extensions=<name Curl extension>
```

Description

Some Curl extensions are required for eZ Publish to work correctly. During setup it is tested if these extensions exist. This setting lists the extensions that need to be tested. The setting "Require" in the same block defines whether only one of these extensions must exist, or whether all of them should.
This setting is case sensitive.

Examples

```
Extensions=curl
```

This is the default setting

5

[ezcversion]

The settings in this block are used to check for the correct eZ Components version by checking the existence of class and function. At the moment it isn't possible to get version information directly, because eZ Components does not have an API to get version information.

This configuration group was added to provide additional tests for the setup wizard.

MinimumVersion (page 2315)
> Sets the minimum eZ Components version required.

TestClass (page 2317)
> Tests the eZ Components class during setup.

TestFunction (page 2316)
> Testing the eZ Components function during setup.

5

MinimumVersion

Summary

Sets the minimum eZ Components version required.

Usage

```
MinimumVersion=<version number>
```

Description

Sets the minimum eZ Components version required for this eZ Publish installation.

Examples

```
MinimumVersion=2009.2
```

The minimum eZ Components version required to set up this eZ Publish installation is 2009.2.

5

TestFunction

Summary

Testing the eZ Components function during setup.

Usage

```
TestFunction=<function>
```

Description

This setting tests if the eZ Components version installed is greater than required version, by testing the function.
The value "walkrecursive" is the default value, for more information regarding "walkrecursive" please visithttp://ezcomponents.org/docs/api/latest/Base/ezcBaseFile.html#walkRecursive

This setting is case sensitive.

Examples

```
TestFunction=walkrecursive
```

This is the default setting.

5

TestClass

Summary

Tests the eZ Components class during setup.

Usage

```
TestClass=<class>
```

Description

Some tests need to be preformed during setup to check if the eZ Components version installed is greater than required version, this is done by testing the class.
The value "ezcBaseFile" is the default value. For more information on ezcBaseFile visit http://ezcomponents.org/docs/api/latest/Base/ezcBaseFile.html

This setting is case sensitive.

Examples

```
TestClass=ezcBaseFile
```

This is the default setting.

5

5.14.34   shopaccount.ini

The configuration blocks are documented in the following sections:

- [AccountSettings] (page)

5

[AccountSettings]

Handler   (page 2320)
       Controls which shop account handler that will be used.

5

Handler

Summary

Controls which shop account handler that will be used.

Usage

```
Handler=ezuser|ezdefault|ezsimple
```

Description

This directive controls how the webshop system will get and store the information about the user when he/she initiates the checkout process. More generally, this directive sets which shop account handler that will be used. A shop account handler is a PHP class providing a mechanism that handles and stores customer information for webshop orders. The system behavior during the checkout process will depend on which shop account handler that is used. The default value of this setting is "ezuser", which means that users do not need to be logged in when making their orders.

There are three built-in shop account handlers called "ezuser", "ezdefault" and "ezsimple". These are located in the "kernel/classes/shopaccounthandlers" directory. The following table describes the built-in shop account handlers and how the checkout process behaves when the different handlers are selected.

Handler	Description	Details of the checkout process
ezuser	Users do not need to be logged in when making orders (the checkout form will be used).	When the "Checkout" button is clicked, the system will bring up the checkout form asking the user to provide necessary information for the order (name, email, country and address of the customer, plus optional company name and comment). After that, the "Confirm order" interface will be displayed. Note that if the user is logged in, then some of the fields in the checkout form (e.g. "First Name") will be automatically pre-filled according to the user's account data (fetched from the content object representing the user).

5

		In order to use this shop account handler, you need to make sure that anonymous users are granted access to the "buy" function of the "shop" module.
ezdefault	Users need to be logged in when making orders.	When the "Checkout" button is clicked, the system will check whether the user is logged in or not. If yes, the system will bring up the "Confirm order" interface. Otherwise, the user will be suggested to log in first and then (if successful) the "Confirm order" interface will be displayed.
ezsimple	Users need to fill in the shop account registration form when making orders.	When the "Checkout" button is clicked, the system will bring up the "Register account information" form asking the user to provide necessary information for creating a new shop account (name, email and post address of the customer). After that, the "Confirm order" interface will be displayed. Note that logged in users can make orders for other people by providing someone else's contact details in the shop account registration form. The system will not compare the information provided in the registration form with the currently logged in user's actual account data. In order to use this shop account handler, you need to make sure that anonymous users are granted access to the "buy" function of the "shop" module. It is not recommended to use the "ezsimple" handler if you

5

charge different VAT (page 375) percentage depending on the country the customer lives in. The main reason is that users are not supposed to specify their country while filling the "Register account information" form; and thus the system will not be able to calculate the amount of VAT for users that are not logged in (they will get an error message).

5

5.14.35   site.ini

The configuration blocks are documented in the following sections:

\begin{itemize} \item{} [BackwardCompatibilitySettings]\refer{label27a6e31f2710223363ac6a0f3efa5cf0}
\item{} [ContentSettings]\refer{label7b699c243359cf00abe4ebb9f1c483e0}
\item{} [DatabaseSettings]\refer{label7aaf05641fd73bdd0aee2202fb212348}
\item{} [DebugSettings]\refer{labelele5c6eea257f488930ace42ae615f6477}
\item{} [DesignSettings]\refer{labelc9cdc7c9889eadce2c50fa696dd9395b}
\item{} [ExtensionSettings]\refer{label7c17ae037eb788283ce753c087ec00b2}
\item{} [FileSettings]\refer{label9ca93550e6a12d0c23417b71e409de8a} \item{}
[FormProcessSettings]\refer{labelaabfd111d0bc8c56c8dbb9f6381357f0} \item{}
[HTTPHeaderSettings]\refer{label68b594a192cd9706b4768f8e673f51b9} \item{}
[InformationCollectionSettings]\refer{label9d730e361404f5eff7dd04181f1bdef6}
\item{} [MailSettings]\refer{label3366c502d3335355b267f58bafd42aa4}
\item{} [OverrideSettings]\refer{labeld481e25b2ab1b9c4c288c1cd5d6dc903}
\item{} [PortAccessSettings]\refer{label601dae0bec9f46a6abc4d2f10b9bacac}
\item{} [ProxySettings]\refer{label11ae32786273b50226db7a8867bd446d}
\item{} [RegionalSettings]\refer{labelff82f39041a5cd6c053e042d8de5d6b8}
\item{} [RoleSettings]\refer{label3b1c7cb459d88825020626c434df23e6}
\item{} [RSSSettings]\refer{label3074feb957be5fa2bc35faf18313f358}
\item{} [SearchSettings]\refer{label2dc60e590c06b5fe7d97dc0f4071a788}
\item{} [Session]\refer{label8fdd6d2abafefcb029fc7cfb0804d901} \item{}
[SetupSettings]\refer{labeleb035c6c3efd3cb5fb843515c0935a6a} \item{}
[ShopSettings]\refer{label0bf54cab3465a43167968b3356186cde} \item{}
[SiteAccessRules]\refer{label8539adf164dad53d66676fb60d4b1ab7} \item{}
[SiteAccessSettings]\refer{label8ba094c2c56016362be9e54c30ec8892} \item{}
[SiteSettings]\refer{label6d61cc9d77fb41a0027c5471d594aadd} \item{}
[SSLZoneSettings]\refer{label9837eedcc0935bf0e0abf912b96d82b0} \item{}
[TemplateSettings]\refer{labeld29b2e4c3aac8b08d6f8ef50c031fa9d} \item{}
[TimeZoneSettings]\refer{labeldebe92bad8db18cbce8ed7ccefeed1b7} \item{}
[TipAFriend]\refer{label2c38800fa60aa4b51b3686448426f6da} \item{}
[UnitSettings]\refer{labeld4ba4ffbf15370a47d8c1846718e40e9} \item{}
[URLTranslator]\refer{label6245ea809bb1015c536271e958072abb} \item{}
[UserSettings]\refer{labelc04083922deace23fb1e69c6aa63609e} \end{itemize}

5

[BackwardCompatibilitySettings]

ObjectRelationTyped (page 2325)
    Controls how relations of the "XML linked" type are handled.

5

ObjectRelationTyped

Summary

Controls how relations of the "XML linked" type are handled.

Usage

```
ObjectRelationTyped=enabled|disabled
```

Description

In eZ Publish versions prior to 3.9, object relations of the "XML linked" type are not supported. From 3.9, a relation of the "XML linked" type is created automatically when an internal link is inserted into an attribute represented by the "XML block" datatype. The "ObjectRelationTyped" setting controls whether the system should include relations of the "XML linked" type when fetching/inspecting all relations. This directive is enabled by default. If disabled, the system will skip relations of the "XML linked" type when fetching/inspecting all relations. In particular, related objects that have relations of the "XML linked" type will not be shown in the "Related objects" window when you edit content, nor will they be present in the list of related objects when an object/node is being viewed from within the administration interface.

5

[ContentSettings]

CacheDir   (page 2338)
    Sets the directory where eZ publish stores view cache files.

CachedViewModes   (page 2336)
    Sets which of the content view modes that use view cache.

CachedViewPreferences   (page 2334)
    Sets the user preferences each view mode depends on.

CacheThreshold   (page 2331)
    Sets the treshold for content cache cleanup

ComplexDisplayViewModes   (page 2332)
    Sets content views that have their content cache expired whenever an object is published.

EditDirtyObjectAction   (page 2330)
    Sets what eZ Publish should do if the user tries to edit a page that has a draft that is newer than the published version.

PreCacheSiteaccessArray   (page 2328)
    Sets the siteaccesses that will have view cache created when an object is published.

PreViewCache   (page 2329)
    Sets if eZ publish should generate the view cache when an object is published.

PreviewCacheUsers   (page 2327)
    Sets the users that should have view cache generated when an object is published.

StaticCache   (page 2333)
    Sets if static caching should be enabled or not.

ViewCaching   (page 2337)
    Sets if view caching should be enabled or not.

5

PreviewCacheUsers

Summary

Sets the users that should have view cache generated when an object is published.

Usage

```
PreviewCacheUsers[]=anonymous|current|user_id1
PreviewCacheUsers[]=anonymous|current|user_id2
...
```

Description

By default eZ publish will generate view cache for the anonymous user. Using this setting you can specify additional user IDs that eZ publish will generate view cache for.
You must enable PreViewCache (page 2329) for this setting to have any effect.

Examples

```
PreviewCacheUsers[]=anonymous
PreviewCacheUsers[]=current
PreviewCacheUsers[]=23
```

Using these settings eZ publish will generate view cache for the anonymous user, the current user and the user with ID 23 when an object is published.

5

PreCacheSiteaccessArray

Summary

Sets the siteaccesses that will have view cache created when an object is published.

Usage

```
PreCacheSiteaccessArray[]=siteaccess1
PreCacheSiteaccessArray[]=siteaccess2
...
```

Description

You must enable PreViewCache (page 2329) for this setting to have any effect. The "Pre-CacheSiteaccessArray" setting controls which siteaccesses the view cache should be generated for when an object is published. Please refer to the "Pre-generation of view cache (page 462)" section for more information.

Examples

```
PreViewCache=enabled
PreCacheSiteaccessArray[]=admin
PreCacheSiteaccessArray[]=base
```

With these settings eZ publish will create view cache when an object is published for the admin and the base siteacesses.

5

PreViewCache

Summary

Sets if eZ publish should generate the view cache when an object is published.

Usage

```
PreViewCache=enabled|disabled
```

Description

Enabling this setting will make the publishing process a bit slower. However, the first request to the page will be a bit faster. For sites with lots of content editors you should disable this setting. Please refer to the "Pre-generation of view cache (page 462)" section for more information.

5

EditDirtyObjectAction

Summary

Sets what eZ Publish should do if the user tries to edit a page that has a draft that is newer than the published version.

Usage

```
EditDirtyObjectAction=showversions|usecurrent
```

Description

If you select *showversions* eZ Publish will display a page with all the available versions, including drafts. The user can then select which version he wants to base the new draft on.
If you select *usecurrent* eZ Publish will create a new draft based on the published version and let the user edit that.

5

CacheThreshold

Summary

Sets the treshold for content cache cleanup

Usage

```
CacheTreshold= number
```

Description

The threshold for file cleanup, if it is exceeded a global expiry is used instead. The value is calculated with the number of affected nodes * viewmodes * translations * sitedesign.
Note: This is an internal change that you should not change unless you are a developer.

5

ComplexDisplayViewModes

Summary

Sets content views that have their content cache expired whenever an object is published.

Usage

```
ComplexDisplayViewModes= view1[; view2]...
```

5

StaticCache

Summary

Sets if static caching should be enabled or not.

Usage

```
StaticCache=enabled|disabled
```

Description

Static caching means that some pages on your system will be stored and served completely in HTML with a huge speed improvement. You can only use static caching for pages that do not have any dynamic elements (e.g the pages are available to everyone and looks exactly the same to all users).

Static caching is configured in staticcache.ini (page 2601).

5

CachedViewPreferences

Summary

Sets the user preferences each view mode depends on.

Usage

```
CachedViewPreferences[viewmode1]=setting1[=defaultvalue1];
setting2[=defaultvalue2]...
CachedViewPreferences[viewmode2]=setting1[=defaultvalue1];
setting2[=defaultvalue2]...
...
```

Description

Sometimes you have conditions based on user preferences (page 1171) in the templates of cached view modes (page 2336). eZ publish needs to know the preferences you use per view mode in order to make sure that the caches are correct. You should use this setting if at least one of your templates (including override templates) use a condition based on a user preference.

A typical symptom of a missing CachedViewPreferences setting is if you change a preference setting and the interface is not updated until you clear the cache. Refer to "Configuring the view cache" for more information.

Note: As of eZ Publish 4.3 the setting in the admin siteaccess is disabled by default in the Administration Interface. In order to use the Preview tab, you first have to enable it in the right menu. This does not apply to eZ Publish 4.4+.

Examples

```
CachedViewPreferences[full]=mysetting;myothersetting=1
```

This example shows a site where at least one of the templates of the "full" view mode uses the mysetting and myothersetting.

```
CachedViewPreferences[full]=admin_navigation_content=1;
admin_children_viewmode=list;admin_list_limit=1
```

This example shows the default setting in the admin siteaccess in eZ Publish 4.3.

```
CachedViewPreferences[full]=admin_navigation_states=0;
admin_navigation_content=0;admin_navigation_details=0;
admin_navigation_languages=0;admin_navigation_locations=0;
admin_navigation_relations=0;admin_navigation_roles=0;
```

5

```
admin_navigation_policies=0;admin_navigation_content=0;
admin_navigation_translations=0;admin_children_viewmode=list;
admin_list_limit=1;admin_edit_show_locations=0;admin_leftmenu_width=10;
admin_url_list_limit=10;admin_url_view_limit=10;admin_section_list_limit=1;
admin_orderlist_sortfield=user_name;admin_orderlist_sortorder=desc;
admin_search_stats_limit=1;admin_treemenu=1;admin_bookmarkmenu=1;
admin_left_menu_width=13
```

If you want to install eZ Publish 4.3 with the old admin design, you have to use the above viewcache setting in the admin siteaccess.

5

CachedViewModes

Summary

Sets which of the content view modes that use view cache.

Usage

```
CachedViewModes=view1[;view2][;view3]...
```

Description

This setting enables the viewcache for the specified view modes (page 173) inside the "view (page 821)" view of the content module. When view caching is enabled, the entire result of the module will be cached. The cache is stored for each possible role combination on your site. This means that your templates can have conditions based on roles even when caching is on.

In addition, you can enable view caching for the "pdf (page 802)" view of the content module by specifying "pdf" in this setting. In this case, the actual PDF file will be cached. However, note that the "pdf" view of the content module is deprecated.

Note: Do not change this setting unless you know what you are doing.

Examples

```
CachedViewModes=full;sitemap;pdf
```

This makes eZ publish use view cache on the "full" and "sitemap" content view modes ("content/ view/full" and "content/view/sitemap"). The output of the "pdf" view within the content module ("content/pdf") will also be cached.

5

ViewCaching

Summary

Sets if view caching should be enabled or not.

Usage

```
ViewCaching=enabled|disabled
```

Description

Viewcache is the terminology we use for cache that stores the complete HTML output of a view (page 146).

You can turn off view cache during development of a site to force eZ publish to render all templates on each request.

Note: Live sites should always have ViewCache enabled.

5

CacheDir

Summary

Sets the directory where eZ publish stores view cache files.

Usage

```
CacheDir=directory_name
```

Description

CacheDir is set relative to CacheDir (page 2383). This means that view cache files are stored within "var_dir/cache_dir/directory_name/" where "var_dir" is specified by the VarDir (page 2384) setting and "cache_dir" is specified by the CacheDir (page 2383) setting.

Examples

```
[FileSettings]
VarDir=var
CacheDir=cache

[ContentSettings]
CacheDir=content
```

Using these settings the viewcache will be stored in "var/cache/content/".

5

[DatabaseSettings]

\begin{description} \item[Charset]\refer{labele900d903ba0bb892c6a175b8faf7b250}\ \\ Sets the character set that eZ publish uses when communicating with the database.

\item[ConnectRetries]\refer{labelc05cc1c9a553b8437c362d9f9634cde9}\ \\ Sets the number of database connection retries.

\item[Database]\refer{labeledda1e5b9e1b82d50a0bae274909deff}\ \\ Sets the database to use when connecting to the database server.

\item[DatabaseImplementation]\refer{labelb81ad3825bd7783c32ac01c34a46c038}\ \\ Sets the type of database you are using.

\item[DatabasePluginPath]\refer{labelb57b27902ad9f8b562e764f87252f95c}\ \\ Sets the path to an external database driver.

\item[DebugTransactions]\refer{label1ed4df0d0d5824eeca55cc521498d38c}\ \\ Allows the system to store transactions, for debugging purposes.

\item[ImplementationAlias]\refer{label60040025be09a401424702e646a2191d}\ \\ Sets alias names for database implementations

\item[Password]\refer{labeld48d14bdf8b61df9da618a68e7b6af28}\ \\ Sets the password that eZ publish uses when logging in to the database.

\item[Server]\refer{labele017a9f721f501cbdd02804c56110c9c}\ \\ Sets the hostname or the IP address of the database server.

\item[SlaverServerDatabase]\refer{labelcb4a17ab42416eb88ca7c313ffe280d5}\ \\ The databases to use when logging in to the slaveservers.

\item[SlaverServerPassword]\refer{label4ea1651fb3007a749a73b3e50b2a90bb}\ \\ The passwords to use when logging in to the slaveservers.

\item[SlaverServerUser]\refer{labelda9ab661d4cab0aa5419ed82291df37d}\ \\ The usernames to use when logging in to the slaveservers.

\item[SlaveServerArray]\refer{labeld4dd39486c1321ac60dfb848187dc7e8}\ \\ The hostnames of the slaveservers to use for read queries.

\item[SlowQueriesOutput]\refer{label705bb9058da2f504de0ec163a347a33b}\ \\ Show queries that where slower than a set amount of time.

\item[Socket]\refer{label37cb675e4c8632559b4c0e4af89d936f}\ \\ Sets the socket eZ publish should use when connecting to the database.

\item[SQLOutput]\refer{labeled1c08eaa69c91503bc78bb384f7aa75}\ \\ Enables the output of SQL queries in the debug output.

\item[Transactions]\refer{label0f7d117c382b0ef667e26360c62f62f8}\ \\ Enables to make eZ publish use transactions to ensure database integrity.

\item[UseBuiltInEncoding]\refer{label33b5c5d2763f4ff31774d251d37a6b70}\ \\ Use the built in character conversion in the database if available.

\item[UsePersistentConnection]\refer{label721ef59d92429405b991340bef1bf4a6}\ \\ Controls if database connections should be kept open between eZ publish runs.

\item[User]\refer{labelb429c3972dc3da5161bf1a408e4a3d0a}\ \\ Sets the username that eZ publish uses when logging in to the database.

\item[UseSlaveServer]\refer{label91b2a8a7c42e7e0dbb6598de7ba14d47}\ \\ Enables the usage of slave database servers for read queries. (MySQL only)

\end{description}

5

DebugTransactions

Summary

Allows the system to store transactions, for debugging purposes.

Usage

```
DebugTransactions=enabled | disabled
```

Description

This setting allows the system to store transactions. This information can then be used in debug mode to search for invalid commits.

When enabled the system will store stack traces for all commits in the system. When unbalanced nested transactions are detected, it will output all commits, together with information on where they occurred.

If this setting is enabled the system will output the stored stack traces as a tree, which can be used to debug the code and help find the missing commit.

By default this setting is disabled.
This setting is case sensitive.

Examples

```
DebugTransactions=disabled
```

This is the default setting.

5

DatabasePluginPath

Summary

Sets the path to an external database driver.

Usage

```
DatabasePluginPath= path_to_database_driver
```

Description

If you want to use a custom database driver you must tell eZ publish where to find it. Use the path relative to the root of your eZ publish installation. eZ publish will search for the file "dbname" + "db.php" in that directory.

Examples

```
DatabaseImplementation=custom
DatabasePluginPath=extensions/mydbdriver/classes/
```

eZ publish will now search for the file customdb.php in the directory extensions/mydbdriver/classes/

5

DatabaseImplementation

Summary

Sets the type of database you are using.

Usage

```
DatabaseImplementation=ezmysql|ezpostgresql|ezoracle
```

Description

Set this option to "ezmysql" if you are using a MySQL database. If you are using PostgreSQL set this option to "ezpostgresql". In case you are using an Oracle database (note that the eZ Publish Extension for Oracle Database is required for this) you should specify "ezoracle" in the "DatabaseImplementation" directive.

5

SlaverServerDatabase

Summary

The databases to use when logging in to the slaveservers.

Usage

```
SlaveServerDatabase[]
SlaveServerDatabase[]= database1
SlaveServerDatabase[]= database2
...
```

Description

Specify one database name on each row of the setting. The order is significant and you have to use the same order for the settings in SlaveServerArray, SlaverServerUser and SlaverServerPassword

Examples

```
SlaverServerDatabase[]
SlaverServerDatabase[]=publishslave
SlaverServerDatabase[]=publishslave2
```

This setup will use the database publishslave in the first slave server and the database publishslave2 in the second slave server.

5

SlaverServerPassword

Summary

The passwords to use when logging in to the slaveservers.

Usage

```
SlaveServerPassword[]
SlaveServerPassword[]= password1
SlaveServerPassword[]= password2
```

Description

Specify one password on each row of the setting. The order is significant and you have to use the same order for the settings in SlaveServerArray, SlaverServerUser and SlaverServerDatabase.

Examples

```
SlaverServerPassword[]
SlaverServerPassword[]=secret
SlaverServerPassword[]=verysecret
```

This setup will use the password secret to log in to the first slave server and the password verysecret to log in to the second slave server.

5

SlaverServerUser

Summary

The usernames to use when logging in to the slaveservers.

Usage

```
SlaveServerUser[]
SlaveServerUser[]= user1
SlaveServerUser[]= user2
```

Description

Specify one username on each row of the setting. The order is significant and you have to use the same order for the settings in SlaveServerArray, SlaverServerPassword and SlaverServerDatabase

Examples

```
SlaverServerUser[]
SlaverServerUser[]=admin
SlaverServerUser[]=root
```

This setup will use the username admin to log in to the first slave server and the username root to log in to the second slave server.

5

SlaveServerArray

Summary

The hostnames of the slaveservers to use for read queries.

Usage

```
SlaveServerArray[]
SlaveServerArray[]= hostname1
SlaveServerArray[]= hostname2
...
```

Description

Specify one database server on each row of the setting. The order is significant and you have to use the same order for the settings in SlaverServerUser, SlaverServerPassword and SlaverServer-Database.

Examples

```
SlaveServerArray[]
SlaveServerArray[]=donald
SlaveServerArray[]=mickey
```

This setup will load balance read queries between the main database server and the database servers located on donald and mickey.

5

UseSlaveServer

Summary

Enables the usage of slave database servers for read queries. (MySQL only)

Usage

```
UseSlaveServer=enabled|disabled
```

Description

Set this option to enabled to make eZ Publish load balance read queries between several SQL servers. You can set the location and log-in information for the slave servers using the settings:

- SlaveServerArray
- SlaverServerUser
- SlaverServerPassword
- SlaverServerDatabase

For all other connection options the values set for the standard database will be used.

This option only works on MySQL and MySQLi setups. You can read more about how to set up MySQL replication here.

5

ImplementationAlias

Summary

Sets alias names for database implementations

Usage

```
ImplementationAlias[ aliasname ]=ezpostgresql|ezmysql
....
```

Description

The various database implementatations in eZ publish uses a database driver. These database drivers have names like ezmysql and ezpostgresql. In order to make let users use more commonly known names like mysql and postgresql you can set up aliases for the database drivers.

Don't change this setting unless you know what you are doing.

Examples

```
ImplementationAlias[]
ImplementationAlias[mysql]=ezmysql
```

This setting means that you can use the alias mysql when you want to use the ezmysql driver.

5

UsePersistentConnection

Summary

Controls if database connections should be kept open between eZ publish runs.

Usage

```
UsePersistentConnection=enabled|disabled
```

Description

Persistent connections are usually faster but have had a lot of unwanted effects in the past. Don't turn this on unless you know what you are doing.

5

SlowQueriesOutput

Summary

Show queries that where slower than a set amount of time.

Usage

```
SlowQueriesOutput= number
```

Description

The number controls how many milliseconds a query must take in order to be displayed in the debug output. 0 means that all queries will be shown.

5

SQLOutput

Summary

Enables the output of SQL queries in the debug output.

Usage

```
SQLOutput=enabled|disabled
```

Description

Debug (page 2548) must be enabled for this setting to have any effect.

5

Socket

Summary

Sets the socket eZ publish should use when connecting to the database.

Usage

```
Socket= number|disabled
```

Description

Set this option to 0 to use the default socket for the selected database.

5

UseBuiltInEncoding

Summary

Use the built in character conversion in the database if available.

Usage

```
UseBuiltInEncoding=true|false
```

Description

If this option is set to false or if the conversion is not available in the database the conversion will be performed by eZ publish at a somewhat slower speed.

5

Charset

Summary

Sets the character set that eZ publish uses when communicating with the database.

Usage

```
Charset= charset
```

Description

If this setting is left empty the setting from i18n.ini will be used. Usually this setting is sufficient.

5

Transactions

Summary

Enables to make eZ publish use transactions to ensure database integrity.

Usage

```
Transactions=enabled|disabled
```

Description

If you enable this setting eZ publish will perform all queries that naturally belong together within a transaction. This ensure that your database integrity will be kept even if the webserver crashes in the middle of the transaction. Don't turn this off unless you know what you are doing.

MySQL requires tables of the type InnoDB in order to use transactions.

5

ConnectRetries

Summary

Sets the number of database connection retries.

Usage

```
ConnectRetries= number
```

Description

This directive can be used to set the number of times eZ Publish should attempt to connect to the database if the initial attempt fails. Note that the number of actual connection attempts equals the initial attempt plus the number of retries specified by this directive. In other words, if the number of connection retries is set to 3, eZ Publish will attempt to connect to the database 4 times before giving up and generating an error message.

Examples

```
ConnectRetries=3
```

This setting makes eZ publish attempt to connect to the database four times in total. One main attempt and three retries.

5

Server

Summary

Sets the hostname or the IP address of the database server.

Usage

```
Server=hostname|ip_address
```

Description

Specify either the hostname or the IP address of the server running the eZ Publish database (may be "localhost" if the database engine and the web server are installed on the same machine).

5

Database

Summary

Sets the database to use when connecting to the database server.

Usage

```
Database= databasename
```

5

Password

Summary

Sets the password that eZ publish uses when logging in to the database.

Usage

```
User= password
```

5

User

Summary

Sets the username that eZ publish uses when logging in to the database.

Usage

```
User= username
```

5

[DebugSettings]

\begin{description} \item[Debug]\refer{label119066fd70990c844047aca8ed003375}\ \\ Sets if you want to display debug information in the rendered page or in a separate popup.

\item[DebugByIP]\refer{label0edd27a90af65b4d634ccaa084f715de}\ \\ Enables debug output for some IP addresses only. Useful when debugging live sites.

\item[DebugByUser]\refer{label506467095c7256dae67ecf3e6c2ff07d}\ \\ Sets whether debug output visibility should be controlled per user.

\item[DebugIPList]\refer{label462719e6f0c3d668d7937c42c31e0c87}\ \\ Sets the hosts that receive debug output.

\item[DebugLogOnly]\refer{labelcbbb9b5f5306d602d5b508494255c520}\ \\ Choose if you want debug strings in the debugoutput or in the log only.

\item[DebugOutput]\refer{label72952bb628aa20f104c9529f7ce9e549}\ \\ Main switch for debug output

\item[DebugRedirection]\refer{labelcc214953777f09745f43d1459da5bf18}\ \\ Enables debugging of internal and external module redirections.

\item[DebugUserIDList]\refer{labelc132cffc2ba09ed3c3b3ad5c947b979c}\ \\ Sets user IDs that will receive debug output.

\item[DisplayDebugWarnings]\refer{label7200a7db34879bde842f03fac140eae4}\ \\ Choose if debug warnings should be displayed explicitely on the top of the page or in the debug log only.

\item[ScriptDebugOutput]\refer{label5eb27386a8eb214d19efa8107913f269}\ \\ Enables debug output for PHP scripts run from the command line.

\end{description}

5

DebugUserIDList

Summary

Sets user IDs that will receive debug output.

Usage

```
DebugUserIDList[]= user_id1
DebugUserIDList[]= user_id2
```

Description

This configuration array controls which users that should see the debug output. The array must contain a list of valid user ID numbers (object ID numbers of user accounts). This setting affects all users regardless of which group they belong to. The "DebugByUser (page 2364)" setting must be enabled for this directive to work. Note that these settings ("DebugByUser" and "DebugUserIDList") work regardless of the value assigned to the master "DebugOutput" directive.

Examples

Example 1

```
[DebugSettings]
DebugByUser=enabled
DebugUserIDList[]
DebugUserIDList[]=14
DebugUserIDList[]=70
```

Only users with accounts having user/object ID "14" and "70" will see the debug output regardless if the master "DebugOutput" switch is enabled or not.

5

DebugByUser

Summary

Sets whether debug output visibility should be controlled per user.

Usage

```
DebugByUser=enabled|disabled
```

Description

This directive makes it possible to control debug output visibility per user. By default, this setting is disabled and debug output is shown to all users. If enabled, debug output will only be shown to the users that have their ID numbers listed in the "DebugUserIDList (page 2363)" configuration array. Note that this setting works regardless of the value assigned to the master "DebugOutput" directive.

Note: DebugByUser controls the debug settings in the [DebugSettings] block, first and foremost DebugOutput. It does not affect template settings.

5

DebugLogOnly

Summary

Choose if you want debug strings in the debugoutput or in the log only.

Usage

```
DebugLogOnly=enabled|disabled
```

Description

Enable this option to remove debug strings from the inline (or popup) debug output. Instead the debug strings will only be appended to the logs which are located in the directory *your_publish_root*/var/log.

5

DisplayDebugWarnings

Summary

Choose if debug warnings should be displayed explicitly on the top of the page or in the debug log only.

Usage

```
DisplayDebugWarnings=enabled|disabled
```

5

DebugRedirection

Summary

Enables debugging of internal and external module redirections.

Usage

```
DebugRedirection=enabled|disabled
```

Description

Whenever an internal redirection occurs the execution of eZ Publish will be stopped and a redirection page with a redirect button will appear. This allows you to see any errors that have occurred before the redirection takes place.

Note: In case DebugRedirection is not enabled or disabled, it is supposed to be a setting array where each item is the start of a path. If the setting array contains a path matching the start of the current path where eZ Publish is redirecting to then debug redirection will be enabled.

This setting is only useful for developers of the eZ Publish core and custom extensions.

5

Debug

Summary

Sets if you want to display debug information in the rendered page or in a separate popup.

Usage

```
Debug=inline|popup
```

Description

Inline means that the debug output will be displayed together with the page that was rendered. Popup means that a separate popup window will be displayed with the debug information. Not all browsers support debug information in a popup. Choose inline if you have any problems.

Please note that if debug is set to "popup" in a virtual host environment then the following rewrite rules must be added to the virtual host block:

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

5

DebugIPList

Summary

Sets the hosts that receive debug output.

Usage

```
DebugIPList[]
DebugIPList[]= ip1|network1
DebugIPList[]= ip2|network2
...
```

Description

The setting DebugByIP must be set to enabled to make this setting have any effect.

Examples

```
DebugIPList[]=1.2.3.4
DebugIPList[]=192.0.0.42
DebugIPList[]=192.0.0.0/27
```

These settings enable debug output for the hosts 1.2.3.4, 192.0.0.42 and the network specified by the range 192.0.0.0/27

5

DebugByIP

Summary

Enables debug output for some IP addresses only. Useful when debugging live sites.

Usage

```
DebugByIP=enabled|disabled
```

Description

The setting DebugIPList controls which hosts will receive debug output. Debug output is sent to everyone if this setting is disabled.

5

ScriptDebugOutput

Summary

Enables debug output for PHP scripts run from the command line.

Usage

```
ScriptDebugOutput=enabled|disabled
```

Description

This setting relies on DebugOutput to be enabled.

5

DebugOutput

Summary

Main switch for debug output

Usage

```
DebugOutput=enabled|disabled
```

Description

Set this switch to enabled to turn on debug output in the rendered pages. There are several options that control what kind of debug output you will get.

5

[DesignSettings]

The DesignSettings are typically overridden for each siteaccess.

AdditionalSiteDesignList   (page 2374)
    Sets the additional site designs.

SiteDesign   (page 2375)
    Sets the most significant design resource.

StandardDesign   (page 2376)
    Sets the least significant (fallback) design resource.

5

AdditionalSiteDesignList

Summary

Sets the additional site designs.

Usage

