eZ Flow

Installation and Configuration Guide

version 1.1

Table of Contents

1. Introduction

eZ Flow is an extension to eZ Publish that has extended functionality for handling the flow of content publication over time, specifically the rotation and placement of content on frontpages. Typical usage is a newspaper frontpage or managing a TV station's portal. eZ Flow is ideal in situations where content changes quickly and there are many changes per day. eZ Flow is design to enable editors to plan the "flow" of content on these pages.

1.1. Target audience

This document explains how to configure eZ Flow and create custom page layouts and block types, and thus is appropriate for programmers or system administrators with a moderate degree of technical understanding and experience with eZ Publish. For information on using eZ Flow, refer to the Usage Guide.

1.2. Conventions

- Code samples, functions, variable names, and on are printed in `monospace font`.
- Filenames and paths are printed in *`monospace italic font`*.
- Commands are printed in **`monospace bold font`**.
- Elements of graphical user interfaces (such as buttons and field labels) are printed in **bold font**.
- Component names (such as an application) are capitalized, for example "Administration Interface".
- In sample URLs, replace "example.com" with the domain name of your site.
- The screenshots in this document might have been modified to fit the page or to illustrate a point, and therefore might not exactly match the display on your site.

1.3. More resources

For assistance with eZ Publish, refer to the following resources:

- **eZ Publish documentation**: eZ Find is an extension to eZ Publish. Where appropriate, there are links in this document to the online versions of the eZ Publish documentation, located at http://www.ez.no/doc.
- **eZ Publish forums**: The forums on the eZ Systems website are a valuable community-driven resource, where eZ Publish users provide assistance and support to each other. Accessing the forums is free. The forums are located at http://ez.no/community/forum.
- **Support from eZ Partners**: eZ's global network of partners provides professional assistance for all eZ products. To find a partner, contact sales@ez.no.
- **Other eZ solutions**: For information about other solutions provided by eZ Systems, refer to http://ez.no/products/solutions.
- **Training and certification**: eZ Systems and eZ Partners offer training courses and certifications for eZ Publish. Contact sales@ez.no or visit http://ez.no/services/training for more information.

1.4. Contacting eZ

For non-technical questions regarding eZ Publish or eZ Systems, please contact us:

- http://ez.no/company/contact
- info@ez.no

1.5. Copyright and trademarks

2. Installing eZ Flow

To install eZ Flow, you must first install the packages on your web server and prepare the database. This process is exactly the same as installing the regular version of eZ Publish.

2.1. Requirements

First, ensure that the web server meets the requirements:

http://ez.no/doc/ez_publish/technical_manual/4_0/installation/normal_installation/requirements_for_doing_a_normal_installation

Ensure that the "Rewrite rules" in the Apache web server settings are configured. These are used by eZ Publish for interpreting the links to other languages on multi-language sites. For an explanation of the rewrite rules, refer to:

http://ez.no/doc/ez_publish/technical_manual/4_0/installation/virtual_host_setup

eZ Flow 1.1 series require following "Rewrite rules":

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

Requirements for working with the Website Interface (such as supported browser versions) are the same as those for the Administration Interface:

http://ez.no/doc/ez_publish/user_manual/4_0/the_administration_interface

The Online Editor (present in both the Website Interface and the Administration Interface) has a more specific set of requirements:

http://ez.no/doc/extensions/online_editor/4_x/requirements

2.2. Installing on Linux/UNIX

After checking that your system meets the requirements, install the software according to the instructions located at:

http://ez.no/doc/ez_publish/technical_manual/4_0/installation/normal_installation/installing_ez_publish_on_a_linux_unix_based_system

In the first step, you will create a database for eZ Publish. Next, you will download, unpack and install the eZ Publish distribution. When you reach the step that instructs you to start the Setup Wizard, proceed to the next section of this document.

2.3. Installing on Windows

After checking that your system meets the requirements, install the software according to the instructions located at:

http://ez.no/doc/ez_publish/technical_manual/4_0/installation/normal_installation/installing_ez_publish_on_windows

In the first step, you will create a database for eZ Publish. Next, you will download, unpack and install the eZ Publish distribution. When you reach the step that instructs you to start the Setup Wizard, proceed to the next section of this document.

3. Running the Setup Wizard

The Website Interface is installed via the regular eZ Publish Setup Wizard. This Setup Wizard documentation is located at:

http://ez.no/doc/ez_publish/technical_manual/4_0/installation/the_setup_wizard

To initiate the eZ Publish Setup Wizard, browse to the URL http://example.com/index.php, where “example.com” is the name of the web server where you installed eZ Publish.

