# gs.group.messages.post.base Documentation

***Release 4.0.0***

**GroupServer.org**

April 28, 2016

Contents


1 Design requirements 3

2 The post content provider 5

3 Microformats used in the post 7

4 The structure of a post 9

5 Changelog 13

6 Indices and tables 19

7 Resources 21

**Author** Michael JasonSmith

**Contact** Michael JasonSmith <mpj17@onlinegroups.net>

**Date** 2015-09-30

**Organization** GroupServer.org

**Copyright** This document is licensed under a Creative Commons Attribution-Share Alike 4.0 International License by OnlineGroups.net.

Contents:

CHAPTER 1

# Design requirements

1. The rendering of a post needs to be **fast.** Posts are the core GroupServer, and are used very heavily.
2. The system to display posts needs to be **flexible.** People can add posts containing any old thing, such as HTML, and plain text.
3. A post may be shown in a number of different **contexts:**
   - On the *Topic* page
   - On the *Post* page
   - On the (rarely visited) *Posts* page
4. The system needs to handle **actions**. While the normal thing to do with a post is to read it, people also hide and share posts.
   - **Hiding** posts needs to be restricted to just the administrator and the author.
   - **Sharing** should be restricted so people are discouraged from sharing links from private groups with the wider public.

CHAPTER 2

# The post content provider

The post content provider is used to render a single message. It provides an *interface*, which is used by the *XML* that generates the page.

## 2.1 Interface

The post content provider implements the interface `IPostContentProvider`. The various attributes are filled by the *XML* provided by the calling code.

**class** `gs.group.messages.post.base.interfaces.`**IPostContentProvider**

**post**
*(Required)* The email message instance that is being displayed.

**position**
The position of the post in the list of posts, as an integer (`int`) above `0` (a *cardinal* value). Defaults to `1`.

**topicName**
*(Required)* The name of the topic that the post appears in, as a Unicode sequence (`str` in Python 3, `unicode` in Python 2). While the post could figure this out, it is faster to have this information provided (see *Design requirements*).

**showPhoto**
Whether the photo of the author should be shown, as a Boolean (`bool`, defaulting to `True`). On both the *Topic* and *Post* page sequential posts by the same author have the profile photo dropped.

**isPublic**
*(Required)* Whether the post is public, as a Boolean (`bool`, with no default). While the post could figure this out, it is faster to have this information provided (see *Design requirements*).

**showRemainder**
Whether to show the snipped bottom-quoting and signature on the post, as a Boolean (`bool`, defaulting to `False`).

## 2.2 XML

The XML that uses the content provider is in two parts. First, the parameters — as defined by `IPost` — are set up by the TAL of the calling code, using the `tal:define` attribute. Second the content provider is called using the `provider:` expression.

For example, the code used by the *Topic* page looks similar to the following.

```
1  <tal:block
2    define="topicName view/topicName;
3            isPublic view/isPublic;">
4    <tal:block repeat="post view/topic">
5      <tal:block
6        define="position repeat/post/number;
7                currAuth python:view.topic[position-1]['author_id'];
8                prevAuth python:view.topic[position-2]['author_id'];
9                rptAuth python: currAuth == prevAuth;
10               showPhoto python:(position==1) or not(rptAuth);"
11       replace="structure provider:groupserver.Post">
12       This is replaced by the body of the post
13     </tal:block>
14   </tal:block>
15 </tal:block>
```

Initially, the `IPost.topicName` and `IPost.isPublic` attributes are set up (lines 2–3). The the topic iterates across all the posts in the topic, setting the `IPost.position` accordingly (line 6). The `IPost.showPhoto` is calculated (line 10), while the `IPost.showRemainder` is left as its default (`False`). Finally, the content provider is called using `structure provider:groupserver.Post`, replacing the content of the `<tal:block>` element (line 11).

CHAPTER 3

# Microformats used in the post

A post uses the following microformats, which can be used to analyse a post.

- Each post is an `<article>` element [1], and it conforms to the `h-entry` microformat [2].
- The metadata for the post is contained within an `<aside>` element [3].
- The topic-title is marked with the `p-name` class, and given a `heading` role [4]. Note that the heading may be hidden by the CSS for the page.
  - The `<article>` and heading are linked by the `aria-labelledby` attribute on the `<article>` [5].
  - The `u-url` (also on the heading) marks the URL of the *permalink* for the post.
- The author metadata is given in an `<address>` element [6], with the `h-card` and `p-author` classes; the name marked with the `p-name` class [7].
- The post-date in in a `<time>` element, with the `datetime` attribute [8] set to the date the post was made in UTC. The `<time>` element is also given the class `dt-published`.

[1] For more on the `<article>` element see <https://developer.mozilla.org/en-US/docs/Web/HTML/Element/article>
[2] See the Microformats wiki for more on `h-entry` <http://microformats.org/wiki/h-entry>
[3] For more on the `<aside>` element see <https://developer.mozilla.org/en-US/docs/Web/HTML/Element/aside>
[4] For more on the `heading` role see <http://www.w3.org/TR/wai-aria/roles#heading>
[5] For more on the WAI-ARIA `aria-labelledby` attribute see <https://developer.mozilla.org/en-US/docs/Web/Accessibility/ARIA/ARIA_Techniques/Using_the_aria-labelledby_attribute>
[6] For more on the `<address>` element see <https://developer.mozilla.org/en-US/docs/Web/HTML/Element/address>
[7] See the Microformats wiki for more on `h-card` <http://microformats.org/wiki/h-card>
[8] For more on the HTML5 Time element see <https://developer.mozilla.org/en-US/docs/Web/HTML/Element/time>