```
AdditionalSiteDesignList[]= design1
AdditionalSiteDesignList[]= design2
...
```

Description

The design resources in AdditionalSiteDesignList[] are checked for templates after the sitedesign (page 2375) but before the standard design (page 2376). The designs are checked in the order they are listed.

You can read more about how design resources are used in the StandardDesign (page 2376) setting.

5

SiteDesign

Summary

Sets the most significant design resource.

Usage

```
SiteDesign=design_name
```

Description

This design resource is considered the most significant design resource. eZ publish will always search for templates in this design first.

You can read more about how design resources are used in the StandardDesign (page 2376) setting.

5

StandardDesign

Summary

Sets the least significant (fallback) design resource.

Usage

```
StandardDesign= design_name
```

Description

The design resource set by StandardDesign is considered the least significant design resource. eZ publish will only use templates found in the standard design if the template could not be found in any of the other designs. It is generally a good idea not to change the standard design to anything different than *standard*. The design named standard comes with eZ publish and provides a default fallback template for all possible templates.

By default eZ publish will look for designs in the *design* directory in the eZ publish root. Using the setting DesignExtensions (page 1837) you can configure extensions to have designs as well.

For general information about designs and siteaccess read the SiteManagement (page 138) documentation.

The settings described here are usually overridden per siteaccess to provide a unique look.

Examples

```
StandardDesign=standard
SiteDesign=mydesign
AdditionalSiteDesignList[]=base
AdditionalSiteDesignList[]=extras
```

This setup will make eZ publish check the design directories in the following order:

1. mydesign

2. base

3. extras

4. standard

[ExtensionSettings]

ActiveAccessExtensions   (page 2378)
        Sets the extensions that are available to eZ publish per siteaccess.

ActiveExtensions   (page 2379)
        Sets the extensions that are available to eZ publish.

ExtensionDirectory   (page 2380)
        Sets the directory where extensions are located.

5

ActiveAccessExtensions

Summary

Sets the extensions that are available to eZ publish per siteaccess.

Usage

```
ActiveAccessExtensions[]= extension1
ActiveAccessExtensions[]= extension2
```

Description

Each extension will have its settings automatically loaded. This setting works similarly to the ActiveExtensions setting but is loaded after the siteaccess is loaded. This means that the settings activated will only be available in the siteaccess this setting is specified.

5

ActiveExtensions

Summary

Sets the extensions that are available to eZ publish.

Usage

```
ActiveExtensions[]= extension1
ActiveExtensions[]= extension2
...
```

Description

Each extension will have its settings automatically loaded. This setting is loaded before the siteaccesses are loaded. Overriding this setting in a siteaccess has no effect.

Examples

```
ActiveExtensions[]
ActiveExtensions[]=ezdhtml
ActiveExtensions[]=myextension
```

This setup loads the extension ezdhtml (Online Editor) and your personal extension myextension.

5

ExtensionDirectory

Summary

Sets the directory where extensions are located.

Usage

```
ExtensionDirectory= path_relative_to_ez_publish_root
```

Description

This setting is set to extension by default. Do not change it unless you know what you are doing.

5

[FileSettings]

\begin{description} \item[CacheDir]\refer{labelfd4d2195b3b6f5306f58fbf1ef4d83af}\ \\ Sets the directory where eZ publish stores cache files.

\item[DirDepth]\refer{labelaf14b2fb8ca5e71aa87b17daf72bda23}\ \\ Sets the number of extra directories that will be made when storing a file.

\item[LogDir]\refer{label8a68b0daedee70b28a9232491d4dd4ba}\ \\ Sets the directory where eZ publish will store its logfiles.

\item[StorageDir]\refer{label9f6dd283986e19498e8eaad443c7b62a}\ \\ Sets the directory eZ publish uses to store files.

\item[StorageDirPermissions]\refer{label52816e1975d9bbdd45f0aa7bcc312993}\ \\ Sets the permissions set on directories created in the storage directory.

\item[StorageFilePermission]\refer{label94dd956919d6757c8c3c9d6ea77b994a}\ \\ Sets the permissions set on files created in the storage directory.

\item[TemporaryDir]\refer{label93346dfbd4e1bccd43b91a344246284c}\ \\ Sets the directory eZ publish uses to store temporary files.

\item[TemporaryPermissions]\refer{labelec647dbcf9384aeacdf3fea333e00511}\ \\ Sets the permissions on temporary files created by eZ publish

\item[VarDir]\refer{label496cb7f52c35f1b12691721721df103a}\ \\ Sets the main directory for file storage in eZ Publish.

\end{description}

5

LogDir

Summary

Sets the directory where eZ publish will store its logfiles.

Usage

```
LogDir= directory_name
```

Description

The log files are stored within / *var_dir*/ *directory_name* where var_dir is specified by the VarDir (page 2384) setting.

Examples

```
VarDir=var
LogDir=log
```

These settings will make eZ publish use the log dir "/var/log"

5

CacheDir

Summary

Sets the directory where eZ publish stores cache files.

Usage

```
CacheDir=directory_name
```

Description

eZ publish creates many cache files to speed up execution. The cache files are stored within "var_dir/directory_name/" where "var_dir" is specified by the VarDir (page 2384) setting.

Note1: Don't change the default setting "cache" unless you know what you are doing.

Note2: Only directories relative to the eZ Publish root are supported. Absolute directories are NOT SUPPORTED. Bear in mind that using an absolute directory will be accepted, but it won't work with some features and is NOT SUPPORTED.

Warning: Assume view caching is on. The expected behavior would be that a cached node which is changed in the admin interface is removed from the cache and regenerated on first access. However, if you use different settings for CacheDir ([FileSettings]) for the site access "example" and "example_admin", then publishing a new version of a document (using example_admin) will not remove the cached node version of "example" (since the cache directories for both site accesses are different).

Examples

```
VarDir=var
CacheDir=cache
```

These settings will make eZ publish use the cache dir "var/cache/".

5

VarDir

Summary

Sets the main directory for file storage in eZ Publish.

Usage

```
VarDir=relative_path_to_directory
```

Description

The directory specified by this setting is used by eZ Publish for file storage. This includes content related files, cache files and other temporary files.

The value of this setting must be the same for all related siteaccesses (those sharing the same database). This is needed for the view cache clearing (page 453) subsystem to function properly. (Otherwise, it won't clear the view cache for related siteaccesses when it is cleared for the current one.)

Similarly, do not use the same VarDir setting for siteaccesses that use different databases.

Note1: Do not change the default value unless you know what you are doing.

Note2: Only directories relative to the eZ Publish root are supported. Absolute directories are NOT SUPPORTED. Bear in mind that using an absolute directory will be accepted, but it won't work with some features and is NOT SUPPORTED.

Examples

```
VarDir=var
```

The directory "var/" is used to store files.

5

DirDepth

Summary

Sets the number of extra directories that will be made when storing a file.

Usage

```
DirDepth= number
```

Description

Filesystems have limitations on how many files you can store within one directory. In order to avoid these problems eZ publish creates extra directories based on the first letters in the filename. This solution spreads the files over many directories. The DirDepth settings controls how many levels of extra directories eZ publish should make.
The default setting '3' support millions of files and should be enough for most sites.

Examples

```
DirDepth=3
```

If you store the file test.jpg eZ publish will store the file in the directory "t/e/s/" the complete path for the file will be "t/e/s/test.jpg".

5

StorageFilePermission

Summary

Sets the permissions set on files created in the storage directory.

Usage

```
StorageFilePermission= permission_setting
```

Description

Note that:

- It is important that the webserver has sufficient permissions to both write and remove directories.

- Settting the permissions to 0666 (read and write to all) is a potential security risk.

The preferred setting is 0660 (full read and write to user and group). This requires apache to have the correct user/group access.

5

StorageDirPermissions

Summary

Sets the permissions set on directories created in the storage directory.

Usage

```
StorageDirPermissions= permission_setting
```

Description

Note that:

- It is important that the webserver has sufficient permissions to both write and remove directories.

- Settting the permissions to 0777 (read and write to all) is a potential security risk.

The preferred setting is 0770 (full read and write to user and group). This requires apache to have the correct user/group access.

5

StorageDir

Summary

Sets the directory eZ publish uses to store files.

Usage

```
StorageDir= dir_name
```

Description

The directory name specified is relative to the var directory specified by VarDir (page 2384). The directory specified by StorageDir is used to store files related to the content of your site.

5

TemporaryPermissions

Summary

Sets the permissions on temporary files created by eZ publish

Usage

```
TemporaryPermissions= permission_setting
```

Description

The permission setting should be specified using the UNIX file permission schema.
Note that:

- It is important that the webserver has sufficient permissions to both write and remove files.

- Settting the permissions to 0777 (read and write to all) is a potential security risk.

The preferred setting is 0770 (full read and write to user and group). This requires apache to have the correct user/group access.

5

TemporaryDir

Summary

Sets the directory eZ publish uses to store temporary files.

Usage

```
TemporaryDir= dir_name
```

Description

eZ publish will use the directory name you choose inside */ var_dir/ cache_dir/ dir_name* where *var_dir* is set by VarDir (page 2384) and *cache_dir* is set by CacheDir (page 2383).

This directory is only used to store files that are used during the rendering of one page. After the page is rendered any temporary files are removed.

5

[FormProcessSettings]

Module   (page 2392)
    Sets if the form module should be enabled or not.

5

Module

Summary

Sets if the form module should be enabled or not.

Usage

```
Module=enabled|disabled
```

Description

The form module is insecure by design and should not be used.

5

[HTTPHeaderSettings]

HeaderList   (page)
> Not documented yet.

5

HeaderList

Summary

Not documented yet.

5

[InformationCollectionSettings]

EmailReceiver   (page 2396)

    Sets the receiver of e-mail generated by the information collection system.

5

EmailReceiver

Summary

Sets the receiver of e-mail generated by the information collection system.

Usage

```
EmailReceiver= e-mail_address
```

Description

Each time the information collector system receives data it is sent to the Email address specified by this setting.

5

[MailSettings]

AdminEmail   (page 2405)
   Sets the mail address of the site administrator.

AllowedCharsets   (page 2403)
   Sets the character sets that eZ Publish sends directly in mail.

ContentType   (page 2401)
   Sets the content type for email sent from eZ Publish.

EmailSender   (page 2404)
   Sets the default sender address for mail sent from eZ Publish.

HeaderLineEnding   (page 2400)
   Sets the line ending character used in emails sent from eZ Publish.

OutputCharset   (page 2402)
   Sets the character set to convert mail into if they are formatted with the wrong character set.

SendmailOptions   (page 2399)
   Sets the additional sendmail options.

Transport   (page 2410)
   Controls how eZ Publish delivers outgoing mail.

TransportConnectionType   (page 2398)
   Sets the connection type in SMTP.

TransportPassword   (page 2406)
   Sets the password to use for authentication with the SMTP server.

TransportPort   (page 2408)
   Sets the port that should be used when connecting to the SMTP server.

TransportServer   (page 2409)
   Sets the hostname of the SMTP server.

TransportUser   (page 2407)
   Sets the user to use for authentication with the SMTP server.

5

TransportConnectionType

Summary

Sets the connection type in SMTP.

Usage

```
TransportConnectionType=<empty>|ssl|sslv2|sslv3|tls
```

Description

This setting allows you to set a secure connection type to use when outgoing mail is delivered by SMTP. Make sure [MailSettings] Transport is set to SMTP.

The following cryptographic protocols can be used as value:

- ssl

- sslv2

- sslv3

- tls

By leaving this setting empty, no encryption will be used. For this reason this setting is left empty by default.

This setting is case sensitive.

Examples

```
[MailSettings]
Transport=SMTP
TransportServer=<SMTP SERVER>
TransportConnectionType=ssl
TransportPort=465
TransportUser=<username>
TransportPassword=<password>
```

The above settings are an example of how your installation needs to be configured to allow your system to send emails with an SSL connection in SMTP. Of course <SMTP SERVER>, <username> and <password> need to be changed in your own server name, user name and password. Port 465 is used since it is a secure SMTP, but this can be changed if you use another SSL port.

SendmailOptions

Summary

Sets the additional sendmail options.

Usage

```
SendmailOptions[]= option1
SendmailOptions[]= option2
...
```

Examples

```
SendmailOptions[]=-r
SendmailOptions[]=nospam@ez.no
```

The resulting option "-r nospam@ez.no" will be passed to sendmail.

5

HeaderLineEnding

Summary

Sets the line ending character used in emails sent from eZ Publish.

Usage

```
HeaderLineEnding=auto| url_encoded_value
```

Description

If you have problems with linebreaks in mail you may want to change this setting. Use URL a URL encoded value if you choose not to use the auto setting. E.g Specify Carriage Return with %0D and Line Feed with %0A.

Examples

```
HeaderLineEnding=%0A%0D
```

Sets linebreaks in e-mail to be LineFeed CarriageReturn.

5

ContentType

Summary

Sets the content type for email sent from eZ Publish.

Usage

```
ContentType=text/plain|text/html
```

Description

The default setting is text/plain which means that mails sent are written in plain text. You can change this setting into text/html if you want to send HTML formatted mail. If you do this, you must also reformat all the mail templates with HTML tags.

5

OutputCharset

Summary

Sets the character set to convert mail into if they are formatted with the wrong character set.

Usage

```
OutputCharset= characterset
```

Description

If you try to send an e-mail formatted in character set not listed in AllowedCharsets (page 2403) eZ Publish will automatically convert the mail into the characterset specified.

5

AllowedCharsets

Summary

Sets the character sets that eZ Publish sends directly in mail.

Usage

```
AllowedCharsets[]= characterset1
AllowedCharsets[]= characterset2
...
```

Description

E-mail that are not in an accepted format will be converted to the format specified by Out-
putCharset (page 2402).

Examples

```
AllowedCharsets[]
AllowedCharsets[]=us-ascii
AllowedCharsets[]=utf-8
```

With this setup eZ Publish will send all e-mail using the character sets us-ascii and utf-8 without
conversion.

5

EmailSender

Summary

Sets the default sender address for mail sent from eZ Publish.

Usage

```
EmailSender= email_address
```

Description

The specified e-mail address will be used as the default value in the from field for mail sent from eZ Publish. If this field is left blank the value in AdminEmail (page 2405) is used instead.

5

AdminEmail

Summary

Sets the mail address of the site administrator.

Usage

```
AdminEmail= email_address
```

Description

The admin email is used for notification mail for the administrator of the site. It is used throughout the system for important updates e.g when new users are created.

5

TransportPassword

Summary

Sets the password to use for authentication with the SMTP server.

Usage

```
TransportPassword= password
```

Description

You must set Transport (page 2410) to "smtp" for this setting to have any effect.

5

TransportUser

Summary

Sets the user to use for authentication with the SMTP server.

Usage

```
TransportUser= username
```

Description

If your SMTP server requires authentication you must provide a username with this setting.

You must set Transport (page 2410) to "smtp" for this setting to have any effect.

5

TransportPort

Summary

Sets the port that should be used when connecting to the SMTP server.

Description

You must set Transport (page 2410) to "smtp" for this setting to have any effect.

5

TransportServer

Summary

Sets the hostname of the SMTP server.

Usage

```
TransportServer=<hostname> | <IP adress>
```

Description

This setting allows you to set a specific SMTP server. You must set [Mailsettings] Transport to "smtp" for this setting to have any effect.

You can use both a normal hostname and an IP address. When using the hostname as value, this setting is case sensitive.

Examples

```
[MailSettings]
Transport=SMTP
TransportServer=<SMTP SERVER>
TransportConnectionType=ssl
TransportPort=465
TransportUser=<username>
TransportPassword=<password>
```

The above settings are an example of how your installation needs to be configured to allow your system to send emails with an SSL connection in SMTP. Of course <SMTP SERVER>, <username> and <password> need to be changed in your own server name, user name and password. Port 465 is used since it is a secure SMTP, but this can be changed if you use another SSL port.

5

Transport

Summary

Controls how eZ Publish delivers outgoing mail.

Usage

```
Transport=sendmail|smtp|file
```

Description

This setting controls how eZ Publish should deliver outgoing mail. There are three transport options:

- "sendmail" - direct delivery through sendmail (must be available on the server)
- "smtp" - indirect delivery using an SMTP relay server
- "file" - mail is stored on the filesystem instead of being sent

The default value of this setting is "sendmail", which means that mail is delivered directly using the sendmail transfer agent. This is available on UNIX/Linux systems only, so Windows users should probable use SMTP.

If sendmail is not available then SMTP should be used. (Make sure that SMTP server details are specified using the TransportServer (page 2409), TransportPort (page 2408), TransportUser (page 2407) and TransportPassword (page 2406) settings.)

If you need to check/view emails generated by the system without actually sending them out, set this directive to "file". Outgoing mails will be stored as separate files under the "var/log/ mail" directory of your eZ Publish installation. Note that this option is for testing and debugging purposes only and should not be used on production/live sites.

From eZ Publish 4.4 (Fuji) onwards, it is possible to send emails with an SSL connection in SMTP.

Examples

Example 1

```
[MailSettings]
Transport=SMTP
TransportServer=<SMTP SERVER>
TransportConnectionType=ssl
TransportPort=465
TransportUser=<username>
TransportPassword=<password>
```

5

The above settings are an example of how your installation needs to be configured to allow your system to send emails with an SSL connection in SMTP. Of course <SMTP SERVER>, <username> and <password> need to be changed in your own server name, user name and password. Also port 465 can be changed into any other SSL port you use.

Example 2

```
[MailSettings]
Transport=SMTP
TransportServer=<SMTP SERVER>
TransportConnectionType=
TransportPort=25
TransportUser=<username>
TransportPassword=<password>
```

The above is an example of using SMTP sending without any encryption connection. <SMTP SERVER>, <username> and <password> need to be changed in your own server name, user name and password. Also port 25 can be changed into another port you use.

5

[OverrideSettings]

Cache   (page 2413)
     Sets if the template override cache should be enabled.

Cache

Summary

Sets if the template override cache should be enabled.

Usage

```
Cache=enabled|disabled
```

Description

Note: Don't turn off the template override cache unless you know what you are doing.

5

[PortAccessSettings]

Portnumber to siteaccess mapping   (page 2415)
    Creates a mapping between a portnumber and a siteaccess.

5

Portnumber to siteaccess mapping

Summary

Creates a mapping between a portnumber and a siteaccess.

Usage

```
number1= siteaccessname1
number2= siteaccessname2
```

Description

You should have one line of this type for each port that is used to access your site.

Examples

```
80=user
81=admin
```

This setup has two siteaccesses. All requests to port 80 use the *user* siteaccess, while all requests to port 81 use the *admin* siteaccess.

5

[ProxySettings]

Password (page 2417)
> The password for outgoing web traffic through a proxy server.

ProxyServer (page 2418)
> Sets the proxy server that can be used for outgoing web traffic.

User (page 2419)
> The username for outgoing web traffic through a proxy server.

5

Password

Summary

The password for outgoing web traffic through a proxy server.

Usage

```
Password=password
```

Description

In case outgoing web traffic is only allowed through a proxy server that requires authentication, you need to provide a valid username/password combination. This directive can be used to provide the password for the proxy authentication. Note that the proxy server address and the username also need to be specified using the "ProxyServer (page 2418)" and "User (page 2419)" settings.

5

ProxyServer

Summary

Sets the proxy server that can be used for outgoing web traffic.

Usage

```
ProxyServer=<hostname|IP address>:<port>
```

Description

In case outgoing web traffic is only allowed through a proxy server, you need to inform eZ Publish about this (otherwise the system will not be able to do certain tasks). You can either use the hostname or the IP address of the proxy server, followed by a colon and a port number. Most proxy servers allow anonymous connections. However, if the proxy server requires authentication, you will also need to provide a valid username/password combination using the "User (page 2419)" and "Password (page 2417)" settings.

Note that CURL support must be enabled in PHP, otherwise outbound connections via proxy will not work.

Examples

Example 1

```
[ProxySettings]
ProxyServer=proxy.example.com:3128
```

This configuration tells the system to use an anonymous proxy server, "proxy.example.com" and port 3128 to access the web. In this case, a username/password combination is not needed.

Example 2

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

This configuration tells the system to use "proxy.example.com" and port 3128 to access the web. In addition, a username/password combination is provided because the proxy server requires authentication.

5

User

Summary

The username for outgoing web traffic through a proxy server.

Usage

```
User=username
```

Description

In case outgoing web traffic is only allowed through a proxy server that requires authentication, you need to provide a valid username/password combination. This directive can be used to provide the username for the proxy authentication. Note that the proxy server address and the password also need to be specified using the "ProxyServer" (page 2418) and "Password" (page 2417) settings.

5

[RegionalSettings]

ContentObjectLocale   (page 2431)
>   Sets the default language for content objects.

ContentXMLCharset   (page 2430)
>   Sets the characterset used when storing XML in content objects.

Debug   (page 2426)
>   Sets if debug mode should be enabled or disabled.

DevelopmentMode   (page 2427)
>   Sets if development mode should be on or off.

HTTPLocale   (page 2433)
>   Sets the locale transmitted to the web clients.

Locale   (page 2434)
>   Sets the locale (currency, date and time settings etc.)

ShowUntranslatedObjects   (page 2421)
>   Sets whether all languages are shown or not.

SiteLanguageList   (page 2423)
>   The prioritized list of site languages.

SystemLocale   (page 2432)
>   Tells PHP to be run in a specific locale.

TextTranslation   (page 2429)
>   Sets if text translation is enabled for template translation.

TranslationCache   (page 2428)
>   Sets if the translation cache should be enabled or disabled.

TranslationExtensions   (page 2424)
>   Sets the extensions that hold translations

TranslationRepository   (page 2425)
>   Sets the default translation repository for eZ publish.

5

ShowUntranslatedObjects

Summary

Sets whether all languages are shown or not.

Usage

```
ShowUntranslatedObjects=enabled|disabled
```

Description

This setting has two possible values:

1. *disabled* - Means that only languages listed in the "SiteLanguageList (page 2423)" setting are displayed.

2. *enabled* - Means that all languages are displayed. 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.

This setting is usually enabled for admin siteaccess and disabled for public siteaccess(es). Please refer to the "Configuring the site languages (page 249)" section for more information.

Examples

Example 1

```
SiteLanguageList[]
SiteLanguageList[]=eng-GB
SiteLanguageList[]=ger-DE
ShowUntranslatedObjects=disabled
```

This will tell the system that British English has the highest priority and German is the second prioritized language. Any other languages will not be shown.

Example 2

```
SiteLanguageList[]
SiteLanguageList[]=eng-GB
SiteLanguageList[]=ger-DE
ShowUntranslatedObjects=enabled
```

5

This will tell the system that British English has the highest priority and German is the second prioritized language. All other languages will still be shown (and editable) but will have less priority.

5

SiteLanguageList

Summary

The prioritized list of site languages.

Usage

```
SiteLanguageList[]=language1
SiteLanguageList[]=language2
...
```

Description

Sets which languages the contents of a site should be displayed in. 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, 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 the "ShowUntranslatedObjects (page 2421)" setting is enabled. Therefore it is important that the SiteLanguageList always contains every languages you intend to use in your site access.
Refer to the "Configuring the site languages (page 249)" section for more information and examples.

Note: If this setting is not specified then only default language will be shown.

5

TranslationExtensions

Summary

Sets the extensions that hold translations

Usage

```
TranslationExtensions[]= extension1
TranslationExtensions[]= extension2
...
```

Description

eZ Publish will search for additional translations in the "translations" directory of your extension.

This setting is commonly overridden in the extension settings to tell eZ Publish that the extension provides translations.

Examples

```
TranslationExtensions[]=myextension
```

eZ Publish will now search the directory "/extension/myextension/translations" for additional translations.

5

TranslationRepository

Summary

Sets the default translation repository for eZ publish.

Usage

```
TranslationRepository= path_to_repository
```

Description

This setting should point to the default directory for eZ publish translations. This setting should be changed by developers only.

5

Debug

Summary

Sets if debug mode should be enabled or disabled.

Usage

```
Debug=enabled|disabled
```

Description

If debug is enabled eZ publish will display information about the locale files that are loaded. You need to turn on Debug (page 2368) globally to see debug output.

5

DevelopmentMode

Summary

Sets if development mode should be on or off.

Usage

```
DevelopmentMode=enabled|disabled
```

Description

Development mode makes eZ publish translate all untranslated string using bork mode. This can be handy when spotting untranslated text.

You should never use development mode in a production environment.

5

TranslationCache

Summary

Sets if the translation cache should be enabled or disabled.

Usage

```
TranslationCache=enabled|disabled
```

Description

This setting is for debugging purposes only. You should never use eZ publish without translation cache in a production environment.

5

TextTranslation

Summary

Sets if text translation is enabled for template translation.

Usage

```
TextTranslation=enabled|disabled
```

Description

This setting controls if eZ publish should translate strings marked with i18n in the templates. eZ publish runs slightly faster when this setting is set to "disabled" This setting is automatically set to disabled if your Locale (page 2434) is set to "eng-GB".

5

ContentXMLCharset

Summary

Sets the characterset used when storing XML in content objects.

Usage

```
ContentXMLCharset=enabled|disabled| character_set
```

Description

- **enabled** - Stores XML fields using the current character set. The current character set is set by the Charset (page 1882) setting. This setting is the correct setting for most people.

- **disabled** - Forces storage in UTF8.

- **character set** - Forces storage in this character set.

5

ContentObjectLocale

Summary

Sets the default language for content objects.

Usage

```
ContentObjectLocale=locale
```

Description

Sets the default language for content objects. This language will be used as the default value in PHP functions that support an optional parameter for language, e.g. in the "eZContent-Class::instantiate()" function. The default value of this setting is "eng-GB".

Note: It is possible but not recommended to change this setting after you have run the setup wizard and added content to your site. If the specified language does not exist in eZ Publish then it will be automatically added (as a new entry in the "ezcontent_language" database table).

Examples

```
ContentObjectLocale=nor-NO
```

Sets the default language to Norwegian (Bokmal).

5

SystemLocale

Summary

Tells PHP to be run in a specific locale.

Usage

```
SystemLocale=locale_name1[,locale_name2][,locale_name3]...
```

Description

Use this directive to set locale for the entire PHP system (this functionality is similar to using the "setlocale" PHP function where "LC_ALL" is passed as the first parameter). You can specify a comma separated list of locale names. If you use a character set specific locale (for example, "no_NO.UTF-8"), make sure this character set matches the output character set of eZ Publish (page 1882). Please note that different operating systems have different naming schemes for locales, and thus you might need to use multiple names in order for your installation to work on multiple platforms (such as developing on Windows and running Linux in production).

Examples

Example 1

```
SystemLocale=de_DE.ISO-8859-1,german
```

This will tell PHP to first use the "de_DE.ISO-8859-1" locale, and if this is not available, then the "german" locale will be used.

Example 2

```
SystemLocale=no_NO.UTF-8,no_NO,norwegian
```

With this configuration each of the three listed elements (starting from "no_NO.UTF-8") will be tried to be set as new locale until success.

5

HTTPLocale

Summary

Sets the locale transmitted to the web clients.

Usage

```
HTTPLocale= locale_setting
```

Description

This setting is usually set automatically from the Locale (page 2434) setting. However you can override it for this setting. For most people the default setting (empty) is sufficient.

5

Locale

Summary

Sets the locale (currency, date and time settings etc.)

Usage

```
Locale= locale_name
```

Description

The locale controls settings related to country specific settings, e.g. language, currency and date and time formating. eZ publish provides many default locale settings in the "/share/locale" directory. Use the filename without the extension (.ini) to tell eZ publish to use that locale.

If you want custom locale settings, simply copy the locale configuration file that is the closest to the configuration you want. Then edit it and make it conform to your needs. Finally, tell eZ publish to use that locale using this setting. Please refer to the "Configuring your site locale (page 245)" section for more information about locales.

Examples

```
Locale=nor-NO
```

Sets the locale to Norwegian settings.

5

[RoleSettings]

EnableCaching   (page 2440)
    Controls whether role caching should be enabled or disabled.

MaxParentDepthLimitation   (page 2436)
    Not documented yet.

PolicyOmitList   (page 2438)
    Excludes modules and views from the permission checking.

ShowAccessDeniedReason   (page 2437)
    Sets if eZ Publish elaborates on the reason for getting access denied when viewing a page.

UserPolicyCache   (page 2439)
    Sets which users' policies to cache.

5

MaxParentDepthLimitation

Summary

Not documented yet.

5

ShowAccessDeniedReason

Summary

Sets if eZ Publish elaborates on the reason for getting access denied when viewing a page.

Usage

```
ShowAccessDeniedReason=enabled|disabled
```

Description

This option is for site debugging purposes and is disabled by default. Enabling elaborate messages can pose a security threat since details about the role setup of your system will be exposed.

5

PolicyOmitList

Summary

Excludes modules and views from the permission checking.

Usage

```
PolicyOmitList[]= module1 [/ view1]
PolicyOmitList[]= module2 [/ view2]
...
```

Description

This setting allows you to exclude complete modules or specific views from permission control. Modules and views excluded from permission control are always accessible for all users.

Examples

```
PolicyOmitList[]=ezinfo
PolicyOmitList[]=user/login
PolicyOmitList[]=user/register
```

These settings excludes all the views in the ezinfo module, and the views login and register in the user module from permission checking.

5

UserPolicyCache

Summary

Sets which users' policies to cache.

Usage

```
UserPolicyCache=enabled|disabled| id1, id2,...
```

Description

This setting has three possible values

1. **enabled** - Cache the policies of all users.

2. **disabled** - No policy cache will be stored.

3. **Comma separated list of IDs** - Store policy cache for all users with their IDs in this list.

Examples

```
UserPolicyCache=23,44,24
```

With this configuration eZ Publish will store policy cache for the users with IDs 23,24 or 44 only.

5

EnableCaching

Summary

Controls whether role caching should be enabled or disabled.

Usage

```
EnableCaching=true|false
```

Description

When role caching is enabled eZ Publish stores the complete permissions set for each user once it is computed. This saves both SQL queries and computation time on consequent requests.

Do not disable role caching unless you know what you are doing.

5

[RSSSettings]

AvailableVersionList   (page 2443)
>   Sets the available RSS versions

CacheTime   (page 2444)
>   Sets the cachetime for RSS feeds in seconds.

DefaultVersion   (page 2442)
>   Sets the default RSS version to use.

NumberOfObjectsDefault   (page 2445)
>   Sets the default value for the number of items in an RSS export.

NumberOfObjectsList   (page 2446)
>   Sets the number of items in an RSS export that the user can choose between in the admin-
>   istration interface.

5

DefaultVersion

Summary

Sets the default RSS version to use.

Usage

```
DefaultVersion= version
```

Description

This setting controls the RSS version that is selected by default when you create a new RSS export. The specified version must be found in the AvailableVersionList (page 2443) setting.

5

AvailableVersionList

Summary

Sets the available RSS versions

Usage

```
AvailableVersionList[]= version1
AvailableVersionList[]= version2
...
```

Description

This setting should only be changed by eZ Publish developers.

5

CacheTime

Summary

Sets the cachetime for RSS feeds in seconds.

Usage

```
CacheTime= number_of_seconds
```

Description

In order to minimize the server load, generated RSS feeds are cached and updated at a regular interval. This settings controls the time between each update of the cache.
Changes to the content being fed will show up in the RSS feed after the number of seconds set at the latest.

5

NumberOfObjectsDefault

Summary

Sets the default value for the number of items in an RSS export.

Usage

```
NumberOfObjectsDefault= number
```

Description

The number must be present in the NumberOfObjectsList (page 2446) setting.

5

NumberOfObjectsList

Summary

Sets the number of items in an RSS export that the user can choose between in the administration interface.

Usage

```
NumberOfObjectsList[]= number1
NumberOfObjectsList[]= number2
...
```

Description

The administration interface displays a dropdown with the numbers set by this setting in the RSS export edit screen.. The dropdown is used to determine the number of items to export in the RSS feed.

5

[SearchSettings]

AllowEmptySearch   (page 2454)
    Sets if users can search for nothing

DelayedIndexing   (page 2449)
    Sets if new content objects are indexed in the search engine upon publishing or if indexing is done by the cronjob

EnableWildcard   (page 2453)
    Sets if wildcard searching is allowed or not

ExtensionDirectories   (page 2448)
    Tells the system where to look for search engine plugins.

LogSearchStats   (page 2456)
    Sets if search statistics should be saved or not.

MaximumSearchLimit   (page 2455)
    Sets the maximum number of returned hits.

MinCharacterWildcard   (page 2452)
    The minimum number of characters a wildcard can represent

SearchEngine   (page 2458)
    Sets which search engine to use.

SearchViewHandling   (page 2457)
    Sets if searches are handled by the search view or in the template.

StopWordThresholdPercent   (page 2450)
    Sets the percentage of hits that a word should be present in before ignoring the word completely

StopWordThresholdValue   (page 2451)
    Sets the minimum number of objects in the database before the stopword functionality is used.

5

ExtensionDirectories

Summary

Tells the system where to look for search engine plugins.

Usage

```
ExtensionDirectories[]=extension1
ExtensionDirectories[]=extension2
...
```

Description

This setting can be used to specify the extension directories where eZ Publish will look for search engine plugins. By default, eZ Publish will look for plugins in the "search/plugins/" subdirectory inside your extension(s).

Examples

Example 1

If you have an extension "findme" that includes a search engine plugin called "myengine", you'll need to put the following lines into an override for the "site.ini" configuration file:

```
[SearchSettings]
ExtensionDirectories[]=findme
SearchEngine=myengine
```

eZ Publish will look for the search engine plugin at "extension/findme/search/plugins/myengine/myengine.php".

Example 2

If your site makes use of the eZ Find extension, the "site.ini.append.php" configuration file located in the "extension/ezfind/settings/" directory will contain the following lines:

```
[SearchSettings]
ExtensionDirectories[]=ezfind
SearchEngine=ezsolr
```

This will instruct the system to use the search engine plugin located at "extension/ezfind/search/plugins/ezsolr/ezsolr.php".

5

DelayedIndexing

Summary

Sets if new content objects are indexed in the search engine upon publishing or if indexing is done by the cronjob

Usage

```
DelayedIndexing=enabled|disabled
```

Description

Delaying the indexing means that some objects will not be found by the search engine even after they have been published. However, indexing can be CPU intensive and delaying the indexing can mean better performance and a more responsive administration interface for editors.

5

StopWordThresholdPercent

Summary

Sets the percentage of hits that a word should be present in before ignoring the word completely

Usage

```
StopWordThresholdPercent= number
```

Description

Searching for common words can lead to extremely many hits. In order to provide a better result the stop word system has been implemented. If a word is present in more than a set percantage of all the objects in the system then this word will be ignore when searched for.

Examples

StopWordThresholdPercent=60

If you search for a word that is present in more than 60% of all the objects then the search word will be ignored.

5

StopWordThresholdValue

Summary

Sets the minimum number of objects in the database before the stopword functionality is used.

Usage

```
StopWordThresholdValue= number
```

Description

This setting should be used together with the StopWordThresholdPercent setting.

5

MinCharacterWildcard

Summary

The minimum number of characters a wildcard can represent

Usage

```
MinCharacterWildcard= number
```

Description

This setting controls the minimum number of characters required for the wildcard to match. If MinCharacterWildcard is set to 2 and you search for "dus*" then "dust" will not be found while "duster" will.

5

EnableWildcard

Summary

Sets if wildcard searching is allowed or not

Usage

```
EnableWildcard=true|false
```

Description

When wildcard searching is enabled you can match on partial words. E.g searching for "dust*" will match both duster and dustdevil. Wildcard search requires a lot of resources and can heavily influence the performance of your site.

5

AllowEmptySearch

Summary

Sets if users can search for nothing

Usage

```
AllowEmptySearch=enabled|disabled
```

Description

Empty searches take a lot of resources and can slow down your site considerably. If you enable empty searches you must use template search view handling and make sure that your template submits enough limitations on the search.

5

MaximumSearchLimit

Summary

Sets the maximum number of returned hits.

Usage

```
MaximumSearchLimit= integer
```

Description

It is possible for the user to select the number of returned hits via a POST variable. In order to avoid very high limits resulting in high server load you can use this setting to set a cap on the number of returned hits.

Examples

MaximumSearchLimit=25

This setting will limit the maximum number of search hits to 25.

5

LogSearchStats

Summary

Sets if search statistics should be saved or not.

Usage

```
LogSearchStats=enabled|disabled
```

Description

If enabled statistics about each search are written to the database. You can view the statistics from the administration interface.

SearchViewHandling

Summary

Sets if searches are handled by the search view or in the template.

Usage

```
SearchViewHandling=default|template
```

Description

This setting has two options

- **default** - The view code does the search and passes the result to template. The template is only responsible for displaying the result.

- **template** - The template does the search and passes the result back to the view code. This way you have to "program" the search yourself providing more flexibility.

5

SearchEngine

Summary

Sets which search engine to use.

Usage

```
SearchEngine=search_engine_name
```

Description

This directive tells the system which search engine to use. The default value is "eZSearchEngine", which means that the built-in eZ Publish search engine will be used.

In previous versions of eZ Publish, the "openFts" value made it possible to enable built-in support for a PostgreSQL-based search engine called OpenFTS. This functionality was mainly for demonstration purposes. It has been deprecated and should not be used.

To add support for your favorite search engine, you can implement your own search engine plugin. Note that it is not recommended to modify the eZ Publish kernel and thus you should implement it as an extension. Make sure you specify correct values in the "SearchEngine" and "ExtensionDirectories (page 2448)" settings located in the [SearchSettings] section of the "extension/<your_extension>/settings/site.ini.append.php" configuration file.

Examples

Example 1

```
[SearchSettings]
SearchEngine=eZSearchEngine
```

This will tell the system to use default eZ Publish search mechanism.

Example 2

If your site makes use of the eZ Find extension, the "site.ini.append.php" configuration file located in the "extension/ezfind/settings/" directory will contain the following lines:

```
[SearchSettings]
ExtensionDirectories[]=ezfind
SearchEngine=ezsolr
```

This will instruct the system to use the search engine plugin located at "extension/ezfind/search/plugins/ezsolr/ezsolr.php".

[Session]

\begin{description} \item[ActivityTimeout]\refer{label9eb3648c0263505f2895e6c730bd537b}\ \\ This setting will define the number of seconds before a user is considered inactive.

\item[BasketCleanup]\refer{label728f2f8f6549a1d6e0598dbfa2e54973}\ \\ Sets how the shopping baskets for expired and removed sessions are cleaned up.

\item[BasketCleanupAverageFrequency]\refer{label85eac58dc6b5be3d96c987fe52807d4c}\ \\ Sets how often the basket cleanup cronjob will actually work when being executed.

\item[CookieDomain]\refer{label1107074af4a8bc3a918443d078064779}\ \\ This setting sets the domain for which the cookie is valid.

\item[CookieHttponly]\refer{label171436f219b3a117eb6a98416282c96c}\ \\ This setting sets whether or not scripts are allowed to access cookies.

\item[CookiePath]\refer{labelc76d45ecdcaec2ed2bd4560826ae150a}\ \\ This setting will set the path to the domain for which the cookie is valid.

\item[CookieSecure]\refer{label7c86439da57641eb45cf0ce3b3aac5ac}\ \\ This setting sets whether or not cookies are allowed to be sent over an unsecure connection.

\item[CookieTimeout]\refer{label0c6ec02754c3f035a09b1c95e8634cab}\ \\ The CookieTimeout setting sets the lifetime of the session cookie.

\item[ForceStart]\refer{label2b6fbc268e61160ac2b1de87e4fac555}\ \\ This setting will allow sessions to always start.

\item[Handler]\refer{labelc6a9cf1a410be4ef12ecf16494151db8}\ \\ Sets the handler responsible for handling sessions.

\item[SessionNameHandler]\refer{label661778b05f56c89d8549075b025e5d0b}\ \\ Sets how session names should be generated

\item[SessionNamePerSiteAccess]\refer{labela155f4bd44eca74441722ad14b77f321}\ \\ Prepends session names with the current siteaccess.

\item[SessionNamePrefix]\refer{label89c013303f066673ac6857ca8dae7a83}\ \\ Sets the prefix eZ publish should use when creating session names.

\item[SessionTimeout]\refer{labelc1bc7fbe674c8b3e324eba18726311a3}\ \\ This setting will define the number of seconds a session will last.

\end{description}

5

ActivityTimeout

Summary

This setting will define the number of seconds before a user is considered inactive.

Usage

```
ActivityTimeout=<integer>
```

Description

This setting will define the number of seconds of inactivity before a session is considered inactive and a user will consequently be logged out. 3600 seconds, the equivalent of 1 hour is standard.This setting is a DB handler specific setting and is therefore only used for the eZSessionHandlerDB is set in the setting Handler in the block [Session]. The PHP handler on the other hand will use the settings defined in php.ini for this.
This setting is case sensitive.

Examples

```
ActivityTimeout=3600
```

3600 seconds, the equivalent of 1 hour is standard.

5

BasketCleanup

Summary

Sets how the shopping baskets for expired and removed sessions are cleaned up.

Usage

```
BasketCleanup=cronjob|pageload
```

Description

If a user added some products into his basket and then stopped shopping (e.g. closed his browser window) without initiating the checkout process, the user's session will expire after a while. The eZ Publish session handling, which is based on the PHP session functionality, takes care about removing expired sessions from the database. It is also possible to remove sessions (either expired or not) using the "Setup - Sessions" part of the admin interface.

If the "BasketCleanup" setting is set to "cronjob" (default), removing a user's session from the database will not delete the shopping basket that was created during this session. These unneeded baskets can be cleaned up periodically by running the "basket_cleanup.php" cronjob.

If you specify "BasketCleanup=pageload", removing a user's session from the database (either automatically by session cleanup or manually by the site administrator) will delete the corresponding shopping basket. The "basket_cleanup.php" cronjob will not do anything. Please note that this configuration is not recommended for sites with many visitors because removing baskets on session cleanup will take a lot of time.

5

BasketCleanupAverageFrequency

Summary

Sets how often the basket cleanup cronjob will actually work when being executed.

Usage

```
BasketCleanupAverageFrequency=number
```

Description

Use this directive to specify how often the shopping baskets will actually be cleaned up when the "basket_cleanup.php" cronjob is executed. This setting is extremely useful in case you are running this cronjob frequently together with other tasks. The default value is 10, which means that baskets will be cleaned about 1 out of 10 times. If you wish to run "basket_cleanup.php" separately from other cronjobs, you can specify "BasketCleanupAverageFrequency=1".

You must set BasketCleanup (page 2461) to "cronjob" for this setting to have any effect.

5

CookieDomain

Summary

This setting sets the domain for which the cookie is valid.

Usage

CookieDomain=<string>

Description

This is an optional settings for the override php.ini session cookie settings. The CookieDomain setting sets the domain for which the cookie is valid.

In order to make cookies visible on all subdomains, the domain must be prefixed with a dot. For example if the domain in "www.php.net", ".php.net' will make cookies visible on all subdomains.

If this setting is undefined, the setting will be used as defined in the php.ini setting session.cookie_domain. For more information on all other session related setting in php.ini please visit http://www.php.net/manual/en/session.configuration.php.

This setting is left empty by default.

This setting is case sensitive.

This setting is available for eZ Publish 4.4 and above.

Examples

CookieDomain=.php.net

5

CookieHttponly

Summary

This setting sets whether or not scripts are allowed to access cookies.

Usage

```
CookieHttponly=false | true
```

Description

This is an optional settings for the override php.ini session cookie settings. The setting CookieHttponly sets whether or not scripts are allowed to access cookies.

Possible values are true and false. When set to true the browser will not allow scripts to access cookies. However, this feature is only supported on php 5.2+

If this setting is undefined, the setting will be used as defined in the php.ini setting session.cookie_httponly. For more information on all other session related setting in php.ini please visit http://www.php.net/manual/en/session.configuration.php.

This setting is case sensitive.

This setting is available for eZ Publish 4.4 and above.

Examples

```
CookieHttponly=true
```

5

CookiePath

Summary

This setting will set the path to the domain for which the cookie is valid.

Usage

```
CookiePath=<string>
```

Description

This is an optional settings for the override php.ini session cookie settings. The CookiePath setting will set the path to the domain where the cookie will work.

Use a single slash ('/') for all paths on the domain.

If this setting is undefined, the setting will be used as defined in the php.ini setting session.cookie_path. For more information on all other session related setting in php.ini please visit http://www.php.net/manual/en/session.configuration.php.

This setting is case sensitive.

This setting is available for eZ Publish 4.4 and above.

Examples

```
CookiePath=/
```

This is the default setting.

5

CookieSecure

Summary

This setting sets whether or not cookies are allowed to be sent over an unsecure connection.

Usage

```
CookieSecure= false | true
```

Description

This is an optional settings for the override php.ini session cookie settings. The setting CookieSecure sets whether or not cookies are allowed to be sent over an unsecure connection.

If set to "true" the setting CookieSecure will only allow cookies to be sent over a secure connection.

Possible values are "true" to enable the secure function and "false" to disable the secure function. False is the default setting.

If this setting is undefined, the setting will be used as defined in the php.ini setting session.cookie_secure. For more information on all other session related setting in php.ini please visit http://www.php.net/manual/en/session.configuration.php.

This setting is case sensitive.

This setting is available for eZ Publish 4.4 and above.

Examples

```
CookieSecure=false
```

This is the default setting.

CookieTimeout

Summary

The CookieTimeout setting sets the lifetime of the session cookie.

Usage

```
CookieTimeout=<integer>
```

Description

This is an optional settings for the override php.ini session cookie settings.

The CookieTimeout setting sets the lifetime of the session cookie, defined in seconds. When the cookie times out it will be removed by the client browser. Note that the maximum session duration is the minimum of this setting and the SessionTimeout setting.

If this setting is left empty or undefined, the setting will be used as defined in the php.ini setting session.cookie_lifetime. For more information on all other session related settings in php.ini please visit http://www.php.net/manual/en/session.configuration.php.

This setting is case sensitive.

Examples

```
[Session]
# Number of seconds a session will last
# 3 days is standard
SessionTimeout=259200
# Number of seconds before a session is considered inactive/logged out
# 1 hour is standard
ActivityTimeout=3600
# Number of seconds a session cookie will last,
# 0 means until browser is closed
# Leaving the field empty means to use the default PHP settings
# (session.cookie_lifetime)
# Example for 6 minutes
CookieTimeout=360
```

With this configuration, HTTP cookies will expire in six minutes. If the site visitor is inactive for six minutes without closing his browser window, the cookie will expire and thus the user's session will be ended. If the user reloads the page later then a new session will be created. However, the previous session will not be removed from the database. It will become inactive in one hour and expired in three days.

ForceStart

Summary

This setting will allow sessions to always start.

Usage

```
ForceStart=enabled | disabled
```

Description

This is a backward compatibility setting. This setting will allow sessions to always start, even if not strictly needed.

This will for instance be used for anonymous users who are just browsing the site. In this case the LastAccessesURI is remembered for these users, which is a session variable to remember the last URL in order to allow redirection after user registration or content editing.

By default it is disabled.

This setting is case sensitive.

This setting is available for eZ Publish 4.4 and above.

Examples

```
ForceStart=disabled
```

This is the default setting.

5

Handler

Summary

Sets the handler responsible for handling sessions.

Usage

```
Handler=<name handler>
```

Description

This settings defines the handler responsible for handling sessions.

When this setting is left empty, "ezpSessionHandlerPHP" is used which basically leaves session handling to PHP so that PHP session handlers like "files" (default) can be used, as defined by php.ini "session.save_handler".

For more information on all other session related settings in php.ini please visit http://www.php.net/manual/en/session.configuration.php.

If you prefer to use the old behavior for the log-in, anonymous user count and session clearing, use the ezpSessionHandlerDB handler. To allow anonymous user count to work as before, you'll additionally need to enable the setting ForceStart in the [Session] block.

This setting is case sensitive. This setting is available for eZ Publish 4.4 and above.

Examples

```
Handler=ezpSessionHandlerDB
```

5

SessionNameHandler

Summary

Sets how session names should be generated

Usage

```
SessionNameHandler=default|custom
```

Description

There are two possible options

1. **default:**The default setting uses PHP to generate session names. If you use the default handler the session will be valid across siteaccess (e.g admin and user site) if they use the same database.

2. **custom:** The custom setting allows you to prefix the session name. The prefix is set with the SessionNamePrefix setting.

5

SessionNamePerSiteAccess

Summary

Prepends session names with the current siteaccess.

Usage

```
SessionNamePerSiteAccess=enabled|disabled
```

Description

When this setting is enabled the generated session names are prepended with the name of the siteaccess used to access the site. This generated unique session names per siteaccess.

Note: Cookies are set per domain, as for security, as well as the cookie for the session id. So using site_ini/session/sessionnamepersiteaccessurl settings with the "*disabled*" value will only work on uri-based site accesses.

5

SessionNamePrefix

Summary

Sets the prefix eZ publish should use when creating session names.

Usage

```
SessionNamePrefix= prefix
```

Description

The prefix set in this variable is prepended to the generated session name. You can override this setting for each siteaccess to generate unique session names per siteaccess.

5

SessionTimeout

Summary

This setting will define the number of seconds a session will last.

Usage

```
SessionTimeout=<integer>
```

Description

This setting will define the number of seconds a session will last. After the specified time has passed the session will no longer be considered valid, and the user will be logged out even if he/ she is active on the site.

259200 seconds, the equivalent of 3 days, is standard.

This setting is a DB handler specific setting and is therefore only used if the eZSessionHandlerDB is set in the setting Handler in the block [Session]. For the PHP handler see note below.

This setting is case sensitive.

Notes on PHP-handler:
1. For the php-based session handler, you'll need to set the php.ini value of:

```
session.gc_maxlifetime=<int>
```

to something higher then default value of 1440 (24 minutes) to avoid session being garbage collected after 24 minutes. For example set it to 259200 for 3 days.

eZ Publish sets this value to value of SessionTimeout during execution, but if there are other applications then eZ Publish 4.4+ running on the same server, or if system is Debian Linux based (including Ubuntu), then you need to set this in php.ini as garbage collection happens outside of eZ Publish execution as well.

2. Session gc scripts / cronjobs in eZ Publish has no effect when you use PHP based handler. You should instead change your php.ini settings to make sure they are cleared regular: http://php.net/manual/en/session.configuration.php

3. On Debian based Linux distroes (including Ubuntu) there is a out of the box cron-job (/etc/cron.d/php5) which uses a script to determine session.gc_maxlifetime (/ usr/lib/php5/maxlifetime).
This script only looks for session.gc_maxlifetime in /etc/php5/*/php.ini, so make sure you have placed your value there.

Examples

5

```
SessionTimeout=259200
```

259200 seconds, the equivalent of 3 days, is standard.

5

[SetupSettings]

\begin{description} \item[CriticalTests]\refer{labelfd05e035d1da220f1773694c99839fc1}\ \\
Sets the tests that must be passed in order to complete the setup procedure.

\item[OptionalTests]\refer{label4f6d3acd45f3e69f6aa271eeed445625}\ \\ Sets the tests optional tests run during the setup procedure.

\item[OverrideSiteDesign]\refer{label53ea97683f70c9fb2e69980d5bf7b4ed}\ \\ Sets the sitedesign that should be used by the setup wizard.

\item[PageLayout]\refer{label7483418fbbb0adcbab1f06496efbc141}\ \\ Sets which pagelayout template to use for the setup wizard.

\end{description}

5

OverrideSiteDesign

Summary

Sets the sitedesign that should be used by the setup wizard.

Usage

```
OverrideSiteDesign= sitedesign_name
```

5

OptionalTests

Summary

Sets the tests optional tests run during the setup procedure.

Usage

```
OptionalTests= test1, test2,...
```

Description

Optional tests can be bypassed simply by clicking next.

5

CriticalTests

Summary

Sets the tests that must be passed in order to complete the setup procedure.

Usage

```
CriticalTests= test1, test2,...
```

5

PageLayout

Summary

Sets which pagelayout template to use for the setup wizard.

Usage

```
PageLayout= template_name
```

Description

Don't change this setting unless you are an eZ publish developer.

5

[ShopSettings]

ClearBasketOnCheckout   (page 2483)
Sets when the basket is cleared.

ClearBasketOnLogout   (page 2481)
Sets whether the basket should be cleared after logging out.

RedirectAfterAddToBasket   (page 2482)
Controls where the user is redirected after adding an item to the basket.

5

ClearBasketOnLogout

Summary

Sets whether the basket should be cleared after logging out.

Usage

```
ClearBasketOnLogout=enabled|disabled
```

Description

This setting has two possible values:

1. *disabled* - Means that the basket is NOT cleared when the user logs out.

2. *enabled* - Means to clear the basket as soon as the user logs out from the system.

Both anonymous and registered users are affected.

Examples

```
ClearBasketOnLogout=enabled
```

This will instruct eZ publish to clear the user's basket as soon as the user logs out from the system.

5

RedirectAfterAddToBasket

Summary

Controls where the user is redirected after adding an item to the basket.

Usage

```
RedirectAfterAddToBasket=basket|reload
```

Description

This setting has two possible values:

1. *basket* - Redirect the user back to the basket to show the newly added item.

2. *reload* - Redirect the user back to the page he came from, this allows the user to continue shopping.

5

ClearBasketOnCheckout

Summary

Sets when the basket is cleared.

Usage

```
ClearBasketOnCheckout=enabled|disabled
```

Description

This setting has two possible values:

1. *disabled* - Means that the basket is cleared when the shop/checkout trigger is done. In practice this means when a user has payed the product and payment system is finished. This is the default value since it means the user can cancel the order and go back to the shop with the basket still intact.

2. *enabled* - Means to clear the basket as soon as the user clicks confirm in the shop/ confirmorder trigger. This may needed by some payment systems. Check the documentation for your payment system to see if this setting must be enabled. The impact on the users is that the basket will not be available when the payment is cancelled.

5

[SiteAccessRules]

Rules   (page)
    Sets which modules and views to enable or disable.

5

Rules

Summary

Sets which modules and views to enable or disable.

Usage

```
Rules[]=function1;parameter1
Rules[]=function2;parameter2
```

Description

The "Rules" setting defines a ruleset that eZ publish uses to determine wether module and views should be available or not. Each line of the ruleset is read and evaluated. The order in which you enable/disable modules does not matter, but more specific rules will override global rules.

The available functions are:

access

The access function is used to change if subsequent *module* or *moduleall* rules should be enabled or disabled. The access function has two possible parameter options

1. **enable** - enable subsequently mentioned modules and views

2. **disable** - disables subsequently mentioned modules and views

module

The module function is used to specify complete modules or specific views.
The module or view will get the access level of the last access level set. To specify a module simply provide the module name as the parameter value:

```
Rules[]=module;name_of_module
```

If you specify a module, the access mode is set for all views in that module.

To specify a single view you need to specify both the name of the module and the view as the parameter value:

```
Rules[]=module;name_of_module/name_of_view
```

5

moduleall

Moduleall is a special mode that should be used on a line on itself without a trailing function definition. It is used to set the current access mode to all modules in the system.

Examples

Example 1

```
Rules[]=access;enable
Rules[]=moduleall
```

This configuration enables all modules and views. This is equal to the default ruleset.

Example 2

```
Rules[]=access;enable
Rules[]=moduleall
Rules[]=access;disable
Rules[]=module;ezinfo
Rules[]=module;content/search
```

This configuration enables all modules and views except all views in the ezinfo module and the view content/search.

Example 3

```
Rules[]=access;disable
Rules[]=moduleall
Rules[]=access;enable
Rules[]=module;content/view
```

This configuration takes a different approach. It disables all modules by default and enables only the modules that you want to use. In this case only the content/view module is available.

5

[SiteAccessSettings]

\begin{description} \item[AnonymousAccessList]\refer{labelf2c78362d35f68c712fbc12d398eefbe}\ \\ Specifies a lists of modules and views that are accessible regardless of the RequireUserLogin setting.

\item[AvailableSiteAccessList]\refer{labelc539bd248e46d2a3fdaee9de37a9edc2}\ \\ Sets the siteaccesses that your eZ Publish installation provides.

\item[CheckValidity]\refer{labele99e4f272210ee7683fba075689ffe52}\ \\ Sets if the setup wizard should be activated or not.

\item[DebugAccess]\refer{labelfce8612c6bb7c4c2ca9514a3ccdb1c9b}\ \\ Turns on debug output for access matching.

\item[DebugExtraAccess]\refer{label3e0f005cb9d9cdcedb5563e271a18398}\ \\ Enables verbose access matching debug information.

\item[ForceVirtualHost]\refer{labelcbbb121eb607eaac95229fb9ea7e6872}\ \\ Sets if eZ Publish should force virtual host mode.

\item[HostMatchElement]\refer{label95587ff076c4d302729960c4c8ffed16}\ \\ Sets which element (separated by a dot) to use when using element host matching.

\item[HostMatchMapItems]\refer{label1dfd3f58a721c5a763ab6584191ca7db}\ \\ Sets the mapping between hostname and siteaccess when using map host-matching.

\item[HostMatchRegexp]\refer{label06fba1b385862e8a33ae1cbad0ad7ab2}\ \\ Sets the regular expression that is used to fetch the siteaccess from the hostname with regexp hostmatching.

\item[HostMatchRegexpItem]\refer{labelaa87d459848490259cf5d7f34f64035c}\ \\ Sets which submatch to use to determine the siteaccess name with regexp hostmatching.

\item[HostMatchSubtextPost]\refer{label75cebfe135256f5c64b79233d1d01546}\ \\ Sets the postfix that occurs in the hostname behind the siteaccess when using text host matching.

\item[HostMatchSubtextPre]\refer{label731fae92421f3d1a9c3c0655749621a2}\ \\ Sets the prefix that occurs in the hostname in front of the siteaccess when using text host matching.

\item[HostMatchType]\refer{label26deea41cfebc463a146d06eb7101ae6}\ \\ Sets which type of host matching to use to select the siteaccess.

\item[MatchOrder]\refer{label9bdac383abf2873c0099cd6e7ee07375}\ \\ Sets the matching algorithms that will be tried to determine which siteaccess to use.

\item[PathPrefix]\refer{labelc806c5437f16a11796eb992fdf472f0a}\ \\ Sets the hidden prefix that will be added to virtual URLs entered by users.

\item[PathPrefixExclude]\refer{label3879cc8eba6606b9b730a1ae2f3f3d46}\ \\ Sets which virtual URLs not to add the hidden prefix to.

\item[RelatedSiteAccessList]\refer{label39cc2430f3c2f37e6c37d31d3e4d9394}\ \\ List of siteaccesses that use the same database.

\item[RequireUserLogin]\refer{label1c57704028be924773b2d22fe0e37c6a}\ \\ Sets if you allow anonymous access to your site.

5

\item[ServerVariableName]\refer{labelf318554e6be0e2a0e3b01041ef9afcc0}\ \\ Not documented yet.

\item[ShowHiddenNodes]\refer{label310d16479a2450f14e91b8db505f8150}\ \\ Sets if hidden nodes should be shown by default or not.

\item[URIMatchElement]\refer{labelc6c36b01a4ec8140cc788dfe27c40196}\ \\ Specifies which element (separated by "/") of the URI that contains the siteaccess.

\item[URIMatchRegexp]\refer{labeld9783a589d05eb89bc9ba0a84ee85796}\ \\ Sets the regular expression that is used to extract the siteaccess from the URI.

\item[URIMatchRegexpItem]\refer{label58782c7db639fecb6b36c1f4059702d7}\ \\ Sets the submatch that is used for siteaccess matching when using regexp URIMatching.

\item[URIMatchType]\refer{labelefd62cd5e4bb87665472e2208260dbe2}\ \\ Configures the URI match access method

\end{description}

5

AnonymousAccessList

Summary

Specifies a lists of modules and views that are accessible regardless of the RequireUserLogin setting.

Usage

```
AnonymousAccessList[]= module1/ view1
AnonymousAccessList[]= module2/ view2
...
```

Description

Using the RequireUserLogin (page 2513) setting you can force users to log in in order to get access to a site. However, those users still need access to the log in page in order to log in. This setting controls the modules and views that are accessible regardless of the RequireUserLogin (page 2513) setting.

The default settings allow access to the modules and views related to logging in and registering a user.

Examples

```
AnonymousAccessList[]=user/register
AnonymousAccessList[]=user/forgotpassword
```

These settings allow access to the register page and the forgotpassword page even if RequireUser-Login is enabled.

5

AvailableSiteAccessList

Summary

Sets the siteaccesses that your eZ Publish installation provides.

Usage

```
AvailableSiteAccessList[]=site_access_name1
AvailableSiteAccessList[]=site_access_name2
...
```

Description

This setting controls the siteaccesses that are available from eZ Publish. Most installations have two siteaccesses: the user site and the administration interface.

This setting has to be set in settings/override/site.ini.append(.php). eZ Publish will use the matching rules to decide which siteaccess a user wants to access. More information about the site access system can be found in the site management (page 138) chapter.

Examples

```
AvailableSiteAccessList[]=example
AvailableSiteAccessList[]=example_admin
```

This system has two siteaccesses: example and example_admin. Specific override settings for the public siteaccess can be found in the directory settings/siteaccess/example/. Specific override settings for the admin siteaccess can be found in the directory settings/siteaccess/example_admin/.

5

CheckValidity

Summary

Sets if the setup wizard should be activated or not.

Usage

```
CheckValidity=true|false
```

5

DebugAccess

Summary

Turns on debug output for access matching.

Usage

```
DebugAccess=enabled|disabled
```

Description

This option is for eZ Publish developers only.

5

DebugExtraAccess

Summary

Enables verbose access matching debug information.

Usage

```
DebugExtraAccess=enabled|disabled
```

Description

This option is for eZ Publish developers only.

5

ForceVirtualHost

Summary

Sets if eZ Publish should force virtual host mode.

Usage

```
ForceVirtualHost=true|false
```

Description

Normally eZ Publish automatically detects if virtual host or non virtualhost should be used. This has been further improved in 4.5, however some special cases require virtual host mode to be forced.

You can read more about the various access methods for eZ Publish in the Access Methods chapter. Specific information about virtual host setups can be found in the Virtual host setup chapter.

5

HostMatchElement

Summary

Sets which element (separated by a dot) to use when using element host matching.

Usage

```
HostMatchElement= number
```

Description

See HostMatchType (page 2501) for examples and more information about host matching.

5

HostMatchMapItems

Summary

Sets the mapping between hostname and siteaccess when using map host-matching.

Usage

```
HostMatchMapItems[]= hostname1; siteaccess1
HostMatchMapItems[]= hostname2; siteaccess2
...
```

Description

Each row in the array defines one hostname to match and the siteaccess that will be used for that hostname.

See HostMatchType (page 2501) for examples and more information about host matching.

5

HostMatchRegexp

Summary

Sets the regular expression that is used to fetch the siteaccess from the hostname with regexp hostmatching.

Usage

```
HostMatchRegexp= regular_expression
```

Description

The regular expression should contain at least one submatch. The setting HostMatchRegexpItem (page 2498) controls which submatch that is used for the siteaccess name.

See HostMatchType (page 2501) for examples and more information about host matching.

5

HostMatchRegexpItem

Summary

Sets which submatch to use to determine the siteaccess name with regexp hostmatching.

Usage

```
HostMatchRegexpItem= number
```

Description

The setting HostMatchRegexp (page 2497) sets the regular expression that is used for matching. See HostMatchType (page 2501) for examples and more information about host matching.

5

HostMatchSubtextPost

Summary

Sets the postfix that occurs in the hostname behind the siteaccess when using text host matching.

Usage

```
HostMatchSubtextPost= number
```

Description

Use this setting together with HostMatchSubtextPre (page 2500) to configure text host-matching. See HostMatchType (page 2501) for examples and more information about host matching.

5

HostMatchSubtextPre

Summary

Sets the prefix that occurs in the hostname in front of the siteaccess when using text host matching.

Usage

```
HostMatchSubtextPre= text
```

Description

Use this setting together with HostMatchSubtextPost to configure text host-matching.
See HostMatchType (page 2501) for examples and more information about host matching.

5

HostMatchType

Summary

Sets which type of host matching to use to select the siteaccess.

Usage

```
HostMatchType=disabled|map|element|text|regexp
```

Description

Use host match when you want to select the siteaccess based on the host part of the URL. Host matching can be used in four different modes or be disabled completely. The available options are:

- **disabled** - Disables host matching. If host matching is specified in the MatchOrder (page 2503) setting it will be ignored.

- **map** - Select the siteaccess based on the complete hostname. Specify the mapping between a hostname and the siteaccess to use with the HostMatchMapItems (page 2496) setting.

- **element** - The siteaccess is specified by a one of the subdomains that are part of the hostname. Select which part of the hostname (separated by dots) that specifies the siteaccess with the HostMatchElement (page 2495) setting.

- **text** - The siteaccess is given by some arbitrary part of the hostname with a fixed pre and post text. Set the pre and post text with the HostMatchSubtextPre (page 2500) and HostMatchSubtextPost (page 2499) settings.

- **regexp** - Use a regular expression to obtain the siteaccess name from the hostname. Set the regular expression and the submatch to use with the HostMatchRegexp (page 2497) and HostMatchRegexpItem (page 2498) settings.

Examples

Map matching

```
HostMatchType=map
HostMatchMapItems[]=mydomain.no;user
HostMatchMapItems[]=admin.mydomain.no;admin
```

This configuration uses the map hostmatching type. If the site is acccessed using the domain mydomain.no the siteaccess named *user* is used. If the domain admin.mydomain.no is used the *admin* siteaccess is used instead.

Element matching

```
HostMatchType=element
HostMatchElement=1
```

This configuration uses the second element in the hostname as siteaccess using element matching. For example the hostname first.second.third.com yields the siteaccess *second*.

Text matching

```
HostMatchType=text
HostMatchSubtextPre=my
HostMatchSubtextPost=site.ez.no
```

This text mathcing configuration uses everything between my and site.ez.no as the siteaccess. E.g myadminsite.ez.no would produce the siteaccess *admin*. The hostname myusersite.ez.no produces the siteaccessname *user*.

Regexp matching

```
HostMatchType=regexp
HostMatchRegexp=^(.+)\.example\.com
HostMatchRegexpItem=1
```

This regexp setup uses the regular expression ^(.+)\.example\.com to match anything in front of .example.com. The first match is set up to hold the siteaccess. E.g the hostname test.example.com yields the siteaccess *test*.

5

MatchOrder

Summary

Sets the matching algorithms that will be tried to determine which siteaccess to use.

Usage

```
MatchOrder= match_type1 [; match_type2]...
```

Description

You can specify several matching types separated by a semicolon. eZ Publish will try each matching type, in the order specified, until a match is found.

The possible algorithms are:

- **host** - Fetch the siteaccess from the hostname. E.g admin.mydomain.com for the admin site and www.mydomain.com for the user site. Use the settings starting with HostMatch to configure host matching.

- **uri** - Fetch the siteaccess from the requested path. E.g mydomain.com/index.php/ *siteaccess_name/*. If you ommit the siteaccess in the URL the default siteaccess will be used. Use the settings starting with UriMatch to configure URI matching.

- **port** - Select the siteaccess based on the port used to access the site. E.g mydomain.com for the user site, and mydomain.com:99 for the administration interface. Use the [PortAccessSettings] group to configure port matching.

- **servervar** - Fetches the siteaccess to use from a server variable found in the PHP $_SERVER global. Use the ServerVariableName setting to choose the server variable that contains the siteaccess to use.

The most common access methods are described in detail in the access methods (page 143) chapter.

5

PathPrefix

Summary

Sets the hidden prefix that will be added to virtual URLs entered by users.

Usage