When installing the Website Interface, note the following:

- **Welcome page**: The eZ Publish Setup Wizard's Welcome page has a button labeled **Finetune**. This option should only be used by people with considerable experience at installing eZ Publish. Most people should click the **Next** button.
- **Site package**: This page lists all the available site packages from http://packages.ez.no/ezpublish/4.0/. (If you cannot see any packages on this page, ensure that your machine has access to the internet.) Locate the package called **eZ Flow**, enable the radiobutton next to it, then click **Next**.
- **Site details**: The **User path** and **Admin path** fields refer to the “siteaccess”. A siteaccess is a collection of configuration settings that dictates the appearance and behavior of the website. For example, these settings determine which design to use and what language to display. An eZ Publish installation may contain multiple siteaccesses, one for each language, in addition to a siteaccess for the Administration Interface.
    - Change the **Title** of the web site to the name you want your site to display in browsers, search engine results, etc.
    - The **Site url** field only needs to be changed if you intend to move the site to a different location. (Consult the online documentation for instructions if this is the case.)
    - The **User path** field contains the name of the siteaccess that is used as the default when a visitor accesses the web site. If you selected English as the default language, the default user path will be “eng” (for example, requests for “http://www.example.com” will be redirected to “http://www.example.com/eng”).
    - It is not necessary to specify the **Admin path**, because an eZ Publish site that uses the Website Interface uses a default admin path (“http://www.example.com/ezwebin_site_admin/”).
- **Site administrator**: This is the default administrative user for the eZ Publish site. Creating additional accounts (including non-administrative accounts used for daily tasks) is described in the eZ Publish documentation. All user accounts must have a unique email address. Therefore, do not use an email account for the admin user that you intend to use later for your “regular” user account.

3.1. Accessing your site

To access your web site, enter the URL you configured during the installation. For example, if you specified “http://example.com/” as the address of your web site, enter that URL in a browser to access the site.

Note that it may take longer than usual to access web pages the first time. This is because the system creates a cache for each page that you access. Once the cache is created, accessing the page in the future will be faster.

To access the Administration Interface for your site, load the URL “http://www.example.com/ezwebin_site_admin/”.

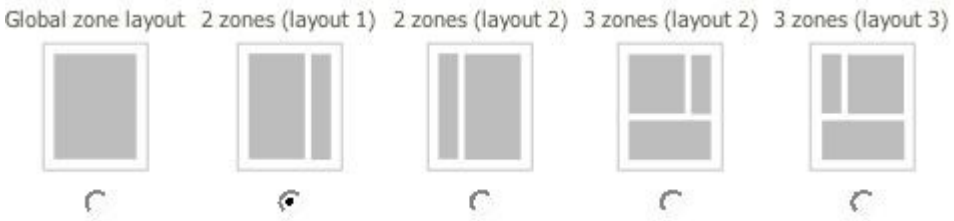
4. Configuring the block system in eZ Flow

When setting up the block system in eZ Flow there are two main things you can configure. You can set up any number of page layouts and you can define any number of custom block types.

By default eZ Flow comes with a Frontpage class with the `Layout` datatype. You can also create custom classes that use the block system. To do this you simply create a new class and add an attribute of the `Layout` datatype.

4.1. Defining page layouts

By default eZ Flow has five layouts, as shown in the screenshot below (which shows the editor's view for selecting layouts in the Website Interface). Because layouts are defined via a configurable plugin system, you can create custom layouts.



4.1.1. Defining new layouts

A "layout" is a combination of zones that are placed on a page. The placement of the zones is defined in a template that is specified as part of the layout configuration. You can define as many layouts as you need. To define a new layout, edit the *`zone.ini`* file (located by default in the directory *`extension/ezflow/settings/zone.ini.append.php`*). Add the layout name to the `AllowedTypes` array.

Once you have defined the layout type, define the following settings:

Setting Name	*Description*	*Valid values*
`ZoneTypeName`	The name of the layout.	String
`Zones[]`	An array defining the available zones and their identifier.	String
`ZoneName[index]`	The visible name of the indexed zone. Use the zone name as the index value.	String
`ZoneThumbnail`	The filename of the thumbnail image to use. The thumbnail shows the arrangement of the zones. The thumbnails are by default located in the directory *`extension/ezflow/design/standard/images/`*	Filename

	ezpage/thumbnails. If you are using a custom design, place the thumbnails in the folder *design/<your_design>/images/ezpage/thumbnails.*	
`Template`	The file containing the template that should be used for the layout. The layout templates are by default located in the directory *extension/ezflow/design/standard/templates/zone.* If you have a custom design the directory will be *design/<your_design>/templates/zone*	Filename
`AvailableForClasses[]`	Zone will be available for classes defined in this array.	String [class_identifier]

This example shows a configuration for a layout called "`2ZonesLayout1`":

```
[General]
AllowedTypes[]=2ZonesLayout1

[2ZonesLayout1]
ZoneTypeName=2 zones (layout 1)

Zones[]=left
Zones[]=right

ZoneName[left]=Left zone
ZoneName[right]=Right zone

ZoneThumbnail=2zones_layout1.gif

Template=2zoneslayout1.tpl
AvailableForClasses[]=frontpage
```

4.1.2. Layout templates for zone definition

The template used by a layout must display the contents of the layout's zones. You can place the zones anywhere in the template. The variable that is made available in the template is called `$zones`. This variable is an array containing different `$zone` objects. Each `$zone` variable contains both a `$zone.name` and `$zone.blocks`.