CHAPTER 4

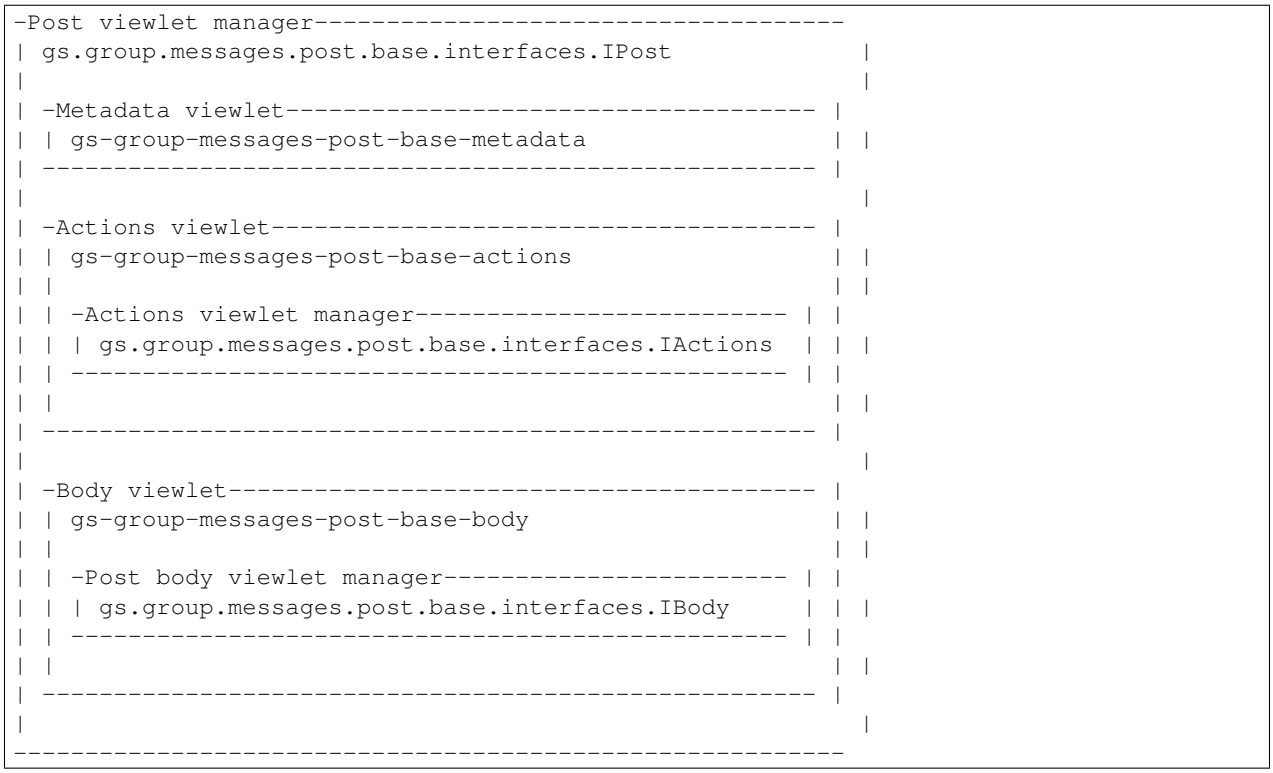
# The structure of a post

The structure of a post in GroupServer may seem simple, but that is just good user-interface design. It is actually quite complex. In this document I will discuss the structure of the viewlets that are used to create the post.

The post itself provides two interfaces.

- To *external* code it looks like a content provider that provides the interface `interfaces.IPostContentProvider` interface.
- To the *internal* code that provide the content of the post it is a *viewlet manager* that provides the `interfaces.IPost` interface.

Three viewlets provided by this product use the post viewlet manager, two of which contain viewlet managers themselves.




**Post:** The overall container that holds the post is a *viewlet manager*, which provides the `interfaces.IPost` interface.

**Metadata:** The metadata for the post is provided by a *viewlet* defined by this product The metadata includes the profile photo of the author, their name, and the date the post was made. (This metadata conforms to some *microformats*.)

**Actions:** The controls the viewer can see (and thereby use) are shown by the Actions. It is split in two: a *viewlet* for the post, which then contains a *viewlet manger* for the content, which provides the class `interfaces.IActions`.

**Post body:** The body is provided by a viewlet that slots into the post viewlet manager. Views of the post provide state that they are managed by the `interfaces.IBody` viewlet manager.

As a consequence of this design, the actual body of the post is provided by other products. Hidden posts are handled by the gs.group.messages.post.hide product, which also controls if the *Hide* button appears in the actions; plain-text posts (`text/plain`) are rendered by the gs.group.messages.post.text product.

## 4.1 Interfaces

Three interfaces are used within the post: `interfaces.IPost` for the overall layout, `interfaces.IActions` for the actions, and `interfaces.IBody` for the actual body.

**class** gs.group.messages.post.base.interfaces.**IPost**
The viewlet manager for the post.

Viewlets that wish to provide content for the post should state that their **manager** is this interface:

```
<browser:viewlet
  name="gs-group-messages-post-base-metadata"
  manager="gs.group.messages.post.base.interfaces.IPost"
  class=".metadata.PostMetadataViewlet"
  template="browser/templates/metadata.pt"
  permission="zope2.View"
  weight="10"
  title="Metadata" />
```

**class** gs.group.messages.post.base.interfaces.