```
PathPrefix=prefix
```

Description

This directive sets the hidden prefix that will be added to virtual URLs entered by users. It is typically useful for multiple public siteaccesses that are used to display different parts of the content tree.

Consider an example siteaccess called "news" that should display content located under the "info" folder, which can be accessed at "www.example.com/news/info/". Normally, if an article called "conference" is located one level beneath the "info" folder, the URL to access this article will be "www.example.com/news/info/conference". If you specify "PathPrefix=info" in the "settings/ siteaccess/news/site.ini.append.php" configuration file, the "conference" article will also be accessible through a shortened URL "www.example.com/news/conference" (in addition to the full URL mentioned above). This is because the system will automatically add the specified prefix after the siteaccess in the URL and look for the matching entry in the database table that stores the URL aliases. Since the URL alias table contains the "info/conference" entry, which points to the "conference" article node, this article will be displayed. The prefix is considered hidden because the URL in the browser remains as "news/conference".

With this configuration, only content located under the "info" folder will be accessible for the "news" siteaccess. To make other subtrees available for this siteaccess, you need to specify the corresponding prefixes using the "PathPrefixExclude" configuration array. For example, let's say that another folder called "articles" is located at the same level as the "info" folder in the content tree. If somebody enters "www.example.com/news/articles" as a URL, the system will automatically add the "info" prefix and attempt to access, without any success, "www.example.com/ news/info/articles". To make the system properly handle virtual URLs in the "articles" subtree, you need to specify "PathPrefixExclude[]=articles" in the same configuration file. This will make all content located in the "articles" folder available for the "news" siteaccess.

If the "news" siteaccess makes use of multiple site languages, make sure you use the most prioritized one when setting the prefix for virtual URLs (see Example 3).

Note that the explanation above only uses URL aliases with lowercase Latin letters. In reality, folder names may contain different characters (spaces, uppercase letters, Unicode characters), while URL aliases are generated according to the "TransformationGroup (page 2563)" and "WordSeparator (page 2562)" configuration directives. By default, the system uses the "urlalias" transformation method and dashes as word separators. For example, if you create a folder called "Company news", it will get an auto-generated URL alias "Company-news" (stored in

5

the database), while the actual URLs for site visitors can contain "company-news", "COMPANY-NEWS", "company-News" and so on. (Refer to "URL transformation rules" for more information.)

However, the "PathPrefix" directive is case sensitive, and when used, the prefix part becomes case sensitive when the full URL is accessed. In the example above, the prefix for the "Company news" folder must be specified as shown below:

```
PathPrefix=Company-news
```

If you specify "company-news", "COMPANY-NEWS" or "company-News" as a prefix, it will not work correctly.

With this configuration, if an article called "Conference" is located one level beneath the "Company news" folder, it can be accessed at both "www.example.com/news/Conference" (short URL) and "www.example.com/news/Company-news/Conference" (full URL). While the short URL is not case sensitive, the prefix part of the full URL is. This means that both URLs can contain, for example, "conference" instead of "Conference". However, if you replace "Company-news" with "company-news", "COMPANY-NEWS" or "company-News" in the full URL, it will not work any longer.

The "PathPrefixExclude" directive is also case sensitive. To make the content located in the "My articles" folder available for the "news" siteaccess, you need to add the corresponding URL alias to the PathPrefixExclude[] configuration array as shown below:

```
PathPrefixExclude[]=My-articles
```

In addition, it is recommended to have "Users" and "Media" listed in the "PathPrefixExclude[]" configuration array in order to make user accounts and media content available for this siteaccess through virtual URLs (otherwise only system URLs like "www.example.com/content/view/full/45/" will work for these items). In the URLs that point to user accounts and media content, the prefix part will become case sensitive.

If your site makes use of another URL alias configuration, URL aliases are generated differently than in the examples above, so be sure to use URLs appropriate for your setup. For example, if the "urlalias_compat" transformation method is used, only lowercase letters are allowed in URL aliases. (This also means that you'll have to specify "users" and "media" in the "PathPrefixExclude" setting in order to make user accounts and media content accessible.)

Examples

Example 1

Suppose you have two folders called "Books" and "Postcards" located right under the "Content" top level node. If you use two public siteaccesses called "bookshop" and "cards" for displaying content located in these folders, you will most likely have something like this in the "[SiteAccessSettings]" section of the "settings/override/site.ini.append.php" configuration file:

```
...
AvailableSiteAccessList[]
AvailableSiteAccessList[]=bookshop
AvailableSiteAccessList[]=cards
RelatedSiteAccessList[]
RelatedSiteAccessList[]=bookshop
RelatedSiteAccessList[]=cards
...
```

With this configuration, content located in both folders will be available for both siteaccesses. The "Books" folder can be accessed using URLs like "www.example.com/bookshop/Books/" and "www.example.com/cards/Books/". In the same way, the "Postcards" folder will be accessible through both "www.example.com/bookshop/Postcards/" and "www.example.com/cards/Postcards/".

Now, you can configure the "bookshop" siteaccess to only display content located in the "Books" folder. To do that, add the following lines to the "site.ini.append.php" configuration file of the "bookshop" siteaccess:

```
[SiteAccessSettings]
PathPrefix=Books
```

With this configuration, content located in the "Books" folder can be accessed using URLs like "www.example.com/bookshop/MyBook" and "www.example.com/cards/Books/MyBook". URLs like "www.example.com/bookshop/Postcards/MyPostcard" will not work any longer.

To configure the "cards" siteaccess to only display content located in the "Postcards" folder, add the following lines to the "site.ini.append.php" configuration file located in the "settings/siteaccess/cards" directory:

```
[SiteAccessSettings]
PathPrefix=Postcards
```

With this configuration, content located in the "Postcards" folder can be accessed using URLs like "www.example.com/cards/MyPostcard" and "www.example.com/cards/Postcards/MyPostcard". URLs like "www.example.com/bookshop/Postcards/MyPostcard" and "www.example.com/cards/Books/MyBook" will not work any longer.

Example 2

Let's have a look at the previous example assuming that the host access method is used instead of the default one. This means that eZ Publish is set up to use different siteaccesses based on the host/domain combinations of the incoming requests. In this case, the "[SiteAccessSettings]" section in the "settings/override/site.ini.append.php" configuration file will most likely contain the following additional lines:

```
...
MatchOrder=host
HostMatchType=map
HostMatchMapItems[]=www.books.example.com;bookshop
HostMatchMapItems[]=www.greetings.example.com;cards
...
```

With this configuration, the system will use the "bookshop" siteaccess if the requested URL starts with "www.books.example.com". The "cards" siteaccess will be used if the requested URL starts with "www.greetings.example.com". The "Books" folder can be accessed using URLs like "www.books.example.com/Books/" and "www.greetings.example.com/Books/". The "Postcards" folder will be accessible through both "www.books.example.com/Postcards/" and "www.greetings.example.com/Postcards/".

If you specify "PathPrefix=Books" for the "bookshop" siteaccess as shown above, content located in the "Books" folder will be accessible through the following URLs:

- www.books.example.com/MyBook (short URL)

- www.books.example.com/Books/MyBook (full URL)

- www.greetings.example.com/Books/MyBook (full URL for the "cards" siteaccess)

URLs like "www.books.example.com/Postcards/MyPostcard" will not work any longer.

If you also specify "PathPrefix=Postcards" for the "cards" siteaccess, content located in the "Postcards" folder will be accessible through the following URLs:

- www.greetings.example.com/MyPostcard (short URL)

- www.greetings.example.com/Postcards/MyPostcard (full URL)

URLs like "www.greetings.example.com/Books/MyBook" will not work any longer.

Example 3

Let's say that your site content exists in English and French languages and 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`

If a folder is called "Company" in English and "Compagnie" in French, it can be accessed using the following URLs:

5

- www.example.com/gb/Company

- www.example.com/gb/Compagnie

- www.example.com/fr/Company

- www.example.com/fr/Compagnie

While the first two URLs bring up the English version of the folder, the other ones will bring up the French version.

If you want the "gb" siteaccess to only display content located in this subtree, you need to add the following lines to the "settings/siteaccess/gb/site.ini.append.php" configuration file:

```
[SiteAccessSettings]
PathPrefix=Company
```

For the "fr" siteaccess, you would have to specify "Compagnie" as a prefix, since French is the most prioritized language for this siteaccess.

Example 4

Suppose that your site content exists in the Norwegian language. If you create a folder called "Ostehvel" (Cheese slicer) located in the "Produkter" (Products) folder, the system will automatically generate a URL alias for it depending on which transformation method is used:

Transformation method	URL alias
urlalias (default)	Produkter/Ostehoevel
urlalias_iri	Produkter/Ostehvel
urlalias_compat	produkter/ostehoevel

You can configure a siteaccess to only display content located in the "Ostehvel" folder and skip the corresponding prefix in the URLs. If the default transformation method is used, this can be done by adding the following line to the "[SiteAccessSettings]" section of the "site.ini.append.php" file for this siteaccess:

```
PathPrefix=Produkter/Ostehoevel
```

If the "urlalias_iri" transformation method is used, the corresponding line will look like this:

```
PathPrefix=Produkter/Ostehvel
```

If the "urlalias_compat" transformation method is used, you need to specify the prefix as shown below:

```
PathPrefix=produkter/ostehoevel
```

5

PathPrefixExclude

Summary

Sets which virtual URLs not to add the hidden prefix to.

Usage

```
PathPrefixExclude[]=url_alias1
PathPrefixExclude[]=url_alias2
...
```

Description

This directive tells the system which subtrees' virtual URLs should be unaffected by the "Path-Prefix (page 2504)" directive. To add a subtree, specify the auto-generated URL alias of the top node in the most prioritized language.

Examples

Example 1

```
PathPrefixExclude[]
PathPrefixExclude[]=Media
PathPrefixExclude[]=Users
...
```

Assuming that the default configuration of URL aliases is used, this will tell the system not to add any hidden prefix to virtual URLs that point to user accounts and media content (everything located in the "Users" and "Media" folders). For example, a URL like "www.mysite.com/Media/Images/MyPic" will be handled "as is", regardless of what is specified in the "PathPrefix" directive.

5

RelatedSiteAccessList

Summary

List of siteaccesses that use the same database.

Usage

```
RelatedSiteAccessList[]=siteaccess_name1
RelatedSiteAccessList[]=siteaccess_name2
...
```

Description

This configuration array contains a list of siteaccesses that use the same database and belong together. These are called "related siteaccesses". This setting is typically useful when running multiple sites on the same eZ Publish installation.

Note that this setting also controls which siteaccesses the view cache should be cleared for when it is cleared for the current siteaccess. (When clearing the view cache of a node, the system will clear it for all related siteaccesses, unless you use different VarDir (page 2384) settings for them.) If the RelatedSiteAccessList setting is not specified, the cache subsystem will use the "AvailableSiteAccessList (page 2490)" setting instead.

If you have several siteaccesses that are using different databases, this setting must be set per siteaccess.

Examples

Example 1

Let's say that you have several siteaccesses called "gb", "no", "fr" and "myadmin" using the same database. When you change the site pages using the admin interface, the view cache of these pages should be cleared not only for your admin siteaccess but for three public siteaccesses as well. Add the following lines to the "settings/override/site.ini.append.php":

```
RelatedSiteAccessList[]=gb
RelatedSiteAccessList[]=no
RelatedSiteAccessList[]=fr
RelatedSiteAccessList[]=myadmin
```

This will instruct the system to clear the view caches for all your siteaccesses.

If you remove "RelatedSiteAccessList[]=gb" then only three siteaccesses will be considered as related and thus editing a site page in the admin interface will lead to clearing the view cache of this page for "myadmin", "no" and "fr" siteaccess (the view cache for "gb" will not be cleared).

5

Example 2

Let's say that you have the following siteaccesses:

- "gb", "no" and "myadmin" using the "mydb" database

- "shop" and "shopadmin" using the "myshopdb" database

This means you have to specify the following settings in their "site.ini.append.php" configuration files:

settings/siteaccess/gb/site.ini.append.php

```
RelatedSiteAccessList[]=no
RelatedSiteAccessList[]=myadmin
```

settings/siteaccess/no/site.ini.append.php

```
RelatedSiteAccessList[]=gb
RelatedSiteAccessList[]=myadmin
```

settings/siteaccess/myadmin/site.ini.append.php

```
RelatedSiteAccessList[]=gb
RelatedSiteAccessList[]=no
```

settings/siteaccess/shop/site.ini.append.php

```
RelatedSiteAccessList[]=shopadmin
```

settings/siteaccess/shopadmin/site.ini.append.php

```
RelatedSiteAccessList[]=shop
```

5

RequireUserLogin

Summary

Sets if you allow anonymous access to your site.

Usage

```
RequireUserLogin=true|false
```

Description

If this setting is set to true, eZ Publish will redirect all requests from users that are not logged in to the log in page.

This setting is typically overridden for each siteaccess. Private siteaccesses like intranets or the admin interface will typically have RequireUserLogin set to true while public sites that everyone can access typically have RequireUserLogin set to false.

5

ServerVariableName

Summary

Not documented yet.

Usage

```
ServerVariableName
```

5

ShowHiddenNodes

Summary

Sets if hidden nodes should be shown by default or not.

Usage

```
ShowHiddenNodes=true|false
```

Description

eZ Publish has the concept of node visibility (page 127). These settings controls whether hidden nodes should be displayed by default or not. The settings controls the behavior of the content fetch functions. ShowHiddenNodes is typically overridden for siteaccesses where all content should be shown regardless of the state, e.g in the administration interface.

5

URIMatchElement

Summary

Specifies which element (separated by "/") of the URI that contains the siteaccess.

Usage

```
URIMatchElement= number
```

Description

See URIMatchType (page 2519) for examples and more information about URI matching.

5

URIMatchRegexp

Summary

Sets the regular expression that is used to extract the siteaccess from the URI.

Usage

```
URIMatchRegexp= regular_expression
```

Description

Use the setting URIMatchRegexpItem (page 2518) to determine the submatch that contains the siteaccess.

See URIMatchType (page 2519) for examples and more information about URI matching.

5

URIMatchRegexpItem

Summary

Sets the submatch that is used for siteaccess matching when using regexp URIMatching.

Usage

```
URIMatchRegexpItem= number
```

Description

The setting URIMatchRegexp (page 2517) specifies the regular expression that should be used for matching.

See URIMatchType (page 2519) for examples and more information about URI matching.

5

URIMatchType

Summary

Configures the URI match access method

Usage

```
URIMatchType=disabled|map|element|text|regexp
```

Description

URI matching is used to select the siteaccess from the location part of the URL: URL matching can be used in two different modes or be disabled completely. The options you have for this settings are:

- **disabled** - Disables URI matching. If URI matching is present in the MatchOrder (page 2503) setting it will be ignored.

- **map** - Select the siteaccess to use based on the first element of a given URL. Specify the mapping between the URL element and the siteaccess to use with the URIMatchMapItems setting.

- **element** - Select the siteaccess to use based on the elements separated by "/" in the location part. If you specify "URIMatchElement=N", the system will take N parts of the URL (separated by "/") and replace slashes with underscores in the resulting string.

- **text** - Match URL using pre- or post- substrings (URIMatchSubtextPre, URIMatchSubtextPost).

- **regexp** - Works similarly to the element setting but you can specify the delimiter with a regular expression. The settings URIMatchRegExp and URIMatchRegExpItem control how to split the URL and what part to use for the siteaccess.

Examples

Example 1 (element matching)

```
URIMatchType=element
URIMatchElement=1
```

With this configuration the second element (the first element is element 0) of the URI will be used as the siteaccess. For example, in the URL "http://www.mysite.com/news/site/admin/content/view/full/32" the siteaccess name will be "news".

Example 2 (element matching)

```
URIMatchType=element
URIMatchElement=3
```

With this configuration the second element (the first element is element 0) of the URI will be used as the siteaccess. For example, in the URL "http://www.mysite.com/news/site/admin/content/view/full/32" the siteaccess name will be "news_site_admin".

Example 3 (map matching)

```
URIMatchType=map
URIMatchMapItems[]=myadmin_de;admin_de
URIMatchMapItems[]=MYADMIN_DE;admin_de
```

This will tell the system to use the "admin_de" siteaccess when the first element of a given URI is 'myadmin_de' or 'MYADMIN_DE'.

5

[SiteSettings]

64bitCompatibilityMode   (page 2523)
    Enables 64bit compatibility mode

DefaultAccess   (page 2527)
    Sets the default site access when the URI access method is used.

IndexPage   (page 2528)
    Sets the page to display when the root "/" of your site is accessed.

LoginPage   (page 2526)
    Sets if eZ Publish should use a custom pagelayout for the log in page.

MetaDataArray   (page 2529)
    Sets the site metadata that is used on several places on your site.

SiteList   (page 2524)
    Sets the siteaccesses available to outside sources (currently used for webdav only)

SiteName   (page 2531)
    The name of your site.

SiteURL   (page 2530)
    Sets the URL of your site. Used e.g when generating links for notifactions and emails.

SSLPort   (page 2525)
    The port that that should be used for SSL requests.

SSLProxyServerName   (page 2522)
    Sets which virtual host that functions as a proxy server for incoming SSL requests.

5

SSLProxyServerName

Summary

Sets which virtual host that functions as a proxy server for incoming SSL requests.

Usage

```
SSLProxyServerName=virtual_host_name
```

Description

This setting was introduced in eZ Publish 3.10 as a part of a functionality that provides support for using SSL proxy servers.

The purpose of this feature is to reduce complexity by decreasing the number of entries in the Apache configuration file. For each eZ Publish installation, only one virtual host is needed, plus the one that handles all incoming SSL requests. It is typically useful when several SSL-enabled eZ Publish sites/installations are running through different virtual hosts on the same server. Using this feature, it is possible to create an additional virtual host that functions as an SSL proxy server (a single-point-of-contact interface between the clients and the sites/installations). It will listen on port 443 and handle all incoming SSL requests no matter which site they were meant for. The requests are internally redirected to the corresponding virtual hosts running the actual sites. Finally, eZ Publish will handle them and provide the actual response. The responses are passed back to the SSL proxy server which then returns them to the clients that issued the corresponding requests.

When redirecting an SSL request to a virtual host, the proxy server sets the "HTTP_X_FORWARDED_SERVER" header. The value will be the name of the proxy server (the name of the virtual host as configured in Apache). All eZ Publish installations running on different virtual hosts must have the "SSLProxyServerName" directive set to the same value. This makes it possible for eZ Publish to distinguish SSL requests from usual HTTP requests. The system will identify a request as an SSL request if the value of the "HTTP_X_FORWARDED_SERVER" header matches the value of the "SSLProxyServerName" configuration setting. Otherwise, it will be classified as an ordinary HTTP request. The response will be passed to the SSL proxy server and then returned to the client.

The default value of the "SSLProxyServerName" directive is "localhost".

5

64bitCompatibilityMode

Summary

Enables 64bit compatibility mode

Usage

```
64bitCompatibilityMode=enabled|disabled
```

Description

This setting allows to enable or disable 64bit compatibility mode. This will affect some PHP functions like crc32.

Examples

```
64bitCompatibilityMode=enabled
```

This enables 64bit compatibility mode in eZ publish.

5

SiteList

Summary

Sets the siteaccesses available to outside sources (currently used for webdav only)

Usage

```
SiteList[]= siteaccess1
SiteList[]= siteaccess2
...
```

5

SSLPort

Summary

The port that that should be used for SSL requests.

Usage

```
SSLPort= integer
```

Description

Note: The default 443 should suffice for most users.

5

LoginPage

Summary

Sets if eZ Publish should use a custom pagelayout for the log in page.

Usage

```
LoginPage=custom|embedded
```

Description

If LoginPage is set to custom, eZ Publish will use loginpagelayout.tpl when users access the login page.

If LoginPage is set to embedded the normal pagelayout.tpl will be used.

5

DefaultAccess

Summary

Sets the default site access when the URI access method is used.

Usage

```
DefaultAccess= <em>siteaccess_name</em>
```

Description

When you use the URI access method (page 143) it is not possible to see which siteaccess you want to use if you are accessing the root of your site. The DefaultAccess setting specifies which siteaccess to choose when this happens.

5

IndexPage

Summary

Sets the page to display when the root "/" of your site is accessed.

Usage

```
IndexPage= internal url
```

Description

Use either a system or a nice URL to specify the index page. System URL's are a bit faster and should be used for the IndexPage setting since the root of your site will be accessed very often.

Examples

IndexPage=/content/view/full/2

This example shows the full view of node 2 as the index page of your site.

5

MetaDataArray

Summary

Sets the site metadata that is used on several places on your site.

Usage

```
MetaDataArray[ meta_name1]= meta_data1
MetaDataArray[ meta_name2]= meta_data2
...
```

Description

This setting can be used to control meta data that is used in several different locations in your site. The default installation defines author, copyright, description and keywords. You are free to define your own metadata. To fetch the metadata use the ezini template operator.

Examples

MetaDataArray[author]=eZ Systems

MetaDataArray[copyright]=eZ Systems

MetaDataArray[description]=Content Management System

MetaDataArray[keywords]=cms, publish, e-commerce, content management, development framework

5

SiteURL

Summary

Sets the URL of your site. Used e.g when generating links for notifactions and emails.

Usage

```
SiteURL= site_url
```

Description

The URL should contain the full hostname and any additional path to the root of your eZ Publish installation.

Examples

SiteURL=ez.no

A typical generated link will now look like http://ez.no/some_url_here.

5

SiteName

Summary

The name of your site.

Usage

```
SiteName=the name of your site
```

Description

This setting is used in the title of the default templates. You may use spaces in the title.

Examples

SiteName=Sigges megaphone warehouse

5

[SSLZoneSettings]

ModuleViewAccessMode   (page 2533)
     Not documented yet.

SSLSubtrees   (page 2534)
     Not documented yet.

SSLZones   (page 2535)
     Not documented yet.

5

ModuleViewAccessMode

Summary

Not documented yet.

5

SSLSubtrees

Summary

Not documented yet.

5

SSLZones

Summary

Not documented yet.

[TemplateSettings]

You can read more about the template system in the templates (page 169) chapter.

\begin{description} \item[AutoloadPathList]\refer{labelbfe7526a9859d96c29d3ebb1819da9e3}\ \\ Sets the directories inside eZ publish itself where eZ publish will look for operator and function definitions.

\item[Debug]\refer{label69311505f19bee61667d2cfc8ba53fc7}\ \\ Turns on/off template debug output.

\item[DelayedCacheBlockCleanup]\refer{label9441a9ec76d3b80cb76ab22cc427aa93}\ \\ Enables delayed removal of expired cache blocks with the "subtree_expiry" parameter.

\item[DevelopmentMode]\refer{label2367887d601be8c20e2ae8c2bbc06050}\ \\ Not documented yet.

\item[ExtensionAutoloadPath]\refer{labelele8a419dac03450557ef91af5e3711f5c}\ \\ Sets the extensions that contain template function or operator definitions.

\item[NodeTreeCaching]\refer{label16e5704ff0fa186c511c11d0893244a2}\ \\ Sets if the template interpreter should cache parsed template files.

\item[ShowMethodDebug]\refer{label370be2bc4c4b81825bbe45e17b9f3fa5}\ \\ Sets if debug information about called functions and operators should be displayed.

\item[ShowUsedTemplates]\refer{labela6d730df20218be8fabdcb44ddbc151a}\ \\ Enables a table in the debug displaying all the templates used to render the current page.

\item[ShowXHTMLCode]\refer{label9d2f933caba203699a8225a0f4079bbb}\ \\ Sets if eZ publish should display template load debug inline or not.

\item[TemplateCache]\refer{label8abc93b7f7751314717e2c76df5ca929}\ \\ Main switch for all the template related caches.

\item[TemplateCompile]\refer{labelb5bddac5e7c269b972029edac900c6fd}\ \\ Sets if the template compiler should be used or not.

\item[TemplateCompression]\refer{label61cd3f8d5cb7fa0b2c30f5171cf6aac7}\ \\ Sets if compiled templates should be compressed or not.

\item[TemplateOptimization]\refer{label2362bd747a3aa36ae26a388354850dc1}\ \\ Sets if the template compiler should try to optimize the produced PHP code.

\item[UseFormatting]\refer{label8a27dc5094943cdd623edc416d689b50}\ \\ Sets if the template compiler should keep whitespace in the compiled template.

\end{description}

5

DelayedCacheBlockCleanup

Summary

Enables delayed removal of expired cache blocks with the "subtree_expiry" parameter.

Usage

```
DelayedCacheBlockCleanup=enabled|disabled
```

Description

This setting is used to delay the clearing of expired template block caches.

If you are using the "cache-block (page 1383)" 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). By default, the "DelayedCacheBlockCleanup" setting is disabled, and the system will automatically remove cache blocks with the "subtree_expiry" parameter when they expire. If this setting is enabled, the expired cache blocks must be removed manually or using the "subtreeexpirycleanup.php" cronjob script.

5

DevelopmentMode

Summary

Not documented yet.

5

TemplateCompression

Summary

Sets if compiled templates should be compressed or not.

Usage

```
TemplateCompression=enabled|disabled
```

Description

Enable template compression to make eZ publish store compiled templates using gzip. This saves a lot of space but is a bit slower.

5

TemplateCache

Summary

Main switch for all the template related caches.

Usage

```
TemplateCache=enabled|disabled
```

Description

If you disable TemplateCache the template system will not do any caching at all. This includes cache-blocks and compiled templates.

Note: Template caching is essential to speed up your site. Do not turn TemplateCache off unless you are developing.

5

UseFormatting

Summary

Sets if the template compiler should keep whitespace in the compiled template.

Usage

```
UseFormatting=enabled|disabled
```

Description

Enabling this setting makes the template compiler keep whitespace formatting in the compiled templates. This results in HTML output that bigger but easier to read.

This setting has no effect if TemplateCompile (page 2543) is disabled.

5

TemplateOptimization

Summary

Sets if the template compiler should try to optimize the produced PHP code.

Usage

```
TemplateOptimization=enabled|disabled
```

Description

Optimized templates will be slightly faster than templates that are not compiled.

This setting has no effect if TemplateCompile (page 2543) is disabled.

5

TemplateCompile

Summary

Sets if the template compiler should be used or not.

Usage

```
TemplateCompile=enabled|disabled
```

Description

If you enable TemplateCompile eZ Publish will convert your templates into executable PHP files. If this option is disabled eZ Publish will interpret each template separately for each page.

eZ Publish will compile the templates on demand. Once a template is compiled it takes less time to process. However, compiling templates takes a lot of time. This can be noticed after you have cleared the cache. If this is a problem for your site you can use the "bin/php/eztc.php" script to compile all your templates before you go live.

5

NodeTreeCaching

Summary

Sets if the template interpreter should cache parsed template files.

Usage

```
NodeTreeCaching=enabled|disabled
```

Description

Enable this setting to make eZ Publish go faster if you have disabled TemplateCompile (page 2543). If TemplateCompile (page 2543) is enabled this setting will have no effect.

5

ShowUsedTemplates

Summary

Enables a table in the debug displaying all the templates used to render the current page.

Usage

```
ShowUsedTemplates=enabled|disabled
```

Description

Debug (page 2548) must be enabled for this setting to have any effect.

As of eZ Publish 4.5, when both "Debug" and "ShowUsedTemplates" are enabled, you will get output information on packed javascript and stylesheet files in your debug list.

See example screenshot below:

(see figure 5.99)

Figure 5.99:

Note: If you have caching enabled you will have to clear the cache for the table to display all templates used to render a page.

5

ShowMethodDebug

Summary

Sets if debug information about called functions and operators should be displayed.

Usage

```
ShowMethodDebug=enabled|disabled
```

Description

Debug (page 2548) must be enabled for this setting to have any effect.

The extra debug output provided by this setting is only useful for kernel developers.

5

ShowXHTMLCode

Summary

Sets if eZ publish should display template load debug inline or not.

Usage

```
ShowXHTMLCode=enabled|disabled
```

Description

If ShowXHTMLCode is enabled eZ publish will display a comment in the rendered output of the browser for each time a new template is loaded.
Debug (page 2548) must be enabled for this setting to have any effect.

Note: If you have caching enabled you may will have to clear the cache for this setting to take effect.

5

Debug

Summary

Turns on/off template debug output.

Usage

```
Debug=enabled|disabled
```

Description

If you enable this setting eZ Publish will display a list of debug messages at the bottom of each page. The output is configurable, but will typically consist of information about the time used to render the page, warnings or errors while executing your template, information about the SQL queries executed and a list of the templates used.
Turning on debug will also make eZ Publish output comments in the HTML output each time it loads a new template.

If things don't work as you expected while developing your site, turn on Debug and make sure that there are no warnings or errors while rendering your page.

Note: If you have caching enabled, some debug messages may disappear the second time you load a page. This is because the cache was used instead of rerendering the page. If you want to see all the debug messages you have to clear the cache or turn off caching alltogether.

When it comes to the debug setting, use either:

- "enabled" to see which template files are loaded or

- "disabled" to supress debug

Warning: Will add debug xhtml comments you your source code, including mails!

Note: No debug on templates starting with <!DOCTYPE to not trigger quirks mode in web browsers!

5

ExtensionAutoloadPath

Summary

Sets the extensions that contain template function or operator definitions.

Usage

```
ExtensionAutoloadPath[]= directory_name1
ExtensionAutoloadPath[]= director_namey2
...
```

Description

eZ Publish will look for the file eztemplateautoload.php inside the directory "/extension/ *directory_name/*autoloads". The eztemplateautoload.php file should contain the function and operator definitions provided by the corresponding extension.

This setting is typically set in the settings for each extension that contain template operators of functions.

Examples

```
ExtensionAutoloadPath[]=myextension
```

This setting will make eZ publish load template function and operator definitions from the file "/extension/myextension/autoloads/eztemplateautoload.php"

5

AutoloadPathList

Summary

Sets the directories inside eZ publish itself where eZ publish will look for operator and function definitions.

Usage

```
AutoloadPathList[]= directory1
AutoloadPathList[]= directory2
...
```

Description

If you want to specify additional paths that eZ publish should search for template operators and functions use the ExtensionAutoLoadPath setting.

Note: Do not change this setting unless you know what you are doing.

5

[TimeZoneSettings]

TimeZone   (page 2552)
    Sets the timezone to use in eZ Publish.

5

TimeZone

Summary

Sets the timezone to use in eZ Publish.

Usage

```
TimeZone=Area/Location
```

Description

In order to run multiple installations of eZ Publish on one server using different timezones it is now possible to override the OS default timezone. The time zones are named as "Area/Location", where "Area" is the name of a continent or ocean, and "Location" is the name of a specific location within that region, usually cities or small islands. (The list of timezones can be found at http://www.php.net/manual/en/timezones.php)

This setting is empty by default, which means that eZ Publish will use the PHP default timezone (controlled by the "date.timezone" value in the "php.ini" configuration file).

Examples

Example 1

```
TimeZone=America/Antigua
```

This will instruct the system to use the "America/Antigua" timezone instead of the PHP default timezone / OS default timezone.

5

[TipAFriend]

FromEmail   (page 2554)
    Sets the from address used when sending tip a friend e-mail.

MaxRequestsPerTimeframe   (page 2555)
    Set the maximum number of tip a friend mail the system will send to one email address.

TimeFrame   (page 2556)
    Sets the time frame for the tip a friend functionality.

5

FromEmail

Summary

Sets the from address used when sending tip a friend e-mail.

Usage

```
FromEmail= email_address
```

Description

The specified e-mail address is used as the "From:" address when sending "tip-a-friend" e-mails. This setting is empty by default, so the email address specified in the "tip-a-friend" form will be used as the "From:" address when sending the e-mail.

Examples

```
FromEmail=no@spam.org
```

This makes eZ Publish use "no@spam.org" as the from address when sending "tip-a-friend" e-mails.

5

MaxRequestsPerTimeframe

Summary

Set the maximum number of tip a friend mail the system will send to one email address.

Usage

```
MaxRequestsPerTimeframe= number
```

Description

eZ Publish will not allow anyone to send more than MaxRequestsPerTimeframe tip a friend mail per TimeFrame (page 2556) to any single mail address. This prevents the possibility to spam a certain email address.

Examples

```
MaxRequestsPerTimeframe=10
TimeFrame=2
```

Using these settings eZ Publish will not allow users to send more than 10 tip a friend mails to a single email address per two hours.

5

TimeFrame

Summary

Sets the time frame for the tip a friend functionality.

Usage

```
TimeFrame= number_of_hours
```

Description

You can control the maximum number of tip a friend mail the system can send to one address using this setting and the MaxRequestsPerTimeframe (page 2555) setting.

5

[UnitSettings]

BinaryUnits   (page 2559)

     Sets which units eZ publish considers binary units.

UseSIUnits   (page 2558)

     Sets if eZ publish should use OSI or SI prefixes for binary numbers

5

UseSIUnits

Summary

Sets if eZ publish should use OSI or SI prefixes for binary numbers

Usage

```
UseSIUnits=true|false
```

Description

If this setting is set to true eZ publish will use SI prefixes for binary numbers. If the setting is set to false eZ publish uses OSI prefixes for binary numbers. You can set the units that eZ publish should consider binary with the BinaryUnits (page 2559) setting.

5

BinaryUnits

Summary

Sets which units eZ publish considers binary units.

Usage