A simple example of a layout template is shown below.

```
{foreach $zones as $zone}
```

```
<h1>Zone: {$zone.name}</h1>

    {foreach $zone.blocks as $block}
        {block_view_gui block=$block}
    {/foreach}

{/foreach}
```

4.2. Defining block types

A “block” is a section within a zone that contains a particular type of content. The content in blocks may be manually specified by a site editor (a “manual block”), may be automatically selected from a specific area in the eZ Publish content tree (a “dynamic block”) or may contain special content such as a tag cloud or banner ads (a “special block”).

The configuration file for managing blocks is called *block.ini*. The default configuration file is located in the eZ Flow extension (extension/ezflow/settings). When creating custom blocks, you should make an appropriate override, either globally or for a specific siteaccess. (Refer to the eZ Publish documentation for information about overrides.)

4.2.1. Creating manual blocks

To create a manual block, enter information about the block in a new section in the *block.ini* file. The definition of a manual block consists of the following elements.

Setting Name	Description	Valid values
Name	The name of the block.	String
NumberOfValidItems	The number of items that the block can display.	Integer
NumberOfArchivedItems	The number of archived items to display.	Integer
ManualAddingOfItems	Enable manually adding items.	enabled or disabled
ViewList[]	Defines the block views used by the template override system.	Block view
ViewName[view_id]	The list of view names displayed in the eZ Flow interface.	ViewList[] name mapped to human-readable name

Below is a configuration example for a three-column news block.

```
[Manual3Items]
Name=3 Column News
NumberOfValidItems=3
NumberOfArchivedItems=5
ManualAddingOfItems=enabled
ViewList[]=3_items1
```

```
ViewName[3_items1]=3 articles in one column
```

After the block is defined, it is displayed to editors in the **Block type** drop-down list in the eZ Flow interface, as shown in the following screenshot.

4.2.2. Template for manual blocks

To create a new block, you must also create an override template. (If no override exists, the default template `templates/block/view/view.tpl` is used.) The templates for manual blocks are located in `design/<custom_design>/override/templates/block`.

Every block override can use two override keys: `Match[type]=` and `Match[view]=`. `Match[type]=` takes the unique block identifier defined in the *`block.ini`* file as a parameter. `Match[type]=` takes the view's unique identifier defined in the block definition in *`block.ini`* as a parameter.

For example:

```
[block_3_items1]
Source=block/view/view.tpl
MatchFile=block/3_items1.tpl
Subdir=templates
Match[type]=Manual3Items
Match[view]=3_items1
```

In every block template a `$block` variable is available that has following attributes:

Attribute	Type	Example value
`id`	String (32 char)	680b73edef7c07d8f3d9de429b4d9b4d
`name`	String	3 Column News
`zone_id`	String (32 char)	082188b2a113cc866cbcf6ca6e5ce7d3
`type`	String	Manual3Items
`view`	String	3_items1
`overflow_id`	String (32 car)	32 char long string that is the id of another block
`fetch_params`	String	Serialized fetch parameters used by dynamic block
`waiting`	Array	Items in the queue
`valid`	Array	Items currently published
`valid_nodes`	Array	Items as objects of

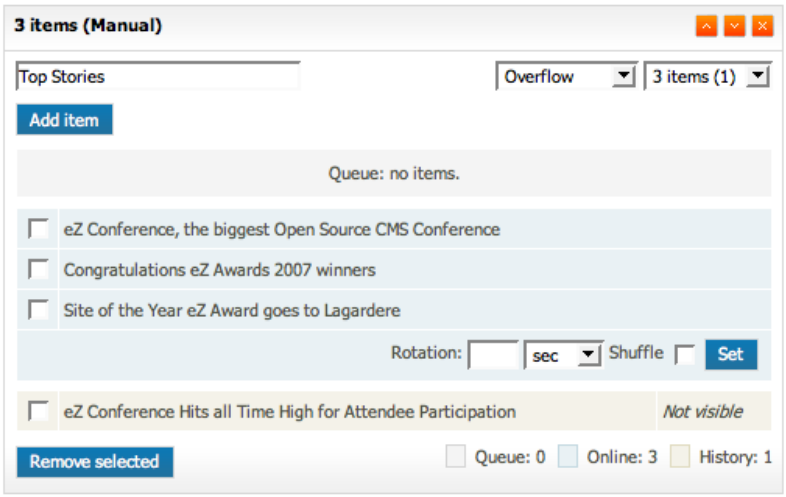
		eZContentObjectTreeNode
archived	Array	List of archived items
view_template	String	View
edit_template	String	Edit

The template code for displaying three items in a column using div-based layout might look like the example below:

```
<div class="attribute-header">
      <h2>{$block.name}</h2>
</div>

{foreach $block.valid_nodes as $valid_node}
      {node_view_gui view=line content_node=$valid_node}
{/foreach}
```

4.2.3. The result of the manual block

After the block is defined editors can add items to the block (as explained in the eZ Flow User Guide). For example, a block with three items might look (in the eZ Flow interface) like the screenshot below:



The final result on the website might look like the screenshot below:

This block is manual, meaning that the editor selects the items to display. There is no automatic fetching of items.

4.2.4. Defining dynamic blocks

While the content in manual blocks is specified by an editor, dynamic blocks are a combination of automatic fetching rules (implemented via a plugin system) and editorially controlled parameters.

The dynamic block settings have more options than the manual block. The following settings need to be configured for each dynamic block.

Setting Name	*Description*	*Valid values*
`Name`	The descriptive name of the block.	Text string
`NumberOfValidItems`	The number of items that can be displayed in the list.	Integer
`NumberOfArchivedItems`	The number of items to have in the visible archive.	Integer
`ManualAddingOfItems`	Disable the ability to manually add items in the block.	Enabled or disabled
`FetchClass`	The class to use for the dynamic fetching rules (PHP class).	String
`FetchFixedParameters[]`	A list of fixed parameters to send to the fetching class.	String value that sets the key and value: `FetchFixedParameters[class]=article`
`FetchParameters[]`	Definition list of all the fetch	Holds list of parameters

	parameters.	used by FetchClass, which are displayed in eZ Flow interface.
`FetchParametersSelectionType[]`	Definition of the selection type to use for the specific fetch parameter. The parameters are added by the editor. This can either be a single or multiple selection of content items.	Single or multiple
`FetchParametersIsRequired[]`	Definition of the fetch parameters that are required. If none are defined all are optional.	True or false
`ViewList[]`	Defines the block views used by the template override system.	Block view
`ViewName[view_id]`	The list of view names displayed in the eZ Flow interface.	`ViewList[]` name mapped to human-readable name
`AllowedClasses[]`	The list of classes which user can add into the manual block type.	Content class identifiers

The following example shows a configured dynamic block:

```
[Dynamic3Items]
Name=3 items (dynamic)
NumberOfValidItems=3
NumberOfArchivedItems=5
ManualAddingOfItems=disabled
FetchClass=ezmLatestObjects
FetchFixedParameters[]
FetchFixedParameters[Class]=article
FetchParameters[]
FetchParameters[Source]=nodeID

# Single / Multiple
FetchParametersSelectionType[Source]=single

FetchParametersIsRequired[]
# True / False
FetchParametersIsRequired[Source]=true

ViewList[]=3d_items
ViewName[3d_items]=3 dynamically fetched items
```

4.2.5. Template for dynamic blocks

To create a new block, you must also create an override template. (If no override exists, the default template `templates/block/view/view.tpl` is used.) The templates for

dynamic blocks are located in `design/<custom_design>/override/templates/block`.

Every block override can use two override keys: `Match[type]=` and `Match[view]=`. `Match[type]=` takes the unique block identifier defined in the *`block.ini`* file as a parameter. `Match[type]=` takes the view's unique identifier defined in the block definition in *`block.ini`* as a parameter.

For example:

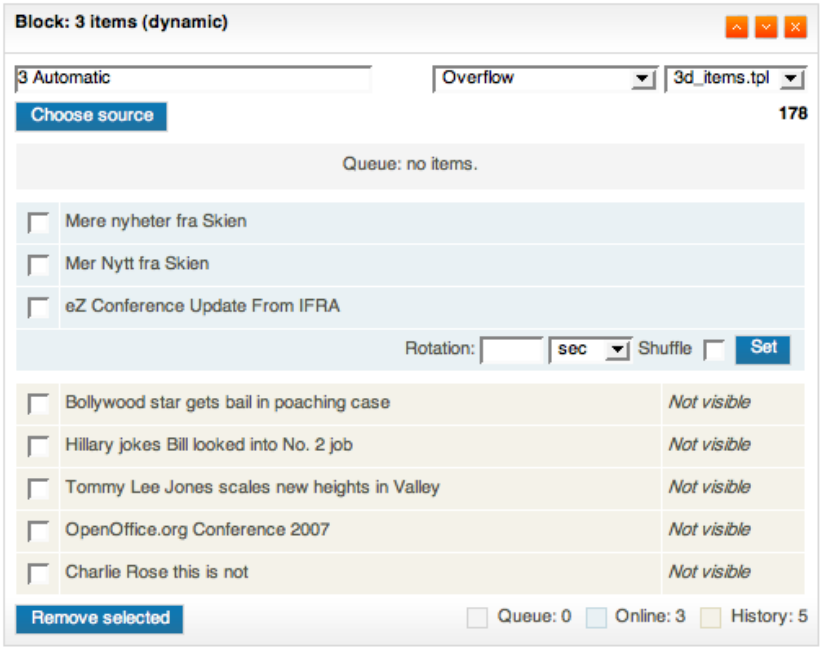
```
[block_3d_items]
Source=block/view/view.tpl
MatchFile=block/3d_items.tpl
Subdir=templates
Match[type]=Dynamic3Items
Match[view]=3d_items
```

The template code for displaying the dynamic block using `div`-based layout might look like the example below:

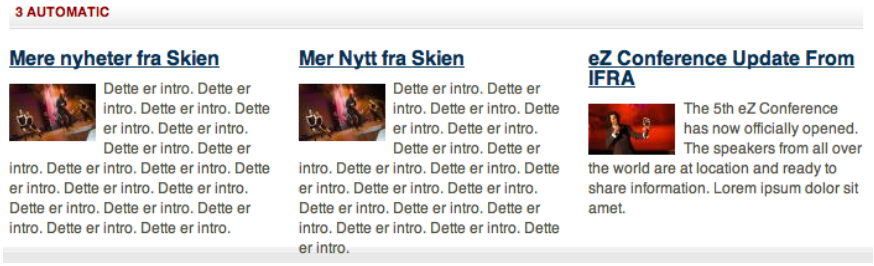
```
{foreach $block.valid_nodes as $valid_node}
     {node_view_gui view=line content_node=$valid_node}
{/foreach}
```

4.2.6. The result of the dynamic block

The following screenshot shows how an dynamic block appears to an editor working in the eZ Flow interface. Notice that after choosing the source(s), the name(s) are displayed to the right of the **Choose source** button.



The display of the dynamic block described above on the website is shown below. From the site visitor's perspective, is looks the same as a manual block. The difference is that when something new is published in the specified block source it is automatically displayed and does not require an editor to make the content "live".



4.2.7. Defining special blocks

In addition to manual and dynamic blocks, eZ Flow also allows for special blocks. Special blocks do not have a pre-defined list of content. A special block can be used to publish content entities like tag clouds or banner ads.

The following settings need to be configured for each special block.

Setting Name	*Description*	*Valid values*
`Name`	The descriptive name of the block.	Text string
`ManualAddingOfItems`	For special blocks disable the ability to manually add items in the block.	Enabled or disabled
`CustomAttributes[]`	List of attributes used by the block. It will show as text input fields where editors can add parameters.	Array
`UseBrowseMode[]`	When set to true, the eZ Flow interface will display a browse button.	True or false
`ViewList[]`	Defines the block views used by the template override system.	Block view
`ViewName[view_id]`	The list of view names displayed in the eZ Flow interface.	`ViewList[]` name mapped to human-readable name

An example of the settings for a special block that displays a tag cloud is shown below.

```
[TagCloud]
Name=Tag cloud
ManualAddingOfItems=disabled
CustomAttributes[]=subtree_node_id
UseBrowseMode[subtree_node_id]=true
ViewList[]=tag_cloud
ViewName[tag_cloud]=Tag cloud
```

4.2.8. Template for special blocks

To create a new block, you must also create an override template. (If no override exists, the default template `templates/block/view/view.tpl` is used.) The templates for special blocks are located in `design/<custom_design>/override/templates/block`.

Every block override can use two override keys: `Match[type]=` and `Match[view]=`. `Match[type]=` takes the unique block identifier defined in the *`block.ini`* file as a parameter. `Match[type]=` takes the view's unique identifier defined in the block definition in *`block.ini`* as a parameter.