```
BinaryUnits= unit1; unit2;...
```

Description

Binary units will get binary prefixes in front of the 'B' for byte or 'b' for bit. The possible units are listed in units.ini (page 2694).

Use the UseSIUnits (page 2558) settings to force eZ publish to use SI prefixes instead of the standard OSI approved prefixes.

The configuration file units.ini (page 2694) allows you to alter the prefixes.

Examples

OSI prefixes

The binary number 1 048 576 byte will be displayed as 1 MB
The binary number 1 048 576 bit will be displayed as 1 Mb

SI prefixes

The binary number 1 048 576 byte will be displayed as 1 MiB
The binary number 1 048 576 bit will be displayed as 1 Mib

5

[URLTranslator]

Filters   (page 2561)

      Sets the active filters that will be used to filter the text in the URLs.

MaximumWildcardIterations   (page 2564)

      Sets how many times the wildcard matches can iterate

TransformationGroup   (page 2563)

      Sets the type of transformation for the URLs.

Translation   (page 2566)

      Enables/disables the URL translation functionality

WildcardTranslation   (page 2565)

      Enables subtree URL translation

WordSeparator   (page 2562)

      Sets which word separator that should be used in the URLs.

5

Filters

Summary

Sets the active filters that will be used to filter the text in the URLs.

Usage

```
Filters[]=filter1
Filters[]=filter2
...
```

Description

This configuration array tells the system which filters that should be used for URL aliases text filtering. Each line contains the name of the class which implements the filtering. The name of the class in lowercase specifies the name of the PHP file which the system will search for to perform filtering. Note that all filters must be placed under the "urlfilters" subdirectory located inside the directory of an extension.

Examples

Example 1

```
Extensions[]
Extensions[]=myfilters

Filters[]
Filters[]=MyFilter
```

This configuration tells the system to use the "MyFilter" filter from the "myfilter.php" file located in the "extension/myfilters/urlfilters" directory. The filter will be used every time a URL alias is created (either by the system or by users).

5

WordSeparator

Summary

Sets which word separator that should be used in the URLs.

Usage

```
WordSeparator=dash|underscore|space
```

Description

This directive controls which word separator that should be used in the URLs. The default value is "dash", which means that system will use dashes as word separators in virtual URLs (e.g. "www.example.com/about-us"). It is also possible to specify "space" or "underscore" so that the system will use spaces or underscores as word separators. However, if the "TransformationGroup (page 2563)" directive is set to "urlalias_compat" (which is the old, pre-eZ Publish 3.10 URL transformation method), the system will ignore the "WordSeparator" setting and use underscores as separators.

Make sure you specify the same "WordSeparator" value for all siteaccesses that share the same database.

Every time this setting is changed, the "bin/php/updateniceurls.php" script must be run in order to update existing URL aliases. Refer to "TransformationGroup (page 2563)" for information about how to run the script.

5

TransformationGroup

Summary

Sets the type of transformation for the URLs.

Usage

```
TransformationGroup=urlalias_iri|urlalias|urlalias_compat
```

Description

This directive sets which method of transformation that should be used when generating virtual URLs. The default value is "urlalias". Refer to the "Character transformation" part of the "Multi-language support for URL aliases" documentation page for information about the available transformation methods. Make sure you specify the same value for all siteaccesses that share the same database.

Every time this setting is changed, the "bin/php/updateniceurls.php" script must be run in order to update the URL aliases stored in the database. This script should be run once for each database, specifying one siteaccess per database:

```
php bin/php/updateniceurls.php --fetch-limit=number -s example
```

Replace "example" with the actual name of the siteaccess and specify the desired number of items to handle per one iteration instead of "number"; the script will automatically do as many iterations as necessary to update all the URLs, based on the number specified. Note that the "fetch-limit" parameter is optional.

The script will create new virtual URLs for all content object nodes in accordance with the selected method of transformation. The time it takes to update all URL aliases is proportional to the number of nodes on your site.

Note that after running the "updateniceurls.php" script, you need to review your configuration settings and update the URL aliases that are used in configuration files. For example, if your site makes use of the "PathPrefix (page 2504)" and "PathPrefixExclude (page 2510)" directives, this functionality might stop working if the specified URL aliases have changed. Also, you need to review the configuration settings in your "override.ini.append.php" file(s), otherwise your override templates activated by the "url_alias" condition will most likely stop working.

5

MaximumWildcardIterations

Summary

Sets how many times the wildcard matches can iterate

Usage

```
MaximumWildcardIterations= number_of_iterations
```

Description

The wildcard matcher system will iterate if the translated url is not a complete url, this allows urls lookup to be recursive and is required for proper subtree history.

Note: This is an implementation specific setting which should not be changed unless you know what you are doing.

5

WildcardTranslation

Summary

Enables subtree URL translation

Usage

```
WildcardTranslation=enabled|disabled
```

Description

Wildcard translations can be used to translate a complete node tree into another one. This is used for example if you move a complete tree of nodes. It is also possible to specify wildcard translations from the admin interface.

Note: Do no disable this option unless you are having problems with wildcardtranslations.

5

Translation

Summary

Enables/disables the URL translation functionality

Usage

```
Translation=enabled|disabled
```

Description

When URL translation is enabled, eZ Publish automatically generates easy URL's for all content objects based on the object names. These URL's can be used instead of using system URL's. Additionaly the custom URL translation system (available from the admin interface) is enabled.

More information about URL translation can be found in the URL translation (page) section

5

[UserSettings]

\begin{description} \item[AnonymousUserID]\refer{label3668398ffcabf91b7f1b86063f0505c8}\ \\ Sets the eZ Publish user that should be used for anonymous page requests.

\item[AuthenticateMatch]\refer{labelefcfcf42eb5fc8ff7a4203a7745de548}\ \\ Sets the fields that are accepted for login authentication.

\item[DefaultSectionID]\refer{labelad5f7fd809469da4629ea4d78ae7ffef}\ \\ Sets the section ID of self registered users.

\item[DefaultUserNameValidationErrorText]\refer{labelff63f07f007d20602296d1313bbecb0e}\ \\ Settings regarding custom validation of user names

\item[DefaultUserPlacement]\refer{labeld7fa8896b801f8554d7eea8a2b39bd83}\ \\ Sets the parent object of all users who register themselves

\item[ExtensionDirectory]\refer{label2e520a0110cb73923cc9612380f12445}\ \\ Sets the extensions that contain login handlers

\item[GeneratePasswordIfEmpty]\refer{label90264c16ad1c88f3e27364301f2a3f55}\ \\ Sets if eZ Publish should generate passwords if the password field is empty.

\item[GeneratePasswordLength]\refer{label25fd15d18785af460104957580518e60}\ \\ Sets the length of passwords generated by eZ Publish.

\item[HashType]\refer{label0152b2503a443a9b95ebb75a9ce68f89}\ \\ Sets the type of encryption that will be used when storing user passwords in the database.

\item[LoginHandler]\refer{label132776f34d4a85e170b3d9fb52b7d8f2}\ \\ Sets the various methods eZ Publish will try to authenticate user logins.

\item[LoginRedirectionUriAttribute]\refer{labelfd0f6b7a8761ccd598b2737acb922b2a}\ \\ Sets which attribute of the user/usergroup class contains redirection URI for the user.

\item[LogoutRedirect]\refer{labelfa9b99a14204a21c7c264a2a99b91121}\ \\ Sets which page to redirect to when a user has logged out.

\item[MaxNumberOfFailedLogin]\refer{label4b682182a32cc7d5c437dc9d70bfa0ff}\ \\ Sets the number of failed login attempts before user accounts are locked.

\item[RedirectOnLogoutWithLastAccessURI]\refer{labeld4153a616e0ed3240cd89184aa360b55}\ \\ Sets if LastAccessURI session variable is used to redirect the user at logout.

\item[RegistrationEmail]\refer{labelcb4c257bc9833b2abaedd3af1b429dd2}\ \\ Sets the receiver of notification emails about new users.

\item[RegistrationFeedback]\refer{labeldc1882d46ea479c040f273bf1ab7c02c}\ \\ Sets the kind of feedback that is sent to users that have registered.

\item[RequireUniqueEmail]\refer{label82cb1ae22c95569e1b32d10c94fbe774}\ \\ Sets if all users must have unique email addresses when registering.

\item[ShowMessageIfExceeded]\refer{labelcdb2667e89ef79fafd86cbe03814a3be}\ \\ Sets whether to display a warning when a user is not allowed to log in.

5

\item[SiteName]\refer{label557213931daa98eee34ab1cf536d3337}\ \\ Sets the site name to use when encrypting user passwords.

\item[TrustedIPList]\refer{label2989b5b33153fb64303617fcb00887f4}\ \\ Sets IP addresses that users can log in from even though their accounts are locked.

\item[UpdateHash]\refer{label5922c39eb189909a7da562c2b95e2740}\ \\ Sets if eZ Publish should update hashes if you have changed the HashType setting

\item[UserClassGroupID]\refer{labelb892c856310a03ae0ab618d465b3e853}\ \\ Sets which classes you can create in the user section in the administration interface.

\item[UserClassID]\refer{labeld48625f9a637f1fd387784db230d74d4}\ \\ Sets the class to use for user registration

\item[UserCreatorID]\refer{label4617037035d1ad463fde9d0e5f04bd50}\ \\ Sets the user that will bet set as the creator of self registering users.

\item[UserGroupClassID]\refer{label318d3b92d99e872227bf5e4f1834e3eb}\ \\ Sets the class ID of the class that represents user groups.

\item[UserNameValidationErrorText]\refer{label621a8ff58372aa6a0f52996aba56e665}\   \\ Settings regarding custom validation of user names

\item[UserNameValidationRegex]\refer{label985e859a55d88701072aef9d85d46397}\ \\ Settings regarding custom validation of user names

\item[UseSpecialCharacters]\refer{labelf4eb48be18aca6c21413b1b041d8cc22}\ \\ Sets if special characters are allowed in passwords.

\item[VerifyUserEmail]\refer{label45b5fcc26abbdf97956f14abf181d43f}\ \\ Sets if new users have to verify their account by email.

\end{description}

5

RedirectOnLogoutWithLastAccessURI

Summary

Sets if LastAccessURI session variable is used to redirect the user at logout.

Usage

```
RedirectOnLogoutWithLastAccessURI=
```

Description

This setting is new since eZ Publish 4.1. This setting controls if the LastAccessURI session variable must be used to decide where to redirect the user at log-out.
Possible values are "enabled" and "disabled"

Examples

```
RedirectOnLogoutWithLastAccessURI=disabled
```

5

DefaultUserNameValidationErrorText

Summary

Settings regarding custom validation of user names

Usage

```
DefaultUserNameValidationErrorText=
```

Description

This setting is added since eZ Publish 4.1. It is used for custom validation of user names which will affect new users, and existing users when they edit their credentials.

The DefaultUserNameValidationErrorText setting provides a default error text when user name validation fails.

Examples

```
UserNameValidationRegex[]=
 UserNameValidationErrorText[]=
 DefaultUserNameValidationErrorText=User login name did not validate!
```

5

UserNameValidationErrorText

Summary

Settings regarding custom validation of user names

Usage

```
UserNameValidationErrorText[]=
```

Description

This setting is added since eZ Publish 4.1. It is used for custom validation of user names which will affect all new users, and existing users when they edit their credentials.

Examples

```
UserNameValidationRegex[ew]=/\s$/
UserNameValidationErrorText[ew]=The username cannot end with a whitespace.
```

5

UserNameValidationRegex

Summary

Settings regarding custom validation of user names

Usage

```
UserNameValidationRegex[]=
```

Description

This setting is available since eZ Publish 4.1. It is used for custom validation of user names which will affect new users, and existing users when they edit their credentials.

Examples

Example 1

```
UserNameValidationRegex[sw]=/^\s/
UserNameValidationErrorText[sw]=The user name cannot start with a white-space.
```

More examples

```
#UserNameValidationRegex[ew]=/\s$/
#UserNameValidationErrorText[ew]=The user name cannot end with a white-space.
#UserNameValidationRegex[rw]=/\s+/
#UserNameValidationErrorText[rw]=The user name cannot contain repeated
white-space.
#UserNameValidationRegex[lb]=/\n|\r/
#UserNameValidationErrorText[lb]=The user name cannot contain line breaks.
#UserNameValidationRegex[t]=/\t/
#UserNameValidationErrorText[t]=The user name cannot contain tabs.
```

Alternative example only allowing 'word' characters

```
#UserNameValidationRegex[w]=/\W/
#UserNameValidationErrorText[w]=The user name can only contain a-z, digits or
underline.
# Or
#UserNameValidationRegex[wh]=/[^A-Za-z0-9_-]/
#UserNameValidationErrorText[wh]=The user name can only contain a-z, digits,
underline or hyphen.
```

5

```
# While testing add this at the end, just remember to remove it when your done.
#UserNameValidationRegex[test]=/\w/
#UserNameValidationErrorText[test]=The username seems to be ok!
```

5

SiteName

Summary

Sets the site name to use when encrypting user passwords.

Usage

```
SiteName=name_of_site
```

Description

The provided site name will be used by the system while encrypting user account passwords when the "HashType (page 2587)" directive is set to "md5_site". The default value is "ez.no".

5

ShowMessageIfExceeded

Summary

Sets whether to display a warning when a user is not allowed to log in.

Usage

```
ShowMessageIfExceeded= true|false
```

Description

If a user's account is automatically locked because of too many invalid login attempts (page 2577), the user will not be able to log in. If the "ShowMessageIfExceeded" setting is set to "true", the message "User is not allowed to login because count of failed login attempts for this user exceeded max number of failed login attempts!" will appear when a siteaccess using the "admin" design is accessed. The default value of this setting is "false", which means that no warning is shown for locked user accounts.

This message is not shown on the user siteaccess by default. If you wish to display it on the user siteaccess, you will have to check the "user_is_not_allowed_to_login" field of the "$User" variable within the "templates/user/login.tpl" template of the design being used, or an override. Refer to the "design/admin/templates/user/login.tpl" template to see how this can be done.

5

TrustedIPList

Summary

Sets IP addresses that users can log in from even though their accounts are locked.

Usage

```
TrustedIPList[]
TrustedIPList[]= ip1|network1
TrustedIPList[]= ip2|network2
```

Description

This configuration array can contain a set of IP addresses and/or ranges. The entered values define trusted hosts and/or zones. Login requests coming from trusted locations will be processed regardless if the user accounts are locked. For example, if the site administrator's account is locked, the administrator will still be able to login from a host/zone which is defined in the trusted IP list.

Examples

Example 1

```
TrustedIPList[]
TrustedIPList[]=192.168.0.1
TrustedIPList[]=192.0.0.0/27
```

If a user's account is locked, the user will still be able to login if the request is coming from 192.168.0.1 or 192.0.0.1 - 192.0.0.30.

5

MaxNumberOfFailedLogin

Summary

Sets the number of failed login attempts before user accounts are locked.

Usage

```
MaxNumberOfFailedLogin= number
```

Description

Sets the number of failed login attempts before user accounts are automatically locked by the system. Note that a locked user account differs from a disabled user account. A locked account can be unlocked by a user with administrator privileges. The default value of this setting is 0, which means that the number of invalid login attempts is unlimited (no user accounts will be locked).

Examples

Example 1

```
MaxNumberOfFailedLogin=5
```

This setting will limit the maximum number of failed login attempts for each user to 5. If a user attempts to log in 6 times with an invalid password, the account will be locked.

5

LoginRedirectionUriAttribute

Summary

Sets which attribute of the user/usergroup class contains redirection URI for the user.

Usage

```
LoginRedirectionUriAttribute[user]= attribute_identifier1
LoginRedirectionUriAttribute[group]= attribute_identifier2
```

Description

The redirection URI can be specified for a user/usergroup using an attribute of the "Text line" datatype. The "LoginRedirectionUriAttribute" setting specifies which attribute of the user/ usergroup class is used for storing the redirection URI. Please refer to the "Advanced redirection after login (page 370)" section of the "Features" chapter for more information.

Note: Specify '/' as the redirect uri in the login form to make it try to read the user / user group object attribute value.

Examples

Example 1

```
[UserSettings]
LoginRedirectionUriAttribute[user]=redirection_uri
```

This will tell the system that a user should always be redirected after login to the URL that is specified in the "redirection_uri" attribute for this user.

Example 2

```
[UserSettings]
LoginRedirectionUriAttribute[group]=start_page
```

This will tell the system that a user should always be redirected after login to the URL that is specified in the "start_page" attribute for the user group that this user belongs to.

Example 3

5

```
[UserSettings]
LoginRedirectionUriAttribute[user]=redirection_uri
LoginRedirectionUriAttribute[group]=start_page
```

This will tell the system that a user should always be redirected after login to the URL that is specified in the "redirection_uri" attribute for this user. If no URL is specified in this attribute then the user will be redirected to the URL that is specified in the "start_page" attribute for the user group that this user belongs to.

5

ExtensionDirectory

Summary

Sets the extensions that contain login handlers

Usage

```
ExtensionDirectory[]= extension_name1
ExtensionDirectory[]= extension_name2
...
```

Description

It is possible to provide custom login handlers using the extension system. This setting specifies the extensions that contain a login handler. eZ Publish will look for the handlers in the path: extension/ *extension_name*/login_handler/

5

LoginHandler

Summary

Sets the various methods eZ Publish will try to authenticate user logins.

Usage

```
LoginHandler[]= handler1
LoginHandler[]= handler2
...
```

Description

eZ Publish will try to authenticate users using the login handlers in the order they are specified. eZ Publish provides the following login handlers:

- **standard** - The default login handler for eZ Publish. Users are authenticated using the user objects found in eZ Publish itself.

- **LDAP** - The LDAP login handler. Users are authenticated through an LDAP server. Settings related to the LDAP login handler can be found in ldap.ini (page 2032).

- **textfile** - The textfile login handler allows users to be specified in a textfile similar to the passwd file on *NIX systems. Settings related to the textfile login handler can be found in textfile.ini (page 2632).

Common for the login handlers different from the standard login handler is that eZ Publish automatically creates these users in eZ Publish itself. These users can then log in using the standard log in handler. Users created this way are updated automatically using cronjobs.

Examples

LoginHandler[]=standard
LoginHandler[]=LDAP

This configuration will try to log in users using the standard handler. If the standard handler fails, eZ Publish will try to authenticate the user via LDAP. If that also fails, the login is unsuccessful.

5

LogoutRedirect

Summary

Sets which page to redirect to when a user has logged out.

Usage

```
LogoutRedirect= URL
```

Examples

```
LogoutRedirect=/user/login
```

The default setting redirects the user to the log in page when logging out.

5

UseSpecialCharacters

Summary

Sets if special characters are allowed in passwords.

Usage

```
UseSpecialCharacters=true|false
```

Description

If UseSpecialCharacters is set to false, eZ Publish will only accept passwords with characters in the range a-z, A-Z and 0-9.

5

RequireUniqueEmail

Summary

Sets if all users must have unique email addresses when registering.

Usage

```
RequireUniqueEmail=true|false
```

Description

We highly recomend setting RequireUniqueEmail to true.

5

AuthenticateMatch

Summary

Sets the fields that are accepted for login authentication.

Usage

```
AuthenticateMatch= type1 [; type2]
```

Description

Types accepted are

- **login** - The username field can be used identify a user.

- **email** - The email field can be used to identify a user. If your site allows several users with the same email address, all the users with this email address will be tested for the correct password.

In order to log in successfully the password field must always be provided.

Examples

AuthenticateMatch=login;email
The default setting allows users to log in using both the email address or the username.

AuthenticateMatch=login
This setting allows users to log in using only the username.

5

UpdateHash

Summary

Sets if eZ Publish should update hashes if you have changed the HashType setting

Usage

```
UpdateHash=true|false
```

Description

If this setting is enabled and eZ Publish discovers a password stored with one of the other hash types the hash is automatically converted. Note that you can't change between any of the md5 settings and the plaintext setting.

5

HashType

Summary

Sets the type of encryption that will be used when storing user passwords in the database.

Usage

```
HashType=md5_user|md5_password|md5_site|crypt|plaintext
```

Description

This setting controls how the system will encrypt user passwords that are stored in the database. The following table reveals the possible values:

Type	Description
md5_user	Generates a hash using the MD5 algorithm based on the username and password.
md5_password	Generates a hash using the MD5 algorithm based solely on the password.
md5_site	Generates a hash using the MD5 algorithm based on the site name (page 2574), username and password. Note that hashes which are generated using this option will not work after the value of the "SiteName" setting is changed (the users will not be able to log in).
crypt	Generates a hash using the crypt() PHP function which again uses the standard Unix DES-based encryption algorithm.
plaintext	The passwords are not encrypted. Instead, they are stored as plain text.

The default value is "md5_user". Note that "plaintext" is not recommended because of security reasons.

5

UserCreatorID

Summary

Sets the user that will bet set as the creator of self registering users.

Usage

```
UserCreatorID= userid
```

Description

Users that register themselves will have a user object generated by eZ Publish. This setting controls which user on the system that will be considered the creator of the user objects.
Note that userid is equal to the ID number of the actual object that represents the user account.

5

UserClassGroupID

Summary

Sets which classes you can create in the user section in the administration interface.

Usage

```
UserClassGroupID= class_group_id
```

Description

This setting controls the classes that are listed in the "Create here" dropdown menu in the users section. You must specify a class group that contains all the user types and group types that you want to create.

Note that there can be only one usergroup class and that all classes representing users must have the ezuser datatype.

5

UserGroupClassID

Summary

Sets the class ID of the class that represents user groups.

Usage

```
UserClassGroupID= classid
```

Description

There can only be one class representing user groups on your system.
Note that classid is the ID number of the class (not the identifier of the class).

5

UserClassID

Summary

Sets the class to use for user registration

Usage

```
UserClassID= classid
```

Description

The UserClassID must specify a content class containing the ezuser datatype. This class will be used when new users are registering on your site.
Note that classid is the ID number of the class (not the identifier of the class).

5

RegistrationEmail

Summary

Sets the receiver of notification emails about new users.

Usage

```
RegistrationEmail= email_address
```

Description

If RegistrationEmail is left empty the setting [MailSettings], AdminEmail will be used instead.

Examples

```
RegistrationEmail=no@spam.org
```

eZ Publish will send notification emails about new users to "no@spam.org".

5

VerifyUserEmail

Summary

Sets if new users have to verify their account by email.

Usage

```
VerifyUserEmail=enabled|disabled
```

Description

If enabled, new users can't log in to your site before they have activated their account. An email will be sent to the new users with a link that activates their user.

5

RegistrationFeedback

Summary

Sets the kind of feedback that is sent to users that have registered.

Usage

```
RegistrationFeedback=email
```

Description

email is the only feedback type supported at the moment.

5

DefaultSectionID

Summary

Sets the section ID of self registered users.

Usage

```
DefaultSectionID= number
```

Description

This setting controls the section ID assigned to new self registering users. If DefaultSectionID is set to "0", new self registering users will get the section id from the parent object set by the DefaultUserPlacement (page 2596) setting.

5

DefaultUserPlacement

Summary

Sets the parent object of all users who register themselves

Usage

```
DefaultUserPlacement= nodeid
```

Description

DefaultUserPlacement should be set to the ID of the node that should hold all self registered users.

5

AnonymousUserID

Summary

Sets the eZ Publish user that should be used for anonymous page requests.

Usage

```
AnonymousUserID= userid
```

Description

When a user that is not logged in is accessing an eZ Publish installation, the request will be handled with the permissions of the anonymous user. You can select any user on your system to represent the anonymous user.
Take care to give the correct permissions to the anonymous user since these permissions are available for all users on your site.
Note that userid is equal to the ID number of the actual object that represents the user account.

5

GeneratePasswordLength

Summary

Sets the length of passwords generated by eZ Publish.

Usage

```
GeneratePasswordLength= number
```

Description

This setting controls the length of passwords generated if GeneratePasswordIfEmpty (page 2599)
is set to true

5

GeneratePasswordIfEmpty

Summary

Sets if eZ Publish should generate passwords if the password field is empty.

Usage

```
GeneratePasswordIfEmpty=true|false
```

Description

If GeneratePasswordIfEmpty is set to true, eZ Publish will automatically accept and generate passwords for users if the password field is empty. You can simply leave out the password field in the user register template.

If GeneratePasswordIfEmpty is set to false the user must select a password that is at least three characters long. Shorter passwords will not be accepted.

5

5.14.36   soap.ini

5.14.37   staticcache.ini

The configuration blocks are documented in the following sections:

- [CacheSettings] (page 2602)

5

[CacheSettings]

AlwaysUpdateArray   (page 2603)
    Sets the static caches that should always be updated when an object is published.

CachedURLArray   (page 2604)
    Sets the content that should be statically cached.

HostName   (page 2607)
    Sets the hostname of the machine generating the static content.

MaxCacheDepth   (page 2605)
    Sets the number of levels that will be statically cached relative to the root of your installation.

StaticStorageDir   (page 2606)
    Sets the directory where the static cache is stored.

5

AlwaysUpdateArray

Summary

Sets the static caches that should always be updated when an object is published.

Usage

```
AlwaysUpdateArray[]= url1
AlwaysUpdateArray[]= url2
...
```

Description

Each line specifies the url of a node that will have its static cache removed whenever an object is updated on the system.
In addition to this the static cache of each node is updated whenever the corresponding object is updated.

5

CachedURLArray

Summary

Sets the content that should be statically cached.

Usage

```
CachedURLArray[]= url1[*]
CachedURLArray[]= url2[*]
...
```

Description

Each line specifies one part of the content tree that will be statically cached. You can specify single nodes or complete subtrees using a wildcard (*). You must specify the full URL of the nodes, not the node names.

Note that the wildcard literally matches all characters. E.g the rule "/weblog*" will match both "/weblog/hello" and "/weblogs/hello".

Examples

```
CachedURLArray[]=/
CachedURLArray[]=/news*
CachedURLArray[]=/weblog*
```

This setup will statically cache the root node and all nodes under the node with the URL news and all nodes under the node with the URL weblog.

5

MaxCacheDepth

Summary

Sets the number of levels that will be statically cached relative to the root of your installation.

Usage

```
MaxCacheDepth= number
```

5

StaticStorageDir

Summary

Sets the directory where the static cache is stored.

Usage

StaticStorageDir= *directory_name*

Description

StaticStorageDir is set relative to the root directory of your eZ Publish installation. Note that you must adjust your rewrite rules if you change this setting.

Examples

StaticStorageDir=static

This setup will make eZ Publish store the statically cached files in "/static".

HostName

Summary

Sets the hostname of the machine generating the static content.

Usage

```
Hostname= hostname [: portnumber]
```

Description

HostName sets the host of the server that serves the uncached pages. The static cache feature uses this to retrieve the generated content if there is no cache available.

Examples

```
HostName=localhost
```

The server that generates the pages is available on localhost.

5

5.14.38 template.ini

The configuration blocks are documented in the following sections:

\begin{itemize} \item{} [AutoLinkOperator]\refer{label699674249a76e0cb055152b52e5932ca}
\item{}   [CharsetSettings]\refer{labela8dbcccf31a6ff542b1ed60b8063db35}
\item{}   [ControlSettings]\refer{label21a1c61eeae519cd7b2af55b0f150787}
\item{}   [PHP]\refer{labelf3c721bcaecbbed553c1b2a903668125}   \item{}
[SimpleTagsOperator]\refer{labelfd007bc7a2326a505fcca84544fc8453}   \item{}
[WashSettings]\refer{label30a1176f7322e1204768d6df517429f1} \end{itemize}

5

[SimpleTagsOperator]

Extensions

Summary

Sets the extensions where the simpletags template operator should look for extra includes.

Usage

```
Extensions[]=<extension_name>
```

Description

This setting lists the extensions which should be examined for extra includes for the simpletags template operator. The includes to search for defined in the setting [SimpleTagsOperator] IncludeList.

This setting is case sensitive.

Examples

```
Extensions[]=ezfind
```

5

IncludeList

Summary

Lists the PHP files that should be included when the "simpletags" operator is used.

Usage

```
IncludeList[]=<include name>
```

Description

This setting lists the PHP files that should be included when the "simpletags" operator is used.It will look in the kernel/classes/simpletags folder or the simpletags subdirectory of the extensions defined in the setting [SimpleTagsOperator] Extensions.

This setting is case sensitive.

This setting is left empty by default.

Examples

```
IncludeList[]=
```

5

TagList

Summary

Lists settings for specific tags.

Usage

```
TagList[<type>]=<tag>
```

Description

This setting will list the settings for specific tags.

This setting is case sensitive.

Examples

Example 1

```
TagList[text]=;;htmlspecialchars,nl2br
```

This is the default setting for everything that is considered normal text, for example text
not inside tags.

Example 2

```
TagList[literal]=<pre>;</pre>;htmlspecialchars
TagList[code]=<pre class="code">;</pre>;htmlspecialchars
TagList[strong]=<b>;</b>;htmlspecialchars,nl2br
TagList[emphasize]=<i>;</i>;htmlspecialchars,nl2br
```

These are the default settings for the literal, strong and emphasize tags.

5

TagList_{mytags}

Summary

Sets personal tag list.

Usage

```
TagList_<tag_list_name>[<tag_name>]=<tags>
```

Description

It is possible to create custom tag lists. Add the name of your tags (in this example "mytags") as the first parameter to simpletags, for example simpletags('mytags').   Next create a list named "TagList_<mytags>" where <mytags> is replaced by the name of the your tag list.

This setting is case sensitive.

Examples

```
TagList_mytags[text]=;;htmlspecialchars,nl2br
```

This a default setting.

5

[AutoLinkOperator]

MaxCharacters

Summary

This setting sets the maximum number of characters to show in autolink links.

Usage

```
MaxCharacters=<integer>
```

Description

This setting sets the maximum number of characters to show in autolink links. Specify the maximum with a number or leave it empty, if you wish to disable the function.When a link string is longer than the maximum value defined here it will be shortened.

This setting is case sensitive.

Examples

Example 1

```
MaxCharacters=72
```

Sets a maximum value of 72 characters.

Example 2

```
MaxCharacters=
```

By leaving the setting empty, the link string will not be shortened.

5

Methods

Summary

This setting lists the URL methods or protocols that should be matched.

Usage

```
Methods[]=<methode_name>
```

Description

This setting lists the URL methods or protocols that should be matched.

This setting is case sensitive.

Examples

```
Methods[]=http
Methods[]=https
Methods[]=ftp
Methods[]=ftps
Methods[]=webdav
Methods[]=webdavs
```

These are the default settings.

5

[PHP]

5

PHPOperatorList

Summary

This setting lists PHP functions and their equivalent template operator names.

Usage

```
PHPOperatorList[<PHP_function>]=<operator_name>
```

Description

This setting lists PHP functions and their equivalent template operator names. It is recommended to give operators a different name than the PHP functions, this ensures that all operators follow the same naming guidelines. An added bonus is that you can change the PHP function later on without being required to do template changes.Only one parameter is supported as value and the PHP function must return the result by value and not by reference.

This setting is case sensitive.

Examples

```
PHPOperatorList[urlencode]=urlencode
```

This is the default setting.

5

[WashSettings]

EmailDotText

Summary

Sets a replacement string for the "." (dot) in an email address.

Usage

```
EmailDotText=<replacement for dot>
```

Description

This setting sets a replacement string for the "." (dot) in an email address. This replacement is used to display email addresses in such a way that SPAM engines cannot fetch them automatically.

This setting is case sensitive.

Examples

Example1

```
EmailDotText=.
```

By default this setting is set to ".".**Example2**

```
EmailDotText=DOT (default value is "." (dot) default value is "." (dot)
-Ricardo Goulart 18/08/
10 11:25
EmailAtText=AT
```

With these configuration settings the email address publish@ez.no will be displayed as publishATezDOTno.

5

EmailAtText

Summary

Sets a replacement string for the "@" (at) in an email address.

Usage

```
EmailAtText=<replacement for at>
```

Description

This setting sets a replacement string for the "@" (at) in an email address. This replacement is used to display email addresses in such a way that SPAM engines cannot fetch automatically.

This setting is case sensitive.

Examples

Example 1

```
EmailAtText=<span class="spamfilter">SPAMFILTER</span>@
```

This is the default setting.

Example 2

```
EmailDotText=DOT
EmailAtText=AT
```

With these configuration settings the email address publish@ez.no will be displayed as publishATezDOTno.

5

[ControlSettings]

DisplayWarnings

Summary

Allows you to display template engine warnings.

Usage

```
DisplayWarnings=enabled | disabled
```

Description

This setting sets whether warnings from the template engine should be displayed on the page or not. Keep in mind that debug warnings are always shown if debugging is enabled. This setting is case sensitive.

Examples

```
DisplayWarnings=enabled
```

Enabled by default

5

AlloUserVariables

Summary

Sets whether user variables can automatically be assigned from the URL.

Usage

```
AllowUserVariables=true | false
```

Description

```
AllowUserVariables=true
```

This is the default and recommend value.

5

OldStyleUserVariables

Summary

This setting enables you to use the old style parsing of user variables in URLs.

Usage

```
OldStyleUserVariables=disabled | enabled
```

Description

This setting enables you to use the old style parsing of user variables in URLs. The new style user variables can contain a "/" (slash) and must be placed in the end of URL.The contents of each user variable starts after the variable name signed as "/<var-name>/" and ends by the next variable name or URLs end.

This setting is case sensitive.

Examples

```
OldStyleUserVariables=disabled
```

This setting is disabled by default.

5

MaxLevel

Summary

Sets the maximum level of template nesting.

Usage

```
MaxLevel=<integer>
```

Description

This setting sets a maximum level of template nesting. This is done in order to avoid infinite recursion.

This setting is case sensitive.

Examples

```
MaxLevel=40
```

This is the default setting.

5

[CharsetSettings]

DefaultTemplateCharset

Summary

Sets the default charset to use.

Usage

```
DefaultTemplateCharset=<charset_name>
```

Description

This setting sets which character set (charset) to use when no charset is specified in the template.The charset defined here will also be used by transform.ini to map charsets to a language group.This setting is case sensitive.

Examples

```
DefaultTemplateCharset=utf-8
```

This is the default setting.

5

DefaultTemplateLocale

Summary

Sets the default system locale.

Usage