For example:

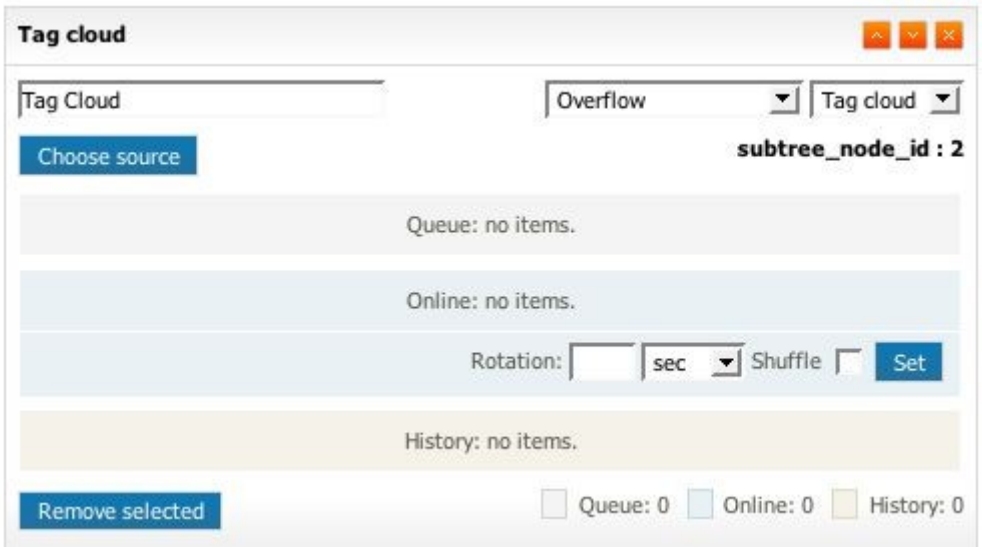
```
[block_tag_cloud]
Source=block/view/view.tpl
MatchFile=block/tag_cloud.tpl
Subdir=templates
Match[type]=TagCloud
Match[view]=tag_cloud
```

The template for a tag cloud is show below:

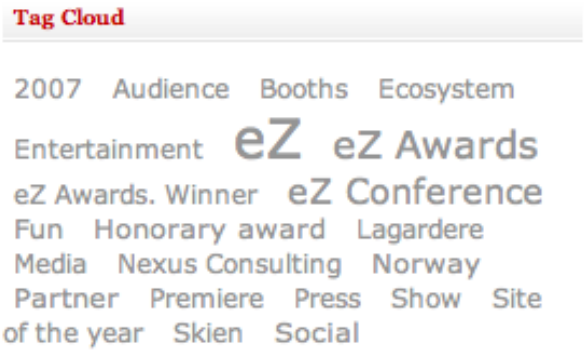
```
<div class="attribute-header">
     <h2>{$block.name}</h2>
</div>

<div class=”attribute-tags”>
     {eztagcloud( hash( 'parent_node_id',
$block.custom_attributes.subtree_node_id ))}
</div>
```

4.2.9. The result of a special block

The tag cloud is displayed in the eZ Flow interface like this:



A site visitor sees the tag cloud like this:



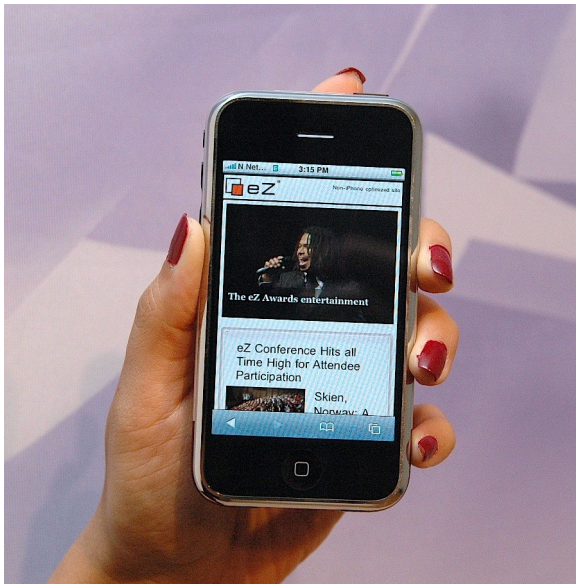
5. iPod Touch and iPhone template example

eZ Flow includes templates tuned for mobile display. These templates are examples that show how you can display content on various devices.

Displaying content on alternate devices is enabled via the eZ Publish siteaccess system, with a separate siteaccess for each type of device. Each siteaccess contains custom templates, specifically:

- Pagelayout
- Article full view
- Gallery full view

The documentation for the templates used in eZ Publish can be found at http://ez.no/doc/ez_publish/technical_manual/4_0/templates



The Apple iPhone includes a full-featured webbrowser which means that it can display most website content (excluding elements such as Flash). It is normal to tune the content specifically for mobile devices. In the case of iPhones, you simply provide some templates that are tuned for the display size (320 pixels wide).

For information on how to tune your CSS and templates for iPhone, visit Apple's development resources:

http://developer.apple.com/iphone/devcenter/designingcontent.html

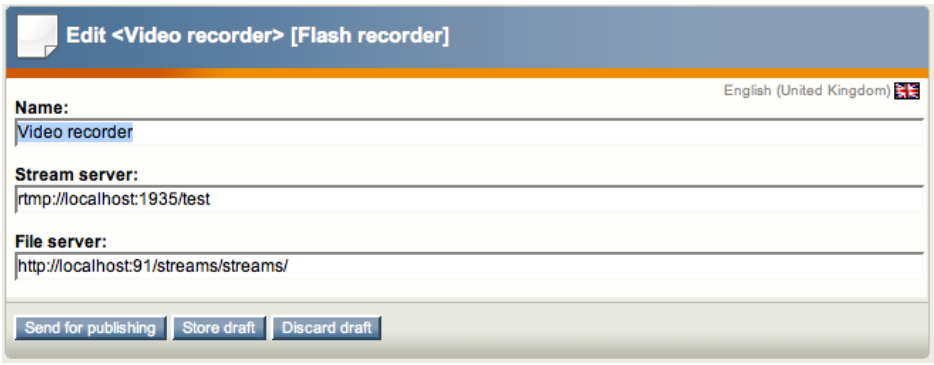
6. Video streaming configuration

The video streaming functionality included with eZ Flow enables recording and streaming live video and playback of recorded videos. It works in conjunction with the Red5 streaming server. For installation information on Red5 visit: http://osflash.org/red5.

6.1. Configuring the video recorder

You can create multiple video recorder interfaces in eZ Flow. Each needs to be connected with the streaming server (Red5) and a file server (Apache). The screenshot below shows the dialog for creating a new **Video recorder** object.

- **Name:** The name of the recorder.
- **Streaming server:** The URL of the rtmp location of Red5.
- **File server:** The server where you have configured Apache to list and serve the previously recorded videos.



6.2. Configuring the file server

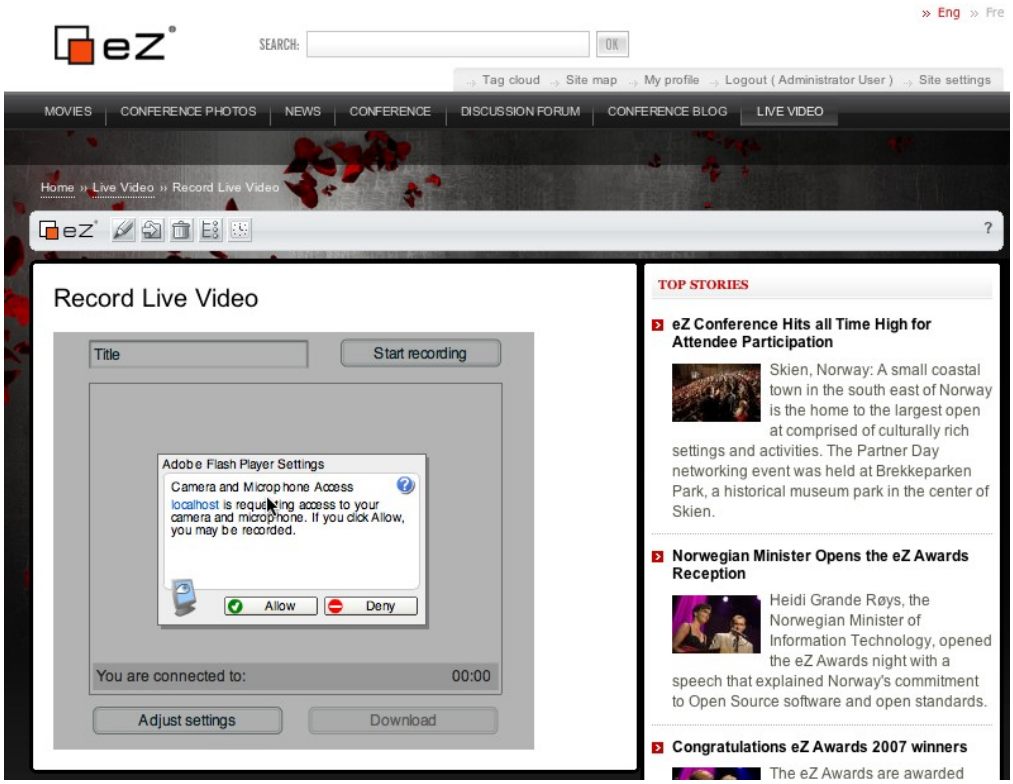
Red5 must be installed on a webserver where Apache is also installed. Apache can be used to list and serve the files that have already been recorded. This is an example of the Apache configuration:

```
# Red5 Streams sharing
<VirtualHost *:100>
    <Directory /Applications/Red5/webapps/test/streams>
        Options FollowSymLinks
        AllowOverride None
    </Directory>

    DocumentRoot /Applications/Red5/webapps/test/streams
    ServerName localhost
</VirtualHost>
```