```
DefaultTemplateLocale=<locale_name>
Note:  the default setting is empty.
```

Description

This setting sets the locale for the entire PHP system to use for character manipulations when no locale is specified in the template.If you use a character set specific locale, for example "no_NO.UTF-8", make sure this character set matches the output character set of eZ Publish. The locale to use depends on your operating system, because different operating systems have different naming schemes for locales. Therefore it is recommended to use multiple names in order for your installation to work on multiple platforms (for example developing on Windows and running Linux in production).When setting multiple locales the syntax is the following:

```
DefaultTemplateLocale=<locale_name>,<locale_name>,<locale_name>
```

Each locale is separated by a comma. The first locale listed will be the fist locale used, then the second, and so on.By default this setting is left empty. This setting is case sensitive.

Examples

Example 1

```
DefaultTemplateLocale=de_DE.ISO-8859-1,german
```

This will tell PHP that if no locale is specified in the template, to first use the "de_DE.ISO-8859-1" locale, and if this is not available, then the "german" locale will be used.

Example 2

```
DefaultTemplateLocale=no_NO.UTF-8,no_NO,norwegian
```

Sets the Norwegian locale. Each of the three listed elements, starting from the first, will be tried to be set as locale until success. The locale that will be used depends on your operating system

Example 3

5

```
DefaultTemplateLocale=C
```

C is a special locale which means 7-bit ASCII

5

AutoConvertOnSave

Summary

Sets autoconversion of the template charset.

Usage

```
AutoConvertOnSave=enabled | disabled
```

Description

This setting allows you to autoconvert the template charset to the HTTPCharset as set in the configuration file i18n.ini with the setting [CharacterSettings] HTTPCharset when saving a template file from the administration interface. It will add or adjust the template header when required.

This setting is case sensitive.

Examples

```
AutoConvertOnSave=enabled
```

This is the default setting

5

5.14.39   textfile.ini

The configuration blocks are documented in the following sections:

\begin{itemize} \item{}   [TextFileSettings]\refer{label652cd2feed547492bda68feae88f9fe3}
\end{itemize}

5

[TextFileSettings]

5

TextFileEnabled

Summary

Enables the textfile login handler.

Usage

```
TextFileEnabled=true | false
```

Description

To enable the textfile login handler, this setting must be set to true.

The textfile handler is enabled by default. This setting is case sensitive.

Examples

```
TextFileEnabled=true
```

This is the default setting.

5

FileName

Summary

Sets the file name.

Usage

```
FileName=<file_name>
```

Description

This setting sets the name of the text file.

This setting is case sensitive and left empty by default.

5

FilePath

Summary

Sets the file location.

Usage

```
FilePath=<file path>
```

Description

This setting sets the location where to find the text file defined with the setting FileName.   A value could be any directory of your eZ Publish installation root directory.If the FilePath is root, the file will need to be saved under the eZ Publish root directory.

This setting is case sensitive.

Examples

Example 1

```
FilePath=root
```

This is the default setting. The file will be located under the root directory of your eZ Publish installation.

Example 2

```
FilePath=var
```

The file will be located under the var directory of your eZ Publish installation.

5

FileFieldSeparator

Summary

Sets the separator.

Usage

```
FileFieldSeparator=<string separator>
```

Description

This setting sets the separator to use between the fields of a file name. Possible values include "tab", ";", ",", "\" and all other string separators.

Examples

Example 1

```
FileFieldSeparator=tab
```

This is the default setting. A tab will be used to separate the file fields.
Example 2

```
FileFieldSeparator=;
```

A ";" will be used to separate the file fields.

5

DefaultUserGroupType

Summary

Sets the default user group type.

Usage

```
DefaultUserGroupType=id | name
```

Description

This setting sets the default user group type. Possible values are "id" (user group ID) or "name" (user group name).

This setting is case sensitive.

Examples

```
DefaultUserGroupType=id
```

This is the default setting.

5

DefaultUserGroup

Summary

Sets the default user group location.

Usage

DefaultUserGroup=<location>

Description

This setting sets the default place to store users from the textfile. As value both the node ID can be used as well as the group name, which depends on the value of the DefaultUserGroupType setting.

This setting is case sensitive.

Examples

DefaultUserGroup=12

This is the default setting.

5

LoginAttribute

Summary

Sets the login fields column.

Usage

```
LoginAttribute=<field_column_number>
```

Description

This setting sets the field column number to use for login.

Examples

```
LoginAttribute=1
```

This is the default setting.

5

PasswordAttribute

Summary

Sets the password field column number.

Usage

```
PasswordAttribute=<field_column_number>
```

Description

This setting sets the field column number to use for password.

Examples

```
PasswordAttribute=3
```

This is the default setting.

5

FirstNameAttribute

Summary

Sets the first name column number.

Usage

FirstNameAttribute=<field_column_number>

Description

This setting sets the field column number to use for the first name.

Examples

FirstNameAttribute=4

This is the default setting.

5

LastNameAttribute

Summary

Sets the last name field column number.

Usage

```
LastNameAttribute=<field_column_number>
```

Description

This setting sets the field column number to use for the last name.

Examples

```
LastNameAttribute=5
```

This is the default setting.

5

EmailAttribute

Summary

Sets the email address field number.

Usage

```
EmailAttribute=<field_column_number>
```

Description

This setting sets the field column number to use for the email address.

Examples

```
EmailAttribute=2
```

This is the default setting.

5

5.14.40 texttoimage.ini

The configuration blocks are documented in the following sections:

\begin{itemize} \item{} [DefaultSettings]\refer{label47d54a5afffa42c65dd7508d22fb1dce} \item{} [ImageSettings]\refer{labela4f654ed1aafabdda8f08c084e8e42f1} \item{} [PathSettings]\refer{label28ae2da4db3173fb82e12118f63a7b90} \item{} [\{style\}]\refer{label8b27aa9b0e901de8526219cfe7ce5fd0} \end{itemize}

5

[ImageSettings]

UseCache

Summary

Enables or disables the use of cache files for text transformation.

Usage

```
UseCache=enabled | disabled
```

Description

This setting allows you to enable or disable the use of cache files for text transformation.
The cache files can be used to map text to what it needs to be transformed into. When
cache is not used (UseChace=disabled) the system will have to calculate the transformation each time.The location of the cache files needs to be defined with the setting [PathSettings]
CacheDir.

This setting is case sensitive.

Examples

```
UseCache=enabled
```

This setting is enabled by default.

5

[PathSettings]

5

FontDir

Summary

Sets the location of the folder that contains the fonts.

Usage

FontDir[]=<path_to_directory>

Description

This seting defines the location of the folder that contains the fonts.

This setting is case sensitive.

Examples

FontDir[]=design/standard/fonts

This is the default setting.

5

CacheDir

Summary

Sets the location of the folder that contains the relevant cache files.

Usage

CacheDir=<path_to_directory>

Description

This setting defines the location of the folder that contains the cache files of the text to image. These cache files are used when the setting [ImageSettings] UseChache is set to enabled. This setting is case sensitive.

Examples

 CacheDir=var/cache/texttoimage

This is the default location.

5

HtmlDir

Summary

Sets the location of the HTML folder that contains the cache files of the text to image.

Usage

HtmlDir=<path_to_directory>

Description

Sets the location of the HTML folder that contains the cache files of the text to image.
This setting is case sensitive.

Examples

 HtmlDir=/var/cache/texttoimage

This is the default location.

5

[DefaultSettings]

5

Family

Summary

This setting set the default font family to use

Usage

```
Family=<font_family_name>
```

Description

This setting sets the default font family to use. The value must have a related .ttf file in the directory defined by the setting FontDir in [PathSettings], for example archtura.ttf.

This setting is case sensitive.

Examples

```
Family=1942
```

Fonts defined with the file 1942.ttf will be used by default.

5

PointSize

Summary

Sets the default font size.

Usage

```
PointSize=<integer>
```

Description

This setting defines the default point size of the font. The value to use is the number of pixels.

Examples

```
PointSize=40
```

This is the default setting.

Angle

Summary

Sets the default font angle.

Usage

```
Angle=<number>
```

Description

This sets the default font angle. This angle defines how cursive the font is displayed. When set to "0" (zero) no angle will be used.

Examples

```
Angle=0
```

This is the default setting

5

XAdjustment

Summary

Sets the left/right placement of the text.

Usage

XAdjustment=<integer>

Description

This setting allows you to adjust the position or placement of the text when for example the rendering went wrong. The x adjustment will allow the text to begin from a specific location according to left/right alignment, which means it will define how far right in number of pixels the text will be displayed. The smaller the value the more left it will be positioned. By default it is set to "0" (zero), which means no adjustments are made and the text will be positioned according to the settings of its own font family.

Examples

XAdjustment=0

This is the default setting, which indicates no adjustments are made.

5

YAdjustment

Summary

Sets the top/bottom placement of the text.

Usage

```
YAdjustment=<integer>
```

Description

This setting allows you to adjust the position or placement of the text when for example the rendering went wrong.

The y adjustment will allow the text to begin from a specific location according to top/ bottom alignment, which means it will define how far up in number of pixels the text will be displayed. The smaller the value the lower it will be positioned. By default it is set to "0" (zero), which means no adjustments are made and the text will be positioned according to the settings of its own font family.

Examples

```
YAdjustment=0
```

This is the default setting, which indicates no adjustments are made.

5

WidthAdjustment

Summary

Sets width of the text.

Usage

```
WidthAdjustment=<integer>
```

Description

This setting allows you to adjust the size of the text, when for example the rendering went wrong.The width adjustment will allow you to define the width of the text in number of pixels, if it needs to differ from the default value. By default it is set to "0" (zero), which means no adjustments are made and the text will have the width according to the settings of its own font family.

Examples

```
WidthAdjustment=0
```

This is the default setting, which indicates no adjustments are made.

5

HeightAdjustment

Summary

Sets height of the text.

Usage

```
HeightAdjustment=<integer>
```

Description

This setting allows you to adjust the size of the text, when for example the rendering when wrong. The height adjustment will allow you to define the height of the text in number of pixels, if it needs to differ from the default value. By default it is set to "0" (zero), which means no adjustments are made and the text will have the height according to the settings of its own font family.

Examples

```
HeightAdjustment=0
```

This is the default setting, which indicates no adjustments are made.

5

BackgroundColor

Summary

Sets the default background color for text.

Usage

```
BackgroundColor=<HTML_color_code>
```

Description

Sets the default background color for text. The values which can be used are HTML color codes. HTML colors are defined using a hexadecimal notation (HEX) for the combination of Red, Green, and Blue color values (RGB).

This setting is case sensitive.

Examples

```
BackgroundColor=#ffffff
```

This is the default value, which indicates that the background color is white.

5

TextColor

Summary

Sets the default text color.

Usage

```
TextColor=<HTML_color_code>
```

Description

Sets the default text color. The values which can be used are HTML color codes. HTML colors are defined using a hexadecimal notation (HEX) for the combination of Red, Green, and Blue color values (RGB).

This setting is case sensitive.

Examples

```
TextColor=#000000
```

This is the default value, which indicates that the text color is black.

5

[{style}]

The default styles are:

- 1942

- a_d_mono

- archtura

- arial

- gallery

- object_text

- sketchy

- smartie

5

Family

Summary

Sets a specific font family for the style defined by the configuration block.

Usage

```
Family=<family_name>
```

Description

This setting sets a specific font family for the style defined by the configuration block.

The values used must have a related .ttf file in the directory defined by the setting FontDir in [PathSettings], for example archtura.ttf.

This setting is case sensitive.

Examples

```
[gallery]
 Family=bluehigc
 PointSize=18
 XAdjustment=0
 YAdjustment=0
 WidthAdjustment=8
 HeightAdjustment=6
 BackgroundColor=#ffffff
 TextColor=#000000
```

These settings define the specific settings for the "gallery" style. Fonts are used according to the specifications defined in bluehigc.tff (location of this file is set with FontDir in [PathSettings]). Although the fonts from the family "bluehigc" are used, the point size, width, height and colors are adjusted according to the settings here.

5

Angle

Summary

Sets a specific font angle for the style defined by the configuration block.

Usage

```
Angle=<number>
```

Description

This setting sets a specific font angle for the style defined by the configuration block. This angle defines how cursive the font is displayed. When set to "0" (zero) no angle will be used.

Examples

```
[a_d_mono]
 Family=a_d_mono
 Angle=0
 XAdjustment=0
 YAdjustment=-1
 WidthAdjustment=1
 HeightAdjustment=1
```

This is the default value for the a_d_mono style.

5

XAdjustment

Summary

Adjusts the left/right placement of the text for the specific style defined by the configuration block.

Usage

```
XAdjustment=<number>
```

Description

This setting allows you to adjust the position or placement of the text for the specific style defined by the configuration block.

The x adjustment will allow the text to begin from a specific location according to the left/right alignment defined here, which means it will define how far right in number of pixels the text will be displayed. The smaller the value the more left it will be positioned. By default it is set to "0" (zero), which means no adjustments are made and the text will be positioned according to the settings of its own font family.

Examples

```
[smartie]
 Family=smartie
 Angle=0
 XAdjustment=2
 YAdjustment=-1
 WidthAdjustment=0
 HeightAdjustment=3
```

For the style "smartie" the text will be positioned two pixels more to the right than defined by default.

5

YAdjustment

Summary

Adjusts the top/bottom placement of the text for the specific style defined by the configuration block.

Usage

```
YAdjustment=<number>
```

Description

This setting allows you to adjust the position or placement of the text for the specific style defined by the configuration block.

The y adjustment will allow the text to begin from a specific location according to top/bottom alignment defined here, which means it will define how far up in number of pixels the text will be displayed. The smaller the value the lower it will be positioned. By default it is set to "0" (zero), which means no adjustments are made and the text will be positioned according to the settings of its own font family.

Examples

```
[sketchy]
 Family=sketchy
 Angle=0
 XAdjustment=0
 YAdjustment=-1
 WidthAdjustment=0
 HeightAdjustment=1
```

For the style "sketchy" the text will be positioned one pixel below the default location.

5

WidthAdjuastment

Summary

Adjusts the width the text for the specific style defined by the configuration block.

Usage

```
WidthAdjustment=<number>
```

Description

This setting allows you to adjust the size of the text for the specific style defined by the configuration block.The width adjustment will allow you to define the width of the text in number of pixels, if it needs to differ from the default value. By default it is set to "0" (zero), which means no adjustments are made and the text will have the width according to the settings of its own font family.

Examples

```
[archtura]
 Family=archtura
 Angle=0
 XAdjustment=0
 YAdjustment=0
 WidthAdjustment=8
 HeightAdjustment=6
 BackgroundColor=#ffffff
 TextColor=#578ec5
```

For the style "archtura" the text will be 8 pixels wide and the text color is light blue.

5

HeightAdjustment

Summary

Adjust the height of the text for the specific style defined by the configuration block.

Usage

```
HeightAdjustment=<number>
```

Description

This setting allows you to adjust the size of the text for the specific style defined by the configuration block. The height adjustment will allow you to define the height of the text in number of pixels, if it needs to differ from the default value. By default it is set to "0" (zero), which means no adjustments are made and the text will have the height according to the settings of its own font family.

Examples

```
[1942]
 Family=1942
 Angle=0
 XAdjustment=-4
 YAdjustment=0
 WidthAdjustment=4
 HeightAdjustment=4
```

For the style "1942" the text will be 4 pixels high.

5

PointSize

Summary

This setting defines the point size for the specific style defined by the configuration block. Usage

Usage

```
PointSize=<integer>
```

Description

This setting defines the point size for the specific style defined by the configuration block.
The value to use is the number of pixels.

Examples

```
[gallery]
Family=bluehigc
PointSize=18
XAdjustment=0
YAdjustment=0
WidthAdjustment=8
HeightAdjustment=6
BackgroundColor=#ffffff
TextColor=#000000
```

The style gallery will use a point size of 18 pixels.

5

TextColor

Summary

Sets the text color for the specific style defined by the configuration block.

Usage

```
TextColor=<HTML_color_code>
```

Description

Sets the text color for the specific style defined by the configuration block. The values which can be used are HTML color codes. HTML colors are defined using a hexadecimal notation (HEX) for the combination of Red, Green, and Blue color values (RGB).

This setting is case sensitive.

Examples

```
[object_text]
 Family=bluehigc
 PointSize=15
 Angle=1
 XAdjustment=0
 YAdjustment=0
 WidthAdjustment=5
 HeightAdjustment=5
 BackgroundColor=#ffffff
 TextColor=#ff0000
```

For the style "object text" the text color will be red.

5

BackgroundColor

Summary

Sets the background color for text for the specific style defined by the configuration block.

Usage

```
BackgroundColor=<HTML_color_code>
```

Description

Sets the background color for text for the specific style defined by the configuration block. The values which can be used are HTML color codes. HTML colors are defined using a hexadecimal notation (HEX) for the combination of Red, Green, and Blue color values (RGB).

This setting is case sensitive.

Examples

```
[arial]
 Family=arial
 PointSize=30
 XAdjustment=0
 YAdjustment=0
 WidthAdjustment=8
 HeightAdjustment=6
 BackgroundColor=#ffffff
 TextColor=#000000
```

For the style "arial" the background color is white.

5

5.14.41   toolbar.ini

5.14.42   transform.ini

The configuration blocks are documented in the following sections:

- [{transformation group}] (page 2674)
- [Transformation] (page 2683)

5

[{transformation group}]

This is an example group.

Each groups set in the setting Groups[] in block [Transformation], needs to have its own block. The settings here are available for the default groups latin, cyrillic, greek, hebrew, search, urlalias_iri, urlalias, urlalias_compat, identifier, uppercase, lowercase and custom groups.

5

Commands

Summary

Sets character transformation rules.

Usage

```
Commands[]=<transformation_command1_groupname>
```

Description

The Commands setting allows you to define the rules of character transformation and define them to the corresponding groups. These commands will be used by the system in addition to the rules defined in the transformation files if any are specified to a certain group.
It is also possible to specify parameters for the commands, these will allowed the system to look for the defined rule in all transformation files (not only the loaded ones). If not specified all corresponding transformation rules will be used.
The following list contains the possible transformation commands:

- normalize: this command normalizes multiple characters with similar meaning to one type. For each parameter, it looks for rules named:

```
<parameter>_normalize
```

  If no parameters are used it will use all normalize rules.

- search_normalize: this command runs normalization rules required for search engine. For each parameter, it looks for rules named:

```
<parameter>_search_normalize
```

  If no parameters are used it will use all search rules.

- transform: this command transforms one type of character into another. For each parameter, it looks for rules named:

```
<parameter1>_to_<parameter2>
```

  If no parameters are used it will find all transformation rules.

- decompose: this command turns special characters into strings representing the character. For each parameter, it looks for rules named:

5

```
<parameter>_decompose
```

If no parameters are used it will find all decompose rules.

- lowercase: this command turns characters into their lowercase variant if possible. For each parameter, it looks for rules named:

```
<parameter>_lowercase
```

If no parameters are used it will find all lowercase rules.

- uppercase: this command turns characters into their uppercase variant if possible. For each parameter, it looks for rules named:

```
<parameter>_uppercase
```

If no parameters are used it will find all uppercase rules.

- transliterate: this command converts from one type of charset to another by finding similar characters or strings. For each parameter, it looks for rules named:

```
<parameter1>_transliterate_<parameter2>
```

If no parameters are used it will find all transliteration rules

- diacritical: this command removes diacriticals from characters. For each parameter, it looks for rules named:

```
<parameter>_diacritical
```

If no parameters are used it will find all diacritical rules.

- rule: this command runs custom transformation rule. For each parameter, it looks for rules named:

```
<parameter>
```

If no parameters are used it will do nothing.

- url_cleanup: this command performs cleanups to make the text presentable in a URL. No parameters can be passed for this command.

- identifier_cleanup: this command performs cleanups to make the text usable as an identifier. No parameters can be passed for this command.

- search_cleanup: this command performs cleanup to make the text parsable by the search engine. For each parameter, it looks for rules named:

5

```
<parameter>_search_cleanup
```

If no parameters are used it will find all search_cleanup rules.

It is also possible to create custom rules in order to transform characters according to your own specific needs.

Examples

example 1

```
[urlalias_iri]

Commands[]
Commands[]=url_cleanup_iri
```

This is the default setting for commands in [urlalias_in]. This command will perform an IRI cleanup only, which means that almost all unicode characters are allowed in URLs with the exceptions being space, ampersand and semi-colon.

example 2

```
[search]

Commands[]
Commands[]=decompose
Commands[]=normalize
Commands[]=diacritical
Commands[]=search_normalize
Commands[]=lowercase
Commands[]=transform(inverted,normal)
Commands[]=transform(math,ascii)
Commands[]=search_cleanup
```

These are the default commands to use for search.

example 3

```
[urlalias]

Commands[]
Commands[]=url_cleanup
Commands[]=transform
```

This configuration means that when the setting TransformationGroup in Site.ini is set to "urlalias", the system will use two commands, url cleanup and transform, in addition to the rules from the transformation files that are loaded by default.

Example 4

```
[transformation]

Groups[]
Groups[]=my_group

[my_group]

Files[]

Extensions[]

Commands[]

Commands[]=transform(math,ascii)
```

This configuration sets a new custom group "my group" that will use the transform command with a parameter "math to ascii". The transformation rules for this case are defined in the corresponding section of "latin.tr" transformation file and will turn some special math symbols into ASCII equivalents.

5

Extensions

Summary

Sets from which extensions the additional transformation files should be used, for this group.

Usage

```
Extensions[]=<name extension>
```

Description

This setting defines extensions that contain transformation files, which should be used for this group. Therefore this setting enables you to set additional rules of character transformation and matching them to the corresponding groups by using extensions. Transformation files located in extensions can also be used to override transformation rules that are used by default. If any extensions are specified to the current group, the system will search for the transformation files in the "transformation" directory of extension.

Note, that if the rules from the default transformation files match the rules from the extension transformation files for the current group, the first ones will be overridden by the last.

Examples

```
Extensions[]=my_extension
```

With this configuration the rules from the "my_extension" extension will override the matching rules from the defined transformation files.

5

Files

Summary

Sets which transformation files are used by this transformation group (block name).

Usage

```
Files[]=<file.tr>
```

Description

Sets the transformation files that are used by this transformation group (the block name), and from which the rules of character transformation should be taken from.

The process of selecting the proper files which are to be used by the system, is described here through an example. Let's say that the current eZ Publish installation uses the "utf-8" charset.

On the first step the system will go through the Charset setting located under the [Transformation] block, which consists of two parts, the charset and the group it maps to, separated with a semicolon. If the first part matches the current charset, the second part is added to the list of blocks that will be used.

For the "utf-8" charset there will be four default matches in Charset:

```
Charsets[]=utf-8;latin
Charsets[]=utf-8;cyrillic
Charsets[]=utf-8;greek
Charsets[]=utf-8;hebrew
```

During the second step, the system will search for the transformation files in the corresponding blocks:

```
[latin]

Files[]=latin.tr
Extensions[]=my_extension

[cyrillic]

Files[]=cyrillic.tr
Extensions[]

[greek]

Files[]=greek.tr
Extensions[]
```

5

```
[hebrew]

Files[]=hebrew.tr
Extensions[]
```

Therefore, there are four transformation files to load that are mentioned in the Files setting (latin.tr, cyrillic.tr, greek.tr and hebrew.tr), also transformation files linked to the used extensions will be loaded.

The repository setting controls where the transformation files mentioned in the Files setting are located. The system will also look for the transformation files in the extensions defined in the Extensions setting, more specifically in the extensions "transformations" folder (for example "my_extension/transformations/transform.tr").

So far five transformation files will be loaded by the system according to our example settings:
latin.tr
cyrillic.tr
greek.tr
hebrew.tr
transform.tr

On the third step the system will look through the other relevant blocks in transform.ini and add the transformation files to the list, if any are specified. Thus, if for example the search engine is working on request and the [search] block in transform.ini looks like this:

```
[search]

Files[]
Files[]=search.tr
```

The system will then also load the "search.tr" file in addition to the other five files that are described above.

So now, when the search engine is working on a request and the current charset of eZ Publish is set to "utf-8", the system will now take the rules of character transformation from the following transformation files:
latin.tr
cyrillic.tr
greek.tr
hebrew.tr
transform.tr
search.tr

In case these different loaded files have the same transformation rules, the system will use the rules from the file that was loaded last. Thus, in the following example the matching rules from "latin.tr" will be overridden with the rules from the "my_extension" extension.

```
[latin]

Files[]
Files[]=latin.tr
```

```
Extensions[]
Extensions[]=my_extension
```

This concludes our example.

The following files are located in the "share/transformations" folder, and can thus be used as value:

- basic.tr: contains the basic formatting rules.

- ascii.tr: contains the rules of transformation for ACII handling, which is the basis for all characters.

- cyrillic.tr: contains the rules of transformation related to Cyrillic characters.

- greek.tr: contains the rules of transformation related to Greek characters.

- hebrew.tr: contains the rules of transformation related to Hebrew characters.

- latin.tr: contains the rules of transformation related to Latin characters

- search.tr: contains the rules of transformation related to search handling, which is used to normalize search requests to make it clear for the search engine.

Examples

example 1

```
[latin]

Files[]=latin.tr
```

In this block the special transformations for latin (latin1, latin extended-a, latin extended-b) are set. And the extra transformation file latin.tr will be used for this group.

example 2

```
[cyrillic]

Files[]=cyrillic.tr
```

In this block the special transformations for cyrillic are set. And the extra transformation file cyrillic.tr will be used for this group.

5

[Transformation]

Settings to control character transformation rules.

Charsets   (page 2684)
   Matches a charset to a group.

Extensions   (page 2689)
   Sets from which extensions the additional transformation files should be used.

Files   (page 2690)
   Sets which transformation files are used by the default transformation group.

Groups   (page 2686)
   Sets the available groups to define the rules of transformation for.

Repository   (page 2693)
   Sets the directory where the default character transformation files are stored.

5

Charsets

Summary

Matches a charset to a group.

Usage

```
Charsets[<group ID>]=<charset>;<group>
```

Description

This setting maps a charset to a group. The group identifier can be used to load certain extra transformation files when that charset is used.
Each entry consists of the following syntax:

```
Charsets[<group ID>]=<charset>;<group>
```

As explained in the setting File in block [Transformation], the process of selecting the proper files to use by the system starts with the system checking the Charset setting. This first step is best described through an example.
Let's say that the current eZ Publish installation uses the "utf-8" charset. The first step the system will take, is to check the Charset setting. This setting consists of two parts, the charset and the group it maps to, separated with a semicolon. If the first part matches the current charset, the second part is added to the list of blocks that will be used.
So if we for example have the following Charsets settings:

```
[Transformation]

Charsets[]=cp437;latin
Charsets[]=utf-8;latin
Charsets[]=utf-8;cyrillic
Charsets[]=cp437;greek
Charsets[]=windows-1255;hebrew
Charsets[]=utf-8;hebrew
```

Then the system would find three matches for the "utf-8" charset, the groups latin, cyrillic and hebrew. Next the system will search for the transformation files in the corresponding blocks [latin], [cyrillic], [hebrew].

Examples

```
Charsets[]=koi8;cyrillic
Charsets[]=koi8-r;cyrillic
Charsets[]=koi8-u;cyrillic
```

```
Charsets[]=iso-8859-5;cyrillic
Charsets[]=cp932;cyrillic
Charsets[]=cp866;cyrillic
Charsets[]=cp855;cyrillic
Charsets[]=windows-1251;cyrillic
Charsets[]=utf-8;cyrillic
```

These are the default settings for the cyrrilic group.

5

Groups

Summary

Sets the available groups to define the rules of transformation for.

Usage

```
Groups[]=<name group>
```

Description

The groups setting allows you to defines the available transformation groups and specify the rules of character transformation for them.
Each group uses its own set of transformation files, extensions and additional transformation rules called Commands. These groups are also used by the search engine and by some datatypes. In addition it is also possible to use the defined groups in custom code.
For each group set here, the rules that should be used for this group should be defined in their own block. For example if Groups[]=latin is set here, then the rules for this group need to be defined in block [latin].
Groups available by default are:

- latin: Sets the transformation rules that can be used if the current charset handles Latin. By default this group uses the transformation file "latin.tr".

- cyrillic: Sets the transformation rules that can be used if the current charset handles Cyrillic. By default this group uses the transformation file "cyrillic.tr".

- greek: Sets the transformation rules that can be used if the current charset handles Greek. By default this group uses the transformation file "greek.tr".

- hebrew: Sets the transformation rules that can be used if the current charset handles Hebrew. By default this group uses the transformation file "hebrew.tr".

- search: Sets the transformation rules that can be used to normalize the search requests and to make it easier for the search engine to interpret it. By default this group uses the transformation file "search.tr".

- urlalias_iri: Sets the transformation rules that can be used to make sure the text of URLs is transformed into characters supported by IRI (for more information please visit http://en.wikipedia.org/wiki/Internationalized_Resource_Identifier)

- urlalias: Sets the transformation rules that can be used to make sure the characters not supported by the URI specialization (for more information please visit http://en.wikipedia.org/wiki/Uniform_Resource_Identifier) are transformed or removed when the URL alias system is making an URL.

- urlalias_compat: Sets the transformation rules that can be used to make sure the text of URLs is transformed into characters supported by this method of transformation.

- identifier: Sets the transformation rules that can be used to make sure the text of identifiers is transformed into ASCII equivalent characters.

- uppercase: Sets the transformation rules that can be used to make sure the characters are transformed into the uppercase.

- lowercase: Sets the transformation rules that can be used to make sure the characters are transformed into the lowercase.

It is also possible to create custom groups and add custom sets of transformation rules to them, in order to transform characters according to specific needs.

Please refer to the TransformationGroup setting in site.ini to learn more about the URL's text transformation methods.

Examples

Example 1

```
Groups[]
Groups[]=search
Groups[]=urlalias_iri
Groups[]=urlalias
Groups[]=urlalias_compat
Groups[]=identifier
Groups[]=uppercase
Groups[]=lowercase
```

This is the default setting.

Example 2

```
[Transformation]

Groups[]
Groups[]=search

[search]

Files[]
Files[]=search.tr

Extensions[]
```

5

```
Commands[]
Commands[]=search_normalize
Commands[]=lowercase
```

This configuration sets a group "search" which uses "search_normalize" and "lowercase" transformation rules in addition to the rules defined in the "search.tr" transformation file.

Example 3

```
[Transformation]

Groups[]
Groups[]=my_group

[my_group]

Files[]

Extensions[]

Commands[]
Commands[]=normalize
Commands[]=diacritical
```

This configuration sets a custom group "my_group" which uses "normalize" and "diacritical" transformation rules.

5

Extensions

Summary

Sets from which extensions the additional transformation files should be used.

Usage

```
Extensions[]=<name extension>
```

Description

This setting defines extensions that contain transformation files, hence enabling you to set additional rules of character transformation and define them to the corresponding groups by using extensions. Transformation files located in extensions can also be used to override transformation rules that are used by default. If any extensions are specified for the current group, the system will search for the transformation files in the "transformation" directory of the extension.

Note, that if the rules from the default transformation files match the rules from the extension transformation files, the first ones will be overridden by the last.

Examples

```
[Transformation]

Files[]
Files[]=basic.tr

Extensions[]
Extensions[]=my_extension
```

With this configuration the matching rules from "basic.tr" will be overridden with the rules from the "my_extension" extension.

5

Files

Summary

Sets which transformation files are used by the default transformation group.

Usage

```
Files[]=<file.tr>
```

Description

This setting sets which default transformation files the system will use to take the rules of character transformation from. For the specific groups the files are matched to the corresponding transformation groups controlled by the Groups setting. For example:

```
[Transformation]

Groups[]
Groups[]=groupname

Files[]
Files[]=latin.tr
Files[]=greek.tr

[groupname]

Files[]=hebrew.tr
```

The process of selecting the proper files, which are to be used by the system, is described here through the following example.

Let's say that the current eZ Publish installation uses the "utf-8" charset.
On the first transformation step the system will go through the Charset setting located under the [Transformation] block which consists of two parts, charset and group, separated by a semicolon. If the first part matches the current charset, the second part is added to the list of blocks that will be used to look for transformation files.
For the "utf-8" charset there will by default be four matches in "Charset":

```
Charsets[]=utf-8;latin
Charsets[]=utf-8;cyrillic
Charsets[]=utf-8;greek
Charsets[]=utf-8;hebrew
```

During the second step, the system will search for the transformation files in the corresponding blocks:

5

```
[latin]

Files[]=latin.tr
Extensions[]=my_extension

[cyrillic]

Files[]=cyrillic.tr
Extensions[]

[greek]

Files[]=greek.tr
Extensions[]

[hebrew]
Files[]=hebrew.tr
Extensions[]
```

Consequently, there are four transformation files to load that are mentioned in the Files setting (latin.tr, cyrillic.tr, greek.tr and hebrew.tr), as well as the transformation files that will be loaded based on the used extensions.

The Repository setting controls where the transformation files mentioned in the Files setting are located. The system will also look for the transformation files in the extensions defined in the Extensions setting, more specifically in the extensions "transformations" folder (for example "my_extension/transformations/transform.tr").

So far five transformation files will be loaded by the system according to our example settings (plus possible files based on the extensions):

latin.tr
cyrillic.tr
greek.tr
hebrew.tr
transform.tr

On the third step the system will look through the other relevant blocks in transform.ini and add the transformation files to the list, if any are specified. Thus, if for example the search engine is working on request and the [search] block in transform.ini looks like this:

```
[search]
Files[]
Files[]=search.tr
```

The system will also load the "search.tr" file in addition to the other five files that are described above.

So now, when the search engine is working on a request and the current charset of eZ Publish is set to "utf-8", the system will now take the rules of character transformation from the following transformation files:

latin.tr
cyrillic.tr
greek.tr
hebrew.tr
transform.tr
search.tr

In case these different loaded files have the same transformation rules, the system will use the rules from the file that was loaded last. Thus, in the following example the matching rules from "latin.tr" will be overridden with the rules from the "my_extension" extension.