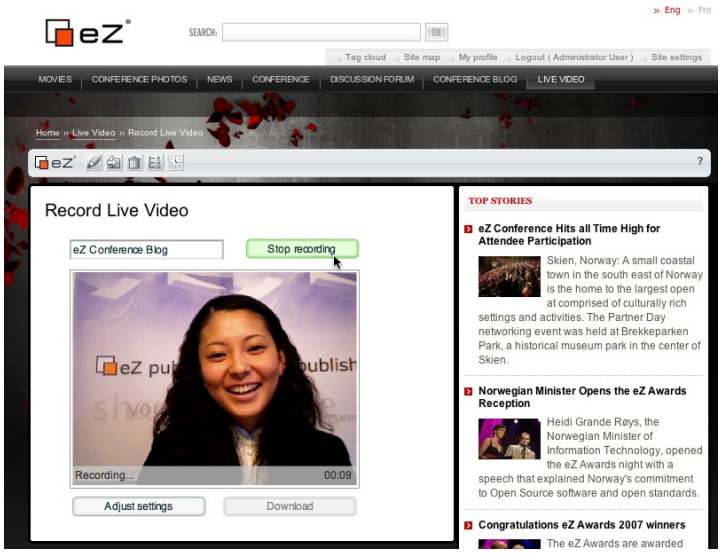
The path */Applications/Red5/webapps/test/streams* should point to the directory in Red5 where the uploaded/streamed files are stored. This depends on the Red5 configuration - refer to the Red5 documentation for details. The file *flash_video_list.php* also needs to be stored in that directory. This example makes the *flash_video_list.php* and the streamed files available on port 100. Refer to the Apache virtualhost configuration for more information on how to configure this for your site.

6.3. Recording video

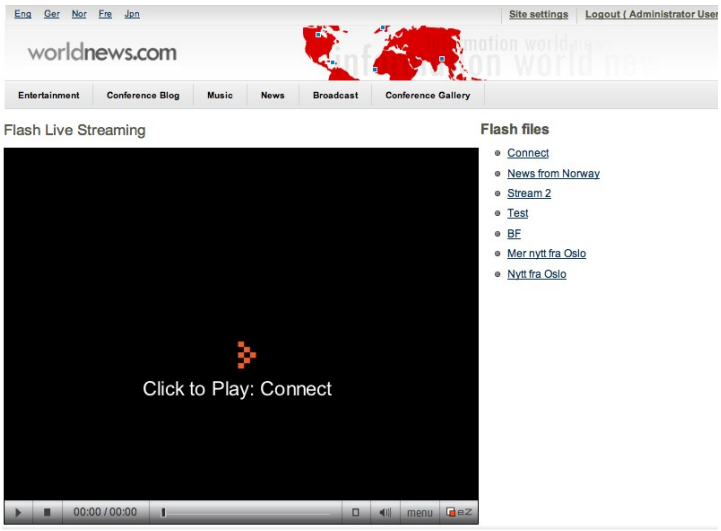
Once the player object has been created you can create video recordings. The screenshot below shows the video recorder's initial dialog where you allow Flash to access your camera.



After Flash is connected to the camera, you have access to the video recorder as shown below. The title of the video can be entered before you start the recording. Once the recording has started the video is available as a live stream for site visitors. Once you stop the recording, the file is encoded as a Flash video and made available for download. Alternatively, the file can be edited in video editing software.



The screenshot below shows the streaming interface and a list of the most recently added videos. If there is currently a live streaming session, it will be automatically listed.



7. Appendix: Fetch Functions

There are four fetch functions defined in the `ezmedia` module which can be used in templates:

- `waiting`
- `valid`
- `archived`
- `valid_nodes`

The description of these fetch functions follows.

7.1. waiting

The fetch function has one parameter, `block_id`. This fetches the waiting items (items in the queue) for a block identified by the `block_id` parameter. The result is the array of items represented by an array with following keys:

- `block_id`
- `object_id`
- `node_id`
- `priority`
- `ts_publication`
- `ts_visible`
- `ts_hidden`
- `rotation_until`
- `moved_to`

Example:

```
fetch( 'ezmedia', 'waiting', hash( 'block_id', $block.id ) )
```

7.2. valid

The same as the `waiting` fetch function, but the array of valid items is returned.

7.3. archived

The same as the `waiting` fetch function, but the array of archived items (items in the archive) is returned.

7.4. valid_nodes

The same as `valid` fetch function, but the array of node objects (for example, of `eZContentObjectTreeNode` class) is returned. This fetch function should be used in the template rendering the block.