```
[latin]

Files[]
Files[]=latin.tr

Extensions[]
Extensions[]=my_extension
```

This concludes our example.

The following files are located in the "share/transformations" folder, and can thus be used as value:

- basic.tr: contains the basic formatting rules.

- ascii.tr: contains the rules of transformation for ACII handling, which is the basis for all characters.

- cyrillic.tr: contains the rules of transformation related to Cyrillic characters.

- greek.tr: contains the rules of transformation related to Greek characters.

- hebrew.tr: contains the rules of transformation related to Hebrew characters.

- latin.tr: contains the rules of transformation related to Latin characters

- search.tr: contains the rules of transformation related to search handling, which is used to normalize search requests to make it clear for the search engine.

Examples

```
Files[]
Files[]=basic.tr
Files[]=ascii.tr
```

This is the default setting.

Repository

Summary

Sets the directory where the default character transformation files are stored.

Usage

```
Repository=directory_name
```

Description

This setting sets where the default character transformation files (*.tr) are stored. The default value is "share/transformations".

All transformation files end with .tr.

Examples

```
Repository=share/transformations
```

This is the default setting.

5

5.14.43   units.ini

The configuration blocks are documented in the following sections:

- [Base] (page 2731)

- [BinaryPrefixes] (page 2724)

- [DecimalPrefixes] (page 2695)

- [FakeBinaryPrefixes] (page 2717)

5

[DecimalPrefixes]

This configuration block contains settings related to decimal prefixes.

atto (page 2699)
Sets the base unit for attoo.

centi (page 2705)
Sets the base unit for centi.

deca (page 2707)
Sets the base unit for deca.

deci (page 2706)
Sets the base unit for deci.

exa (page 2714)
Sets the base unit for exa.

femto (page 2700)
Sets the base unit for femto.

giga (page 2711)
Sets the base unit for giga.

hecto (page 2708)
Sets the base unit for hecto.

kilo (page 2709)
Sets the base unit for kilo.

mega (page 2710)
Sets the base unit for mega.

micro (page 2703)
Sets the base unit for micro.

milli (page 2704)
Sets the base unit for milli.

nano (page 2702)
Sets the base unit for nano.

peta (page 2713)
Sets the base unit for peta.

pico (page 2701)
Sets the base unit for kilo.

tera (page 2712)
Sets the base unit for tera.

yocto   (page 2697)
  Sets the base unit for yocto.

yotta   (page 2716)
  Sets the base unit for yotta.

zepto   (page 2698)
  Sets the base unit for zepto.

zetta   (page 2715)
  Sets the base unit for zetta.

5

yocto

Summary

Sets the base unit for yocto.

Usage

```
yocto=<factor>;<symbol>
```

Description

This setting sets the factor value and the symbol for an array to use for yocto. The factor is used for calculating values in the power of 10.
This setting is case sensitive.

Examples

```
yocto=-24;y
```

This is the default setting, indicating that the base unit for yocto is "y" and the factor is 10^{-24}.

5

zepto

Summary

Sets the base unit for zepto.

Usage

```
zepto=<factor>;<symbol>
```

Description

This setting sets the factor value and the symbol for an array to use for zepto. The factor is used for calculating values in the power of 10.
This setting is case sensitive.

Examples

```
zepto=-21;z
```

This is the default setting, indicating that the base unit for zepto is "z" and the factor is 10^{-21}.

5

atto

Summary

Sets the base unit for attoo.

Usage

```
atto=<factor>;<symbol>
```

Description

This setting sets the factor value and the symbol for an array to use for atto. The factor is used for calculating values in the power of 10.
This setting is case sensitive.

Examples

```
atto=-18;a
```

This is the default setting, indicating that the base unit for atto is "a" and the factor is 10^{-18}.

5

femto

Summary

Sets the base unit for femto.

Usage

```
femto=<factor>;<symbol>
```

Description

This setting sets the factor value and the symbol for an array to use for femto. The factor is used for calculating values in the power of 10.
This setting is case sensitive.

Examples

```
femto=-15;f
```

This is the default setting, indicating that the base unit for femto is "f" and the factor is 10^{-15}.

5

pico

Summary

Sets the base unit for kilo.

Usage

```
pico=<factor>;<symbol>
```

Description

This setting sets the factor value and the symbol for an array to use for pico. The factor is used for calculating values in the power of 10.
This setting is case sensitive.

Examples

```
pico=-12;p
```

This is the default setting, indicating that the base unit for pico is "p" and the factor is 10^{-12}.

5

nano

Summary

Sets the base unit for nano.

Usage

```
nano=<factor>;<symbol>
```

Description

This setting sets the factor value and the symbol for an array to use for nano. The factor is used for calculating values in the power of 10.
This setting is case sensitive.

Examples

```
nano=-9;n
```

This is the default setting, indicating that the base unit for nano is "n" and the factor is 10^{-9}.

5

micro

Summary

Sets the base unit for micro.

Usage

```
micro=<factor>;<symbol>
```

Description

This setting sets the factor value and the symbol for an array to use for micro. The factor is used for calculating values in the power of 10.
This setting is case sensitive.

Examples

```
micro=-6;
```

This indicates that the base unit for micro is "" and the factor is 10^{-6}.

5

milli

Summary

Sets the base unit for milli.

Usage

```
milli=<factor>;<symbol>
```

Description

This setting sets the factor value and the symbol for an array to use for milli. The factor is used for calculating values in the power of 10.
This setting is case sensitive.

Examples

```
milli=-3;m
```

This is the default setting, indicating that the base unit for milli is "m" and the factor is 10^{-3}.

5

centi

Summary

Sets the base unit for centi.

Usage

```
centi=<factor>;<symbol>
```

Description

This setting sets the factor value and the symbol for an array to use for centi. The factor is used for calculating values in the power of 10.
This setting is case sensitive.

Examples

```
centi=-2;c
```

This is the default setting, indicating that the base unit for centi is "c" and the factor is 10^{-2}.

deci

Summary

Sets the base unit for deci.

Usage

```
deci=<factor>;<symbol>
```

Description

This setting sets the factor value and the symbol for an array to use for deci. The factor is used for calculating values in the power of 10.
This setting is case sensitive.

Examples

```
deci=-1;d
```

This is the default setting, indicating that the base unit for deci is "d" and the factor is 10^{-1}.

5

deca

Summary

Sets the base unit for deca.

Usage

```
deca=<factor>;<symbol>
```

Description

This setting sets the factor value and the symbol for an array to use for deca. The factor is used for calculating values in the power of 10.
This setting is case sensitive.

Examples

```
deca=1;da
```

This is the default setting, indicating that the base unit for deca is "da" and the factor is 10^1.

5

hecto

Summary

Sets the base unit for hecto.

Usage

```
hecto=<factor>;<symbol>
```

Description

This setting sets the factor value and the symbol for an array to use for hecto. The factor is used for calculating values in the power of 10.
This setting is case sensitive.

Examples

```
hecto=2;h
```

This is the default setting, indicating that the base unit for hecto is "h" and the factor is 10^2.

5

kilo

Summary

Sets the base unit for kilo.

Usage

```
kilo=<factor>;<symbol>
```

Description

This setting sets the factor value and the symbol for an array to use for kilo. The factor is used for calculating values in the power of 10.
This setting is case sensitive.

Examples

```
kilo=3;k
```

This is the default setting, indicating that the base unit for kilo is "k" and the factor is 10^3.

5

mega

Summary

Sets the base unit for mega.

Usage

```
mega=<factor>;<symbol>
```

Description

This setting sets the factor value and the symbol for an array to use for mega. The factor is used for calculating values in the power of 10.
This setting is case sensitive.

Examples

```
mega=6;M
```

This is the default setting, indicating that the base unit for mega is "M" and the factor is 10^6.

5

giga

Summary

Sets the base unit for giga.

Usage

```
giga=<factor>;<symbol>
```

Description

This setting sets the factor value and the symbol for an array to use for giga. The factor is used for calculating values in the power of 10.
This setting is case sensitive.

Examples

```
giga=9;G
```

This is the default setting, indicating that the base unit for giga is "G" and the factor is 10^9.

5

tera

Summary

Sets the base unit for tera.

Usage

```
tera=<factor>;<symbol>
```

Description

This setting sets the factor value and the symbol for an array to use for tera. The factor is used for calculating values in the power of 10.
This setting is case sensitive.

Examples

```
tera=12;T
```

This is the default setting, indicating that the base unit for tera is "T" and the factor is 10^{12}.

5

peta

Summary

Sets the base unit for peta.

Usage

```
peta=<factor>;<symbol>
```

Description

This setting sets the factor value and the symbol for an array to use for peta. The factor is used for calculating values in the power of 10.
This setting is case sensitive.

Examples

```
peta=15;P
```

This is the default setting, indicating that the base unit for peta is "P" and the factor is 10^{15}.

5

exa

Summary

Sets the base unit for exa.

Usage

```
exa=<factor>;<symbol>
```

Description

This setting sets the factor value and the symbol for an array to use for exa. The factor is used for calculating values in the power of 10.
This setting is case sensitive.

Examples

```
exa=18;E
```

This is the default setting, indicating that the base unit for exa is "E" and the factor is 10^{18}.

5

zetta

Summary

Sets the base unit for zetta.

Usage

```
zetta=<factor>;<symbol>
```

Description

This setting sets the factor value and the symbol for an array to use for zetta. The factor is used for calculating values in the power of 10.
This setting is case sensitive.

Examples

```
zetta=21;5
```

This is the default setting, indicating that the base unit for zetta is "Z" and the factor is 10^{21}.

5

yotta

Summary

Sets the base unit for yotta.

Usage

```
yotta=<factor>;<symbol>
```

Description

This setting sets the factor value and the symbol for an array to use for yotta. The factor is used for calculating values in the power of 10.
This setting is case sensitive.

Examples

```
yotta=24;Y
```

This is the default setting, indicating that the base unit for yotta is "Y" and the factor is 10^{24}.

5

[FakeBinaryPrefixes]

This configuration block contains settings related to fake binary prefixes.

exa   (page 2718)
Sets the base unit for exa.

giga   (page 2721)
Sets the base unit for giga.

kilo   (page 2723)
Sets the base unit for kilo.

mega   (page 2722)
Sets the base unit for mega.

peta   (page 2719)
Sets the base unit for peta.

tera   (page 2720)
Sets the base unit for tera.

5

exa

Summary

Sets the base unit for exa.

Usage

```
exa=<factor>;<symbol>
```

Description

This setting sets the factor value and the symbol for an array to use for exa. The factor is used for calculating values in the power of 2.
This setting is case sensitive.

Examples

```
exa=60;E
```

This is the default setting, indicating that the base unit for exa is "E" and the factor is 2^{60}.

5

peta

Summary

Sets the base unit for peta.

Usage

```
peta=<factor>;<symbol>
```

Description

This setting sets the factor value and the symbol for an array to use for peta. The factor is used for calculating values in the power of 2.
This setting is case sensitive.

Examples

```
peta=50;P
```

This is the default setting, indicating that the base unit for peta is "P" and the factor is 2^{50}.

5

tera

Summary

Sets the base unit for tera.

Usage

```
tera=<factor>;<symbol>
```

Description

This setting sets the factor value and the symbol for an array to use for tera. The factor is used for calculating values in the power of 2.
This setting is case sensitive.

Examples

```
tera=40;T
```

This is the default setting, indicating that the base unit for tera is "T" and the factor is 2^{40}.

5

giga

Summary

Sets the base unit for giga.

Usage

```
giga=<factor>;<symbol>
```

Description

This setting sets the factor value and the symbol for an array to use for giga. The factor is used for calculating values in the power of 2.
This setting is case sensitive.

Examples

```
giga=30;G
```

This is the default setting, indicating that the base unit for giga is "G" and the factor is 2^{30}.

5

mega

Summary

Sets the base unit for mega.

Usage

```
mega=<factor>;<symbol>
```

Description

This setting sets the factor value and the symbol for an array to use for mega. The factor is used for calculating values in the power of 2.
This setting is case sensitive.

Examples

```
mega=20;M
```

This is the default setting, indicating that the base unit for mega is "M" and the factor is 2^{20}.

5

kilo

Summary

Sets the base unit for kilo.

Usage

```
kilo=<factor>;<symbol>
```

Description

This setting sets the factor value and the symbol for an array to use for kilo. The factor used for calculating values in the power of 2.
This setting is case sensitive.

Examples

```
kilo=10;k
```

This is the default setting, indicating that the base unit for kilo is "k" and the factor is 2^{10}.

5

[BinaryPrefixes]

This configuration block contains settings related to binary prefixes.

exbi (page 2725)
  Sets the base unit for exabyte.

gibi (page 2728)
  Sets the base unit for gigabyte.

kibi (page 2730)
  Sets the base unit for kilobyte.

mebi (page 2729)
  Sets the base unit for megabyte.

pebi (page 2726)
  Sets the base unit for petabyte.

tebi (page 2727)
  Sets the base unit for terabyte.

5

exbi

Summary

Sets the base unit for exabyte.

Usage

```
exbi=<factor>;<symbol>
```

Description

This setting sets the factor value and the symbol for an array to use for exabyte. The factor used for calculating values in the power of 2.
This setting is case sensitive.

Examples

```
exbi=60;Ei
```

This is the default setting, indicating that the base unit for kilobyte is "Ei" and the factor is 2^{60}.

5

pebi

Summary

Sets the base unit for petabyte.

Usage

```
pebi=<factor>;<symbol>
```

Description

This setting sets the factor value and the symbol for an array to use for petabyte. The factor used for calculating values in the power of 2.
This setting is case sensitive.

Examples

```
pebi=50;Pi
```

This is the default setting, indicating that the base unit for petabyte is "Pi" and the factor is 2^{50}.

5

tebi

Summary

Sets the base unit for terabyte.

Usage

```
tebi=<factor>;<symbol>
```

Description

This setting sets the factor value and the symbol for an array to use for terabyte. The factor used for calculating values in the power of 2.
This setting is case sensitive.

Examples

```
tebi=40;Ti
```

This is the default setting, indicating that the base unit for terabyte is "Ti" and the factor is 2^{40}.

5

gibi

Summary

Sets the base unit for gigabyte.

Usage

```
gibi=<factor>;<symbol>
```

Description

This setting sets the factor value and the symbol for an array to use for gigabyte. The factor used for calculating values in the power of 2.
This setting is case sensitive.

Examples

```
gibi=30;Gi
```

This is the default setting, indicating that the base unit for gigabyte is "Gi" and the factor is 2^{30}.

5

mebi

Summary

Sets the base unit for megabyte.

Usage

```
mebi=<factor>;<symbol>
```

Description

This setting sets the factor value and the symbol for an array to use for megabyte. The factor used for calculating values in the power of 2.
This setting is case sensitive.

Examples

```
mebi=20;Mi
```

This is the default setting, indicating that the base unit for megabyte is "Mi" and the factor is 2 20.

5

kibi

Summary

Sets the base unit for kilobyte.

Usage

```
kibi=<factor>;<symbol>
```

Description

This setting sets the factor value and the symbol for an array to use for kilobyte. The factor used for calculating values in the power of 2.
This setting is case sensitive.

Examples

```
kibi=10;Ki
```

This is the default setting, indicating that the base unit for kilobyte is "Ki" and the factor is 2^{10}.

5

[Base]

This configuration block contains settings related to base units.

ampere   (page 2737)
> Sets the base unit for ampere.

bit   (page 2732)
> Sets the base unit for bit.

byte   (page 2733)
> Sets the base unit for byte.

candela   (page 2734)
> Sets the base unit for candela.

gram   (page 2739)
> Sets the base unit for grams.

kelvin   (page 2736)
> Sets the base unit for kelvin.

meter   (page 2740)
> Sets the base unit for meter.

mole   (page 2735)
> Sets the base unit for mole.

second   (page 2738)
> Sets the base unit for seconds.

5

bit

Summary

Sets the base unit for bit.

Usage

```
bit=<base_unit>
```

Description

This setting sets the base unit to use for bit. By default this is set to "b".
This setting is case sensitive.

Examples

```
bit=b
```

This is the default setting.

5

byte

Summary

Sets the base unit for byte.

Usage

```
byte=<base_unit>
```

Description

This setting sets the base unit to use for byte. By default this is set to "B".
This setting is case sensitive.

Examples

```
byte=B
```

This is the default setting.

5

candela

Summary

Sets the base unit for candela.

Usage

```
candela=<base_unit>
```

Description

This setting sets the base unit to use for candela. By default this is set to "cd".
This setting is case sensitive.

Examples

```
candela=cd
```

This is the default setting.

5

mole

Summary

Sets the base unit for mole.

Usage

```
mole=<base_unit>
```

Description

This setting sets the base unit to use for mole. By default this is set to "mol".
This setting is case sensitive.

Examples

```
mole=mol
```

This is the default setting.

5

kelvin

Summary

Sets the base unit for kelvin.

Usage

```
kelvin=<base_unit>
```

Description

This setting sets the base unit to use for kelvin. By default this is set to "K".
This setting is case sensitive.

Examples

```
kelvin=K
```

This is the default setting.

5

ampere

Summary

Sets the base unit for ampere.

Usage

```
ampere=<base_unit>
```

Description

This setting sets the base unit to use for ampere. By default this is set to "A".
This setting is case sensitive.

Examples

```
ampere=A
```

This is the default setting.

5

second

Summary

Sets the base unit for seconds.

Usage

```
second=<base_unit>
```

Description

This setting sets the base unit to use for seconds. By default this is set to "s".
This setting is case sensitive.

Examples

```
second=s
```

This is the default setting.

5

gram

Summary

Sets the base unit for grams.

Usage

```
gram=<base_unit>
```

Description

This setting sets the base unit to use for gram. By default this is set to "g".
This setting is case sensitive.

Examples

```
gram=g
```

This is the default setting.

5

meter

Summary

Sets the base unit for meter.

Usage

```
meter=<base_unit>
```

Description

This setting sets the base unit to use for meter. By default this is set to "m".
This setting is case sensitive.

Examples

```
meter=m
```

This is the default setting.

5

5.14.44 upload.ini

The configuration blocks are documented in the following sections:

\begin{itemize} \item{} [CreateSettings]\refer{label221bca6559839efee47bb6eed0ccbb0b} \item{} [ExampleAction]\refer{labelf5001c99967e048f1befb9fedb4dfa29} \item{} [LocationSettings]\refer{labelcd0711faa1f01b07d255b19727cc4859} \item{} [UploadSettings]\refer{label175e2d8eb43d3ddb1a327cb077b10d40} \item{} [\{class_ identifier\}_ClassSettings]\refer{labeld0469e762fea2cc6589d7265b296706d} \end{itemize}

5

[ExampleAction]

NavigationPartIdentifier

Summary

Sets the navigation part to use for the page.

Usage

```
NavigationPartIdentifier=<identifier>
```

Description

This setting defines the navigation part to use for the page.

This setting is case sensitive and optional.

Examples

```
[BugFileUpload]
NavigationPartIdentifier=ezsetupnavigationpart
```

The configuration block refers to the action used when uploading a bug file.

ReturnType

Summary

Sets the type of the return values.

Usage

```
ReturnType=NodeID | ObjectID
```

Description

This setting defines the type of the return values.Possible values are:

- NodeID: this value will return the node IDs of the selected items
- ObjectID: this value will return the object IDs of the selected items

This setting is case sensitive and optional.

Examples

```
[RelatedObjectUpload]
ReturnType=ObjectID
```

The configuration block refers to the action used when uploading a new file as related object. The return type in this case will be the Object ID.

5

ParentNodes

Summary

Sets the nodes where the object should be placed.

Usage

```
ParentNodes[]=<node ID> | <node alias>
```

Description

This setting defines the nodes where the object should be placed. Each entry is either an alias or the node ID.The aliases that can be used as value must be defined with the setting [Up-loadSettings] AliasList or be one of the system defined aliases. The system defined aliases are by default:

- content: this alias represents the content root node as defined with [NodeSettings] RootN-ode in content.ini. The Node ID is usually 2

- users: this alias represents the user root node as defined with [NodeSettings] UserRootN-ode in content.ini. The Node ID is usually 5

- media: this alias represents the user root node.

- setup: this alias represents the user root node.

This setting is case sensitive and optional.

Examples

```
ParentNodes[]
ParentNodes[]=media
```

5

This is the default setting

[{class_identifier}_ClassSettings]

FileAttribute

Summary

Sets the class attribute identifier which will be used to get the file data.

Usage

```
FileAttribute=<class_attribute_identifier>
```

Description

This setting allows you to set the class attribute identifier which will be used to get the file data.
This setting is case sensitive.

Examples

```
[image_ClassSettings]
FileAttribute=image
NameAttribute=name
```

For the image class the file attribute is image.

5

NameAttribute

Summary

Sets the class attribute identifier which will get the name.

Usage

```
NameAttribute=<class attribute identifier>
```

Description

This setting will set the class attribute identifier which will get the name, which is calculated by using the setting NamePattern.This setting is case sensitive.

Examples

```
[image_ClassSettings]
NameAttribute=name
```

For the image class the name attribute is name.

5

NamePattern

Summary

Sets the pattern that will be used to define the name of the object.

Usage

```
NamePattern=<tag>
```

Description

This setting sets the pattern that will be used to define the name of the object. The following tags are available:

- <original_filename>: The same filename of the uploaded file is used as it was named on the locale machine

- <original_filename_base>: Similar to original_filename. Again the same filename of the uploaded file is used as on the locale machine, but only the base of the name is used and, i.e, no suffix.

- <original_filename_suffix>: Similar to original_filename. Again the same filename of the uploaded file is used as on the locale machine, but without the suffix

- <mime_type>: The detected MIME type of the file is used.

This setting is case sensitive.

Examples

```
[flash_ClassSettings]
FileAttribute=file
NameAttribute=name
NamePattern=<original_filename_base>
```

For the flash class the name will be the same as the original file name but without suffixes.

5

[CreateSettings]

5

MimeClassMap

Summary

Maps MIME types or groups to class identifiers.

Usage

```
MimeClassMap[<MIME type>]=<class identifier>
MimeClassMap[<MIME group>]=<class identifier>
```

Description

This setting maps either MIME types or MIME groups to class identifiers. The key can be either a MIME type, for example [image/jpeg], or a MIME group, for example [image]. By using only the group, the entire MIME group will be mapped to the class identifier.

This setting is case sensitive.

Examples

```
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
```

These are the default settings. It illustrates how to map different MIME types of the same MIME group to different class identifiers. For example, video/quicktime is mapped to quicktime, video/x-msvideo is mapped to windows_media and video/vnd.rn-realvideo is mapped to real_video.

5

MimeUploadHandlerMap

Summary

This setting maps MIME type to the name of an upload handler. When the handler is found for a given MIME type, it will be used instead of the normal attribute based upload.

Usage

```
MimeUploadHandlerMap[<MIME_type>]=<handler_identifier>MimeUploadHandlerMap[<MIME_
group>]=<handler_identifier>
```

Description

This setting maps MIME type to the name of an upload handler. When the handler is found for a given MIME type, it will be used instead of the normal attribute based upload.The <handler_identifier> will be used for both the class name and the filename of the PHP file. The system will look for the file named <handler_identifier>.php.The key can either be the full MIME type, for example [image/jpeg], or just the MIME group, like [image]. By using only the group the handler can take care of the entire set of MIME types.

By default this setting is left empty or commented out. This setting is case sensitive.

Examples

```
MimeUploadHandlerMap[application/vnd.sun.xml.writer]=ezopenofficeupload
```

The above example setting can be used for of OpenOffice.org handling. The system will look for ezopenofficeupload.php.

5

DefaultClass

Summary

Sets the class to use for all files that are not recognized.

Usage

```
DefaultClass=<class_name>
```

Description

This setting set a class to use for all files that are not recognized. By default this setting is set to the file class.

This setting is case sensitive.

Examples

```
DefaultClass=file
```

This is the default setting. The file class will be used for all files that are not recognized.

5

[LocationSettings]

ClassList

Summary

Sets the classes in the drop down list.

Usage

```
ClassList[]=<class ID>
```

Description

This setting lists the class identifiers that should be included in the location drop down list.

This setting is case sensitive.

Examples

```
ClassList[]
ClassList[]=folder
```

This is the default setting

5

RootNode

Summary

Sets the root node from which to fetch upload locations.

Usage

```
RootNode=<node_ID> | <node_path>
```

Description

This setting defines the root node from which to fetch upload locations. As possible value either the node ID can be used or a node path. The following node paths are possible:

- content: The content structure
- users: The user tree
- media: the media tree for images, files and media

This setting is case sensitive.

Examples

```
RootNode=media
```

This the default setting

5

MaxDepth

Summary

Sets the maximum number of levels in which to look for objects to upload.

Usage

```
MaxDepth=<integer>
```

Description

This setting sets the maximum number of levels in which to look for objects to upload.Defining a depth of 1 means only direct descendants will be checked.

Examples

```
MaxDepth=3
```

This the default setting

5

MaxItems

Summary

Sets the maximum number of elements in the drop down list.

Usage

```
MaxItems=<integer>
```

Description

This setting sets the maximum number of elements in the drop down list. Keep in mind that setting to a high value might slow down the system and make the dropdown hard to use.Using a number between 10 and 30 is best.

Examples

```
MaxItems=30
```

This the default setting

5

[UploadSettings]

DefaultReturnType

Summary

Sets the default return type

Usage

DefaultReturnType=<**type**>

Description

Sets the default type to return to after uploading. Possible values are "NodeID" and "ObjectID".

This setting is case sensitive.

Examples

DefaultReturnType=NodeID

This is the default setting

5

AliasList

Summary

Sets node aliases.

Usage

```
AliasList[<alias name>]=<node_ID>
```

Description

This setting maps node aliases to node IDs and makes it possible to define custom alias names.For start nodes and top level nodes it is possible to use text aliases. Besides the aliases defined with this setting, the following aliases are defined by default:

- content: this alias represents the content root node as defined with [NodeSettings] RootNode in content.ini. The Node ID is usually 2.

- users: this alias represents the user root node as defined with [NodeSettings] UserRootNode in content.ini. The Node ID is usually 5.

This setting is case sensitive and left empty by default.

5

5.14.45   viewcache.ini

The configuration blocks are documented in the following sections:

- [class_identifier] (page 2767)
- [ViewCacheSettings] (page 2763)

5

[ViewCacheSettings]

ClearRelationTypes   (page 2765)
    Controls whether the view caches for objects related in various ways should be cleared or not.

SmartCacheClear   (page 2764)
    Sets if the smart viewcache cleaning system is turned on/off.

5

SmartCacheClear

Summary

Sets if the smart viewcache cleaning system is turned on/off.

Usage

```
SmartCacheClear=enabled|disabled
```

Description

This setting makes it possible to activate the smart viewcache cleaning system (page 456) (disabled by default).

5

ClearRelationTypes

Summary

Controls whether the view caches for objects related in various ways should be cleared or not.

Usage

```
ClearRelationTypes[]=type1
ClearRelationTypes[]=type2
...
```

Description

This directive sets which related and reverse related objects the view caches should be cleared for as a response to the view cache being cleared for a specific object. More precisely, it can be used to control whether the view caches for objects that are related in various ways (common, linked, embedded and so on) should be cleared or not. This setting is an array of strings where only eight pre-defined values can be used (refer to the following table).

Type	Description
common	Clear cache for related objects that have relations of the "common" type.
reverse_common	Clear cache for reverse related objects that have relations of the "common" type.
linked	Clear cache for related objects that have relations of the "XML linked" type.
reverse_linked	Clear cache for reverse related objects that have relations of the "XML linked" type.
embedded	Clear cache for related objects that have relations of the "XML embedded" type.
reverse_embedded	Clear cache for reverse related objects that have relations of the "XML embedded" type.
attribute	Clear cache for related objects that have relations at the attribute level.
reverse_attribute	Clear cache for reverse related objects that have relations at the attribute level.

Note that if you need to set up additional/custom rules that control which nodes the view cache should be cleared for when the view cache for a specific object is being cleared, you'll have to use smart viewcache cleaning (page 456).

Examples

Example 1

```
[ViewCacheSettings]
ClearRelationTypes[]
SmartCacheClear=disabled
```

When clearing the view cache for an object, the system will automatically clear the view cache for the following nodes:

- All published nodes of this object and their parent nodes

- Nodes of the objects that have the same keyword (if any)

Since the "ClearRelationTypes" configuration array is empty, the view caches for related and reverse related objects will not be cleared.

Example 2

```
[ViewCacheSettings]
ClearRelationTypes[]
ClearRelationTypes[]=common
ClearRelationTypes[]=reverse_common
ClearRelationTypes[]=reverse_embedded
SmartCacheClear=disabled
```

When clearing the view cache for an object, the system will automatically clear the view cache for the following nodes:

- All published nodes of this object and their parent nodes

- Nodes of the objects that have the same keyword (if any)

- 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

Note that the first two items on the list are not controlled by the "ClearRelationTypes" setting.

5

[class_identifier]

\begin{description} \item[AdditionalObjectIDs]\refer{label30d2d281eefca275ed749c1b2784d0f2}\ \\ Sets which arbitrary objects that the view caches will be cleared for when an object of the "class_identifier" class is changed.

\item[ClearCacheBlocks]\refer{label321f1994462697325951c20a9b55176c}\ \\ This is a new optional setting for clearing cache blocks available in version 4.3 and upwards of eZ Publish

\item[ClearCacheMethod]\refer{label4e7d83dd22a19918b479f470a959c1d7}\ \\ Sets which method(s) to use when clearing the view caches for additional (parent) nodes.

\item[DependentClassIdentifier]\refer{labelf1f216aefa588fec758b80ee31f544d6}\ \\ Sets which content classes that will be considered as "dependent classes" when an object of the "class_identifier" class is changed.

\item[MaxParents]\refer{label45ba2e142cf9d6e975e1f91aaab52fb9}\ \\ Limits clearing the view caches for additional nodes to a specified number of levels.

\item[ObjectFilter]\refer{labelbae6ee7c38912593f61dbce081e2e5ab}\ \\ Limits clearing the view caches for additional nodes to a specified list of object IDs.

\end{description}

5

ClearCacheBlocks

Summary

This is a new optional setting for clearing cache blocks available in version 4.3 and upwards of eZ Publish

Usage

```
ClearCacheBlocks=disabled
```

Description

All cache blocks without ignore_content_expiry are cleared on publish, with this you can disable it for selected classes.

The default setting is:

#ClearCacheBlocks=enabled

5

AdditionalObjectIDs

Summary

Sets which arbitrary objects that the view caches will be cleared for when an object of the "class_identifier" class is changed.

Usage

```
AdditionalObjectIDs[]=object_id1
AdditionalObjectIDs[]=object_id2
...
```

Description

If smart viewcache cleaning is enabled, this directive can be used to make the system clear the view caches for a set of arbitrary objects when an object of the "class_identifier" class is changed. If you specify a list of object ID numbers using the "AdditionalObjectIDs[]" configuration array, the view cache will be cleared for all the locations (nodes) of these objects. You can find more information and examples in the "Smart view cache cleaning (page 456)" section.

5

DependentClassIdentifier

Summary

Sets which content classes that will be considered as "dependent classes" when an object of the "class_identifier" class is changed.

Usage

```
DependentClassIdentifier[]=class_identifier1
DependentClassIdentifier[]=class_identifier2
...
```

Description

If smart viewcache cleaning is enabled, this directive can be used to specify the list of dependent classes. When a published object of the "class_identifier" class is changed, svcs will check the parent nodes that are listed in the "path_string" attribute for all the object's nodes/locations and clear the view cache for each parent node that encapsulates an object of a dependent class. You can find more information and examples in the "Smart view cache cleaning (page 456)" section.

5

ClearCacheMethod

Summary

Sets which method(s) to use when clearing the view caches for additional (parent) nodes.

Usage

```
ClearCacheMethod[]=method1
 ClearCacheMethod[]=method2
 ...
```

Description

If smart viewcache cleaning is enabled, this directive sets which method(s) that will be used when clearing the view caches for additional (parent) nodes. This setting is an array of strings where only the following pre-defined values can be used: "object", "parent", "relating", "keyword", "siblings", "all", "children". These methods are described in the "Smart view cache cleaning (page 456)" section.

Note: The "children" method for clearing view cache was made available as of version 4.5.

5

ObjectFilter

Summary

Limits clearing the view caches for additional nodes to a specified list of object IDs.

Usage

```
ObjectFilter[]=object_id1
ObjectFilter[]=object_id2
...
```

Description

If smart viewcache cleaning is enabled, this directive can be used to set a limitation for clearing the view caches of additional nodes (svcs will only clear the view caches of those additional nodes that encapsulate the objects listed in the "ObjectFilter[]" configuration array). You can find more information and examples in the "Smart view cache cleaning (page 456)" section.

5

MaxParents

Summary

Limits clearing the view caches for additional nodes to a specified number of levels.

Usage

```
MaxParents=number
```

Description

If smart viewcache cleaning is enabled, this directive can be used to set a limitation for clearing the view caches of additional nodes. When a published object of the "class_identifier" class is changed, svcs will scan the parent nodes that are listed in the "path_string" attribute for all the object's nodes/locations and clear the view cache for each parent node that encapsulates an object of a dependent class. The maximal quantity of nodes that will be scanned is controlled by the "MaxParents" setting.

You can find more information and examples in the "Smart view cache cleaning (page 456)" section.

5

5.14.46 webdav.ini

The configuration blocks are documented in the following sections:

\begin{itemize} \item{} [DisplaySettings]\refer{label895bb83033ec93b26a8f20232a307e86}
\item{} [FolderSettings]\refer{label5cd2f35dcd8cf6c17cada80ed3872c44}
\item{} [GeneralSettings]\refer{label4c04924711319174c08f91459f502341}
\item{} [GetSettings]\refer{labelf501ef582b6d5dd4f4bc4d38bcf79b67} \item{}
[PutSettings]\refer{label2cec346bf18a079e889e96d03a85df5a} \item{}
[RenameSettings]\refer{labelc4ceb85ac2119f50269b41cacdc07070} \end{itemize}

5

[GeneralSettings]

5

EnableWebDAV

Summary

Sets WebDAV functionality.

Usage

```
EnableWebDAV=true | false
```

Description

This setting allows you to enable or disable WebDAV support. WebDAV is an abbreviation for "Web-based Distributed Authoring and Versioning" and 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.

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.

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 file system (as directories and files). When enabling WebDAV additional settings must be set in site.ini and upload.ini. For more information on WebDAV please visit http://ez.no/doc/ez_publish/ technical_manual/4_4/features/webdav

By default this function is disabled ("false").

This setting is case sensitive

Examples

```
EnableWebDAV=false
```

This is the default setting.

Logging

Summary

Enables WebDAV logging

Usage

```
Logging=enabled | disabled
```

Description

This setting allows you to enable or disable WebDAV Logging. This setting must be set to enabled to use the WebDAV functionality.

This setting is case sensitive.

Examples

```
Logging=disabled
```

5

StartNode

Summary

Sets the node to start from when using WebDAV.

Usage

```
StartNode=<node_ID>
```

Description

This setting is available since eZ Publish 4.1 and it defines the custom start node for WebDAV operations. Sets the node to start from when using WebDAV. For each siteaccess an override file webdav.ini.append.php should be created which specifies the start node of that specific siteaccess.

This setting is case sensitive.

The value is the Node ID number. By default this is set to 2 (this is the Node ID of /site/content).

Examples

```
StartNode=2
```

For this siteaccess WebDAV should start from Node ID 2, which by default corresponds to /site/content.

5

FolderClasses

Summary

Sets the classes that need to be seen as a folder.

Usage

```
FolderClasses[]=<classname>
```

Description

This setting list the classes that should always be seen as a folder by WebDAV. If not specified here, a class containing an image attribute will be seen as an image by default.

This setting is case sensitive.

Examples

```
FolderClasses[]=folder
```

This is the default setting.

5

[DisplaySettings]

5

FileAttribute

Summary

Deprecated

Usage

```
FileAttribute[]=<attribute>
```

Description

Deprecated

5

[GetSettings]

FileAttribute

Summary

Deprecated

Usage

```
FileAttribute[]=<attribute>
```

Description

Deprecated

5

[PutSettings]

Mime

Summary

Deprecated

Usage

```
MIME[<MIME_type>]=<class>
```

Description

DEPRECATED

Setting per mimetype to select which class to create.

This setting is case sensitive.

5

DefaultClass

Summary

Deprecated

Usage

```
DefaultClass=<class>
```

Description

DEPRECATED

5

[FolderSettings]

FolderClass

Summary

Deprecated

Usage

```
FolderClass=<folder_class_ID>
```

Description

DEPRECATED

This setting defines the folder class ID, for example "1".

NameAttribute

Summary

Deprecated

Usage

```
NameAttribute=<attribute>
```

Description

DEPRECATED

This setting is used to determine which attribute to use when creating, naming or renaming a folder, for example "name".

5

[RenameSettings]

NameAttribute

Summary

Sets the attribute to use when renaming.

Usage

```
NameAttribute[<node_ID>]=<attribute>
```

Description

This setting is used to determine which attribute to use when renaming. The attribute is mapped to a class ID to allow for specific attribute settings.This settings in commented out by default by adding "#".This setting is case sensitive.

Examples

```
#NameAttribute[5]=name
#NameAttribute[2]=title
#NameAttribute[12]=filename
```

These are the default settings, but they are commented out by default.

5

5.14.47   wordtoimage.ini

The configuration blocks are documented in the following sections:

\begin{itemize} \item{} [WordToImageSetting]\refer{label1d24aee8d2c40e6b08276ffcb6595d13}
\end{itemize}

5

[WordToImageSetting]

IconRoot

Summary

This setting defines the location of the icons folder.

Usage

```
IconRoot=/share/icons
```

Description

This setting defines the location of the icons folder. The icons used to replace text and character combination need to be stored here.

This setting is case sensitive.

Examples

```
<span class="k" style="margin-left: 0px; margin-right: 0px;
"><span class="na" style="margin-left: 0px; margin-right: 0px;">IconRoot</
span><span class="o" style="margin-left: 0px; margin-right: 0px;">=</
span><span class="s" style="margin-left: 0px; margin-right: 0px;">/share/
icons</span></span>
```

5

ReplaceText

Summary

Sets which text that needs to be converted to an icon.

Usage

```
ReplaceText[]=:)
```

Description

This setting shows the text or character combination that should be replaced by an image.
The image to convert to is defined directly below ReplaceText with the setting ReplaceIcon in
[WordToImageSettings].

So whenever a special character sequences is published, it will be replaced by the image
icon defined in ReplaceIcon. For example, the sequence ":-)" will be replaced by a small image
of a smiling face.

This setting is case sensitive.

Examples

```
<span class="k" style="margin-left: 0px; margin-right: 0px;
"><span class="na" style="margin-left: 0px; margin-right: 0px;
"><span class="na" style="margin-left: 0px; margin-right: 0px;">IconRoot</
span><span class="o" style="margin-left: 0px; margin-right: 0px;">=</
span><span class="s" style="margin-left: 0px; margin-right: 0px;">/share/
icons</span></span></span>
<span class="na" style="margin-left: 0px; margin-right: 0px;">ReplaceText[]</
span><span class="o" style="margin-left: 0px; margin-right: 0px;">=</
span><span class="s" style="margin-left: 0px; margin-right: 0px;">:D</span>
<span class="na" style="margin-left: 0px; margin-right: 0px;">ReplaceIcon[]</
span><span class="o" style="margin-left: 0px; margin-right: 0px;">=</
span><span class="s" style="margin-left: 0px; margin-right: 0px;
">big-smile.gif</span>
<span class="na" style="margin-left: 0px; margin-right: 0px;">ReplaceText[]</
span><span class="o" style="margin-left: 0px; margin-right: 0px;">=</
span><span class="s" style="margin-left: 0px; margin-right: 0px;">:-D</span>
<span class="na" style="margin-left: 0px; margin-right: 0px;">ReplaceIcon[]</
span><span class="o" style="margin-left: 0px; margin-right: 0px;">=</
span><span class="s" style="margin-left: 0px; margin-right: 0px;
">big-smile.gi</span>f
```

5

This is a default setting, when ":D" or ":-D" is typed, the system will look for the image big-smile.gif, a smiley with a big smile, in the folder (root of your eZ Publish installation)/share/icons to replace it.

5

ReplaceIcon

Summary

Sets which icons that should be converted to text.

Usage

```
ReplaceIcon[]=happy.gif
```

Description

When a character combination is published that is defined in the setting ReplaceText in [WordToImageSettings] it will need to be replaced by an image.   This setting sets the image file to convert to when the specific character combination is used which is set by ReplaceText, straight above this setting. The image files used as value should be stored in the folder defined in IconRoot in [WordToImageSettings].

So whenever a special character sequences is published, it will be replaced by the image icon defined in ReplaceIcon. For example, the sequence ":-)" will be replaced by a small image of a smiling face.

This setting is case sensitive.

Examples

```
IconRoot=/share/icons
ReplaceText[]=;)
ReplaceIcon[]=blunk.gif
ReplaceText[]=;-)
ReplaceIcon[]=blunk.gif
```

This is a default setting, when ";)" or ";-)" is typed, the system will look for the image blunk.gif, a winking smiley, in the folder (root of your eZ Publish installation)/share/icons to replace it.

5

5.14.48   workflow.ini

You can find an overview over the workflow system in the workflows (page 167) concept chapter.
The configuration blocks are documented in the following sections:

- [EventSettings] (page 2804)
- [OperationSettings] (page 2802)
- [SimpleShippingWorkflow] (page 2799)

5

[SimpleShippingWorkflow]

ShippingCost   (page 2801)
    Sets the cost of the shipping.

ShippingDescription   (page 2800)
    Sets the description that will appear on the order when using the simle shipping workflow.

5

ShippingDescription

Summary

Sets the description that will appear on the order when using the simle shipping workflow.

Usage

```
ShippingDescription= description
```

ShippingCost

Summary

Sets the cost of the shipping.

Usage

```
ShippingCost= number
```

Description

Adds the specified amount to orders if the workflow is enabled.. The currency is the same as the currency of the order.

Note: This workflow does not work with multilangauge sites.

5

[OperationSettings]

AvailableOperations   (page)
    Sets the triggers that are avaible from the administration interface.

5

AvailableOperations

Summary

Sets the triggers that are avaible from the administration interface.

Usage

```
AvailableOperations=[before|after_] module1_trigger1[;[before|after_] module2_
trigger2[;...]
```

Description

The availableOperations specifies the triggers that are available in the system. Triggers are specified per module and are given a trigger name. Additionally, each trigger point can have both a before and after trigger. Each section of AvailableOperations specifies one module and a trigger name. Additionally you can specify if the trigger is a before trigger by adding before or after in front of the module name. If the trigger is both a before and an after trigger, simply don't specify anything.

eZ Publish will look for the file operation_definition.php in the module directory. Each trigger activated in this setting must be properly set up in the operation_definition.php of the corresponding module.

Examples

```
AvailableOperations=content_publish;before_shop_confirmorder;shop_checkout
```

These settings will enable and make eZ Publish search for the following triggers:

Module	Trigger name	Trigger time
content	publish	before
content	publish	after
shop	confirmorder	before
shop	checkout	before
shop	checkout	after

5

[EventSettings]

AvailableEventTypes   (page 2805)
    Sets the name of the event types that eZ publish should search for.

ExtensionDirectories   (page 2806)
    Sets the extensions that contain event types.

RepositoryDirectories   (page 2807)
    Sets the directories where eZ publish will search for event types.

5

AvailableEventTypes

Summary

Sets the name of the event types that eZ publish should search for.

Usage

```
AvailableEventTypes[]= event_group1_ event_name1
AvailableEventTypes[]= event_group2_ event_name2
...
```

Description

This setting contains a list of event types. These event types consist of an event group and an event name. These two items are separated with an underscore (_).

The combination of the settings ExtensionDirectories (page 2806), and AvailableEventTypes specify where eZ publish will look for additional event types.
eZ publish will search for the extensions in "/extension/ *extension_name*/eventtypes/ *event_group*/ *event_name*type.php" where *extension_name* is specified by ExtensionDirectories (page 2806) and *event_group* and *event_name* are specified by AvailableEventTypes.

Examples

```
ExtensionDirectories[]=ezpaynetdirect
AvailableEventTypes[]=event_ezapprove
AvailableEventTypes[]=paynet_ezpaynetdirect
```

These settings will make eZ publish search for the following files for the events.
extensions/ezpaynetdirect/eventtypes/event/ezapprove/ezapprove.php
extensions/ezpaynetdirect/eventtypes/paynet/ezpaynetdirect/ezpaynetdirecttype.php

5

ExtensionDirectories

Summary

Sets the extensions that contain event types.

Usage

```
ExtensionDirectories[]= extension1
ExtensionDirectories[]= extension2
...
```

Description

eZ publish will automatically search the extensions specified by ExtensionDirectories for event types. By default eZ publish will search inside the "eventtypes" directory inside your extension. The exact location of the events in this directory is specified with the AvailableEventTypes (page 2805) setting.

Most frequently ExtensionDirectories is specified in the workflow.ini.append inside custom extensions to indicate that the extensions has custom event types. This makes eZ publish recognize the events as soon as the extension is enabled.

Examples

```
ExtensionDirectories[]=ezpaynetdirect
```

The ezpaynetdirect extension uses a workflow which is located in: "/extension/ezpaynetdirect/eventtypes/ *AvailabeEventTypes*"
This setting makes eZ publish search for event types in "/extension/ezpaynetdirect/eventtypes/ *AvailabeEventTypes*".

The *AvailableEventTypes* are described in the AvailableEventTypes (page 2805) setting.

5

RepositoryDirectories

Summary

Sets the directories where eZ publish will search for event types.

Usage

```
RepositoryDirectories[]= directory1
RepositoryDirectories[]= directory2
...
```

Description

The RepositoryDirectories specifies a list of directories where built in workflow event types can be found. The exact location of the workflow in this directory is specified with the AvailableEvent-Types (page 2805) setting.

If you have an extension with a workflow, then use the ExtensionDirectories (page 2806) setting to register the event types.

Note: Don't change this setting unless you know what you are doing.

5

5.15 Libraries

\begin{description} \item[ezdb]\refer{labele61624e175dae794ca26a94cebe6c0cd}\ \\ Provides a database abstraction layer.

\item[ezdbschema]\refer{label6147aff2990fd374860b5cb7c0d14a3f}\ \\ Provides a cross database schema checker and update tool.

\item[ezfile]\refer{label01669b85d9f39b8836eea03785e8c5fa}\ \\ Provides cross platform file and compression utilities.

\item[ezi18n]\refer{labelb4fe8b5cf071dcb6647bc120658de1dd}\ \\ Provides functionality for reading ".ts" files and for translating text in templates.

\item[ezimage]\refer{labelbc6d3b4d52ebc440a35b1a074ef67392}\ \\ Provides an abstracted image manipulation interface.

\item[ezlocale]\refer{labelb64ed55fe5f895bcd7f3c4ffaa72b92a}\ \\ Provides functionality for localizing dates, currencies, etc.

\item[ezpdf]\refer{labelbfc1c4afb1d7e9de022acb5f0843c3da}\ \\ Provides a solution for using the template system to generate PDFs (DEPRECATED).

\item[ezsoap]\refer{label1a89623c36ea1f2dae33447619dddd87}\ \\ Provides a low level interface for the SOAP protocol.

\item[eztemplate]\refer{label769ff5d77215d50fd8646e5fb9f57469}\ \\ Provides the template interpreter, compiler and the basic functions and operators.

\item[ezutils]\refer{labelfc0280a67ed2f3090fad58fbc1df3608}\ \\ Provides small utilities (ini file parser, MIME type handlers, E-mail handling, etc.).

\item[ezwebdav]\refer{label3358f6a564b5855dfcbc94b5399199b7}\ \\ Provides a low level communication interface for the WebDAV protocol.

\item[ezxml]\refer{labela828f414f7cf82f2baab15ba3a460c86}\ \\ Provides low level XML DOM parsing and construction utilities (DEPRECATED).

\end{description}

5

5.15.1   ezdb

Provides a database abstraction layer.

5

5.15.2   ezdbschema

Provides a cross database schema checker and update tool.

5

5.15.3   ezfile

Provides cross platform file and compression utilities.

5.15.4   ezi18n

Provides functionality for reading ".ts" files and for translating text in templates.

5

5.15.5 ezimage

Provides an abstracted image manipulation interface.

5

5.15.6 ezlocale

Provides functionality for localizing dates, currencies, etc.

5

5.15.7   ezpdf

Provides a solution for using the template system to generate PDFs (DEPRECATED).

5

5.15.8   ezsoap

Provides a low level interface for the SOAP protocol.

5

5.15.9 eztemplate

Provides the template interpreter, compiler and the basic functions and operators.

5

5.15.10   ezutils

Provides small utilities (ini file parser, MIME type handlers, E-mail handling, etc.).

5

5.15.11   ezwebdav

Provides a low level communication interface for the WebDAV protocol.

5

5.15.12   ezxml

In previous versions of eZ Publish, this library provided low level XML DOM parsing and construction utilities. It has been deprecated in eZ Publish 4. Instead, the system makes use of the DOM functions in PHP 5 (refer to "Requirements for doing a normal installation " for more information).

5

5.16   XML tags

The "XML block" (page 601) datatype supports the following tags / elements:

- Headings (page 603)
- Bold text (page 604)
- Italic text (page 605)
- Unformatted text (page 606)
- Lists (page 607)
- Tables (page 608)
- Hyperlinks (page 610)
- Anchors (page 612)
- Object embedding (page 613)
- Custom tags (page 616)
- Paragraphs (page 617)

5

5.17   Scripts

eZ Publish comes with a collection of scripts that cover the needs of various typical tasks. The scripts are grouped based on their type and purpose. The following table contains an overview of available script groups.

5.17.1   Generic

Generic Generic scripts for manual maintenance. ./bin/php

The "./bin/php" directory contains generic PHP scripts that are mainly used for manual mainte-
nance (clearing the caches, updating the search index, copying and removing large subtrees and
so on). In order to be able to run these scripts, you need to have PHP CLI installed. The scripts
must be run from the root directory of your eZ Publish installation.

5

Parameters

This page provides an overview of generic parameters that are used by many of the PHP scripts located in the "./bin/php" directory.

help

help -h, --help

Output information about the script and supported parameters.

5

siteaccess

siteaccess -s <siteaccess>, --siteaccess=<siteaccess>

The siteaccess to run the script for. If omitted, the default siteaccess (page 2527) will be used.

5

quiet

quiet -q, --quiet

Run the script in "quiet" mode. No output will be generated.

5

debug

debug -d[<option>[,<option2>...]], --debug[=<option>[,<option2>...]]

Output debug information. The following options can be used to control what the debug output should include: "all" (everything), "accumulator" (accumulators), "include" (included files), "timing" (timing points), "error" (errors), "warning" (warnings), "notice" (notices).

5

colors

colors -c, --colors

Use ANSI colors when generating debug output (default).

5

no-colors

no-colors --no-colors

Do not use ANSI colors when generating debug output.

5

login

login -l <username>, --login=<username>

The username of the account that the script should run as (for example "Administrator"). Note that a corresponding password should be specified using the "password (page 2832)" parameter.

5

password

password -p <password>, --password=<password>

The password that belongs to the user account which is specified using the "login (page 2831)" parameter.

5

logfiles

logfiles --logfiles

Generate log files. This parameter must be used together with the debug parameter (”-d” or ”--debug”). The log files (for example ”warning.log”) will be stored in the ”var/log” directory of the eZ Publish installation.

5

no-logfiles

no-logfiles --no-logfiles

Do not generate log files (default).

5

verbose

verbose -v, --verbose

Output verbose / additional information.

ezcache.php

ezcache.php Clears the eZ Publish caches. parent

This script makes it possible to clear various eZ Publish caches - that is, either expire the specified caches (default) or physically remove the cache files (requires the "--purge" parameter).

To run the script, navigate into the root directory of your eZ Publish installation and use the following shell command:

```
php bin/php/ezcache.php [<parameters>]
```

Replace "[<parameters>]" with one or more of the script parameters described above. Note that you need to have PHP CLI installed in order to run the script from within a system shell.

Sometimes the script is unable to clear the caches because of restrictive file/directory permission settings. You can check whether the caches have been cleared by inspecting the contents of the various cache subdirectories within the "var" directory (typically "var/cache/" and "var/<name_ of_siteaccess>/cache/" directories).

Selective cache cleanup

eZ Publish generates different types of caches. While the "--clear-all" parameter gets rid of everything, the "--clear-id" parameter makes it possible to remove only certain types of caches (configuration settings, compiled templates, URL aliases, image variations, etc.). The identifiers of the caches to be removed must be specified as a comma separated list.

A "cache tag" is assigned to every cache type. The cache tag reveals which group a particular cache belongs to (ini, image, template, content and so on). The "--clear-tag" parameter makes it possible to clear all caches that belong to a specified group. Note that a cache type (for example, the "Template block cache") can be assigned to several groups.

The following table shows the different caches, the corresponding identifiers and cache tags (the group they belong to).

5

Name	Identifier	Cache tags
INI cache	ini	ini
Global INI cache	global_ini	ini
Image alias cache	imagealias	image
Template cache	template	template
Template override cache	template-override	template
Template block cache	template-block	template, content
Content view cache	content	content
URL alias cache	urlalias	content
RSS cache	rss_cache	content
Class identifier cache	classid	content
Sort key cache	sortkey	content
Tree menu cache	content_tree_menu	content
User info cache	user_info_cache	user
Codepage cache	codepage	codepage
Character transformation cache	chartrans	i18n

Clustering compatibility

The script also works if the caches are stored in the database (typically happens when running eZ Publish in a cluster environment (page 293)). When the "--purge" parameter is specified, the cache entries are DELETEd from the tables. When it is not specified, the system will only mark the entries as expired either using UPDATE queries or using the expiry values in "expiry.php".

Example 1

```
php bin/php/ezcache.php --clear-all
```

Expire all caches.

Example 2

```
php bin/php/ezcache.php --clear-all --purge
```

Clear all caches (either physically remove all cache files or delete all cache entries from the database).

Example 3

```
php bin/php/ezcache.php --clear-all --purge -dwarning
```

Clear all caches and output warnings that occur during execution. For example, if the system is not able to locate one of the enabled extensions, the console output could look like this:

```
Purging : Content view cache, Global INI cache, INI cache, Codepage cache,
Class identifier cache, Sort key cache, URL alias cache, Character
transformation cache, Image alias, Template cache, Template block cache,
Template override cache, RSS cache, User info cache, Content tree menu
(browser cache)

################## DEBUG ##################
Warning: ()
Extension 'ezdhtml' does not exist, looked for directory 'extension/ezdhtml'
```

Example 4

```
php bin/php/ezcache.php --clear-id=user_info_cache
```

Expire all user info caches.

Example 5

```
php bin/php/ezcache.php --clear-tag=ini --purge
```

Clear both INI cache and global INI cache.

Example 6

```
php bin/php/ezcache.php --clear-all --iteration-sleep='5.5'
--iteration-max=100 --purge
```

Clear all caches. The purging will be done with the interval between the iterations equal to 5.5 seconds. During one iteration, a maximum number of 100 items will be removed.

Example 7

```
php bin/php/ezcache.php --expiry='-2 days'
```

Expire all caches that are older than two days.

5

Example 8

```
php bin/php/ezcache.php --expiry='last monday' --purge
```

Clear all caches generated before last Monday.

5

help

help -h, --help

Output information about the script and supported parameters.

5

siteaccess

siteaccess -s <siteaccess>, --siteaccess=<siteaccess>

The siteaccess to run the script for. If omitted, the default siteaccess (page 2527) will be used.

5

quiet

quiet -q, --quiet

Run the script in "quiet" mode. No output will be generated.

5

debug

debug -d[<option>[,<option2>...]], --debug[=<option>[,<option2>...]]

Output debug information. The following options can be used to control what the debug output should include: "all" (everything), "accumulator" (accumulators), "include" (included files), "timing" (timing points), "error" (errors), "warning" (warnings), "notice" (notices).

5

colors

colors -c, --colors

Use ANSI colors when generating debug output (default).

no-colors

no-colors --no-colors

Do not use ANSI colors when generating debug output.

5

logfiles

logfiles --logfiles

Generate log files. This parameter must be used together with the debug parameter ("-d" or "--debug"). The log files (for example "warning.log") will be stored in the "var/log" directory of the eZ Publish installation.

5

no-logfiles

no-logfiles --no-logfiles

Do not generate log files (default).

5

verbose

verbose -v, --verbose

Output verbose / additional information.

ezsubtreeremove.php

ezsubtreeremove.php Safely removes large subtrees. parent

This script makes it possible to remove subtrees. Typically, it is used to remove subtrees containing more nodes than the "MaxNodesRemoveSubtree (page 1760)" setting allows (these cannot be removed from within the administration interface).

By default, the script will remove content in the same way as it is done from within the administration interface when the "Move to trash" checkbox is selected. In other words, the removed objects will end up in the trash while the tree structure itself (the nodes) will be lost forever. The "--ignore-trash" parameter allows objects to be removed without being put in the trash first.

An object processed by this script will only be put in the trash (or removed when the "--ignore-trash" parameter is used) if it doesn't have other/additional locations than the ones that were specified using the "--nodes-id" parameter.

Note that the script will access the node tree as a certain user. This is controlled by the "UserCreatorID (page 2588)" setting. If the specified user is not allowed to remove a particular subtree, the script will output an error message.

Example 1

```
php bin/php/ezsubtreeremove.php --nodes-id=65
```

Remove node 65 and all its sub items. If the object encapsulated by node 65 does not have other locations, it will be moved to the trash.

Example 2

```
php bin/php/ezsubtreeremove.php --nodes-id=65 --ignore-trash
```

Remove node 65 and all its sub items. If the object encapsulated by node 65 does not have other locations, it will be permanently removed.

Example 3

```
php bin/php/ezsubtreeremove.php --nodes-id=60,65,78
```

Nodes 60, 65, 78 and all their sub items will be removed.

colors

colors -c, --colors

Use ANSI colors when generating debug output (default).

5

debug

debug -d[<option>[,<option2>...]], --debug[=<option>[,<option2>...]]

Output debug information. The following options can be used to control what the debug output should include: "all" (everything), "accumulator" (accumulators), "include" (included files), "timing" (timing points), "error" (errors), "warning" (warnings), "notice" (notices).

5

help

help -h, --help

Output information about the script and supported parameters.

ignore-trash

ignore-trash --ignore-trash

Permanently delete the specified subtrees. Without this parameter, the content will be moved to the trash (note that the tree structure will be lost forever).

5

logfiles

logfiles --logfiles

Generate log files. This parameter must be used together with the debug parameter ("-d" or "--debug"). The log files (for example "warning.log") will be stored in the "var/log" directory of the eZ Publish installation.

5

no-colors

no-colors --no-colors

Do not use ANSI colors when generating debug output.

5

no-logfiles

no-logfiles --no-logfiles

Do not generate log files (default).

5

nodes-id

nodes-id --nodes-id=<node_id>[,<node_id2>€]

The node ID of the subtree or subtrees that should be removed. In case multiple subtrees are to be removed, their node IDs must be provided as a comma separated list. Both the specified nodes and their sub items will be removed. This parameter is required.

5

quiet

quiet -q, --quiet

Run the script in "quiet" mode. No output will be generated.

siteaccess

siteaccess -s <siteaccess>, --siteaccess=<siteaccess>

The siteaccess to run the script for. If omitted, the default siteaccess (page 2527) will be used.

5

updateisbn13.php

updateisbn13.php Imports the latest ISBN ranges to the database. parent

This script makes it possible to import the latest ISBN ranges to the database. Without this information, the system may not be able to handle ISBN-13 codes properly (refer to the documentation of the ISBN datatype (page 543) for more information). Since the calculation rules for ISBN codes can be changed on a monthly basis, you may need to run the script regularly in order to keep the range information in the database up to date.

Before running the script, download the "ranges.js" file that contains information about the different ranges from http://www.isbn-international.org/converter/ranges.js and copy it to your eZ Publish installation. When running the script, use the "--file" parameter to tell the script where to look for the downloaded file. By default, the database configuration of the default siteaccess will be used. You can use the optional "--siteaccess" parameter to make the script import ranges to the database that is used by a particular siteaccess. In addition, the optional "--db-database", "--db-driver", "--db-host", "--db-user" and "--db-password" parameters make it possible to combine custom database options with the database configuration of the specified (or default) siteaccess.

Example 1

```
php bin/php/updateisbn13.php --file=var/ranges.js
```

Import data from the "ranges.js" file to the database that is used by the default siteaccess. The script will look for "ranges.js" in the "var" directory of the eZ Publish installation.

Example 2

```
php bin/php/updateisbn13.php --file=ranges.js --siteaccess=example
```

Import data from the "ranges.js" file to the database that is used by the "example" siteaccess. The script will look for "ranges.js" in the root directory of the eZ Publish installation.

Example 3

```
php bin/php/
updateisbn13.php --file=ranges.js --siteaccess=example --db-database=test
```

Import data from the "ranges.js" file located in the root directory of the eZ Publish installation. The script will use the database configuration of the "example" siteaccess but ignore the value of the "Database" directive and use the specified custom database name instead. If the "example" siteaccess makes use of a database called "foo" on localhost, the script will import ISBN ranges to the "test" database on localhost.

5

Example 4

```
php bin/php/updateisbn13.php --file=ranges.js --db-database=test db-user=root
db-password=secret
```

Import data from the "ranges.js" file located in the root directory of the eZ Publish installation. The script will use the specified custom database name, user name and password, while the type of the database and the hostname will still be fetched from the database configuration of the default siteaccess.

5

db-driver

db-driver --db-driver=<database_type>

The type of the database, for example "ezmysql". If omitted, the type specified in the "DatabaseImplementation (page 2343)" configuration directive will be used.

5

db-database

db-database --db-database=<database_name>

The name of the database to update. If omitted, the database name specified in the "Database (page 2359)" configuration directive will be used.

5

db-password

db-password --db-password=<password>

The password that belongs to the database user. If omitted, the password specified in the "Password (page 2360)" configuration directive will be used.

5

db-user

db-user --db-user=<database_user>

The username that will be used to access the database. If omitted, the name specified in the "User (page 2361)" configuration directive will be used.

5

db-host

db-host --db-host=<database_host>

The hostname or the IP address of the server running the database engine, for example "localhost". If omitted, the value specified in the "Server (page)" configuration directive will be used.

5

file

file --file=<path_to_file>

The path to the file that contains the list of ranges (relative to the root directory of eZ Publish). This parameter is required.

5

no-logfiles

no-logfiles --no-logfiles

Do not generate log files (default).

logfiles

logfiles --logfiles

Generate log files. This parameter must be used together with the debug parameter (”-d” or ”--debug”). The log files (for example ”warning.log”) will be stored in the ”var/log” directory of the eZ Publish installation.

5

no-colors

no-colors --no-colors

Do not use ANSI colors when generating debug output.

5

colors

colors -c, --colors

Use ANSI colors when generating debug output (default).

debug

debug -d[<option>[,<option2>...]], --debug[=<option>[,<option2>...]]

Output debug information. The following options can be used to control what the debug output should include: "all" (everything), "accumulator" (accumulators), "include" (included files), "timing" (timing points), "error" (errors), "warning" (warnings), "notice" (notices).

5

quiet

quiet -q, --quiet

Run the script in "quiet" mode. No output will be generated.

5

siteaccess

siteaccess -s <siteaccess>, --siteaccess=<siteaccess>

The siteaccess to run the script for. If omitted, the default siteaccess (page 2527) will be used.

5

help

help -h, --help

Output information about the script and supported parameters.

updateisbn13.php (as of 4.5)

updateisbn13.php (as of 4.5) Imports the latest ISBN ranges to the database.   ./bin/php/
updateisbn13.php

This script makes it possible to import the latest ISBN ranges to the database. Without this
information, the system may not be able to handle ISBN-13 codes properly (refer to the docu-
mentation of the ISBN datatype (page 543) for more information). Since the calculation rules
for ISBN codes can be changed on a monthly basis, you may need to run the script regularly in
order to keep the range information in the database up to date.

When running the script, use the "--url" parameter to tell the script where to look for the range
file. By default, the database configuration of the default siteaccess will be used. You can use the
optional "--siteaccess" parameter to make the script import ranges to the database that is used
by a particular siteaccess. In addition, the optional "--db-database", "--db-driver", "--db-host", "-
-db-user" and "--db-password" parameters make it possible to combine custom database options
with the database configuration of the specified (or default) siteaccess.

Example 1

```
php bin/php/updateisbn13.php --url=http://www.isbn-international.org/
agency?rmxml=1
```

Import data from isbn international to the database that is used by the default siteaccess.

Example 2

```
php bin/php/updateisbn13.php --url=http://www.isbn-international.org/
agency?rmxml=1 --siteaccess=example
```

Import data from isbn international to the database that is used by the "example" siteaccess.

Example 3

```
php bin/php/updateisbn13.php --url=http://www.isbn-international.org/
agency?rmxml=1 --siteaccess=example --db-database=test
```

Import data from isbn international. The script will use the database configuration of the "ex-
ample" siteaccess but ignore the value of the "Database" directive and use the specified custom
database name instead. If the "example" siteaccess makes use of a database called "foo" on local
host, the script will import ISBN ranges to the "test" database on localhost.

Example 4

```
php bin/php/updateisbn13.php --url=http://www.isbn-international.org/
agency?rmxml=1 --db-database=test db-user=root db-password=secret
```

Import data from isbn international. The script will use the specified custom database name, user name and password, while the type of the database and the host name will still be fetched from the database configuration of the default siteaccess.

no-logfiles

no-logfiles --no-logfiles

Do not generate log files (default).

5

logfiles

logfiles --logfiles

Generate log files. This parameter must be used together with the debug parameter ("-d" or "--debug"). The log files (for example "warning.log") will be stored in the "var/log" directory of the eZ Publish installation.

5

no-colors

no-colors --no-colors

Do not use ANSI colors when generating debug output.

5

colors

colors -c, --colors

Use ANSI colors when generating debug output (default).

debug

debug -d[<option>[,<option2>...]], --debug[=<option>[,<option2>...]]

Output debug information. The following options can be used to control what the debug output should include: "all" (everything), "accumulator" (accumulators), "include" (included files), "timing" (timing points), "error" (errors), "warning" (warnings), "notice" (notices).

5

quiet

quiet -q, --quiet

Run the script in "quiet" mode. No output will be generated.

5

siteaccess

siteaccess -s <siteaccess>, --siteaccess=<siteaccess>

The siteaccess to run the script for. If omitted, the default siteaccess (page 2527) will be used.

5

help

help -h, --help

Output information about the script and supported parameters.

5

5.17.2 Upgrade

Upgrade Scripts for updating the existing content of sites when upgrading from older versions.
./update/common/scripts

The "./update/common/scripts" directory contains various scripts that are intended to update the existing site data when an eZ Publish installation is being upgraded. Refer to "The system upgrade scripts" documentation page for more information.

5

5.17.3   Cronjobs

Cronjobs Scripts for automated periodic and scheduled maintenance. ./cronjobs

The "./cronjobs" directory contains cronjob scripts for automated periodic and scheduled maintenance. These are executed by the main maintenance script called "runcronjobs.php". Refer to "The cronjob scripts (page 356)" documentation page for more information.

5

5.17.4   Miscellaneous

Miscellaneous Scripts that do not belong to any of the groups above. ./

Not documented yet.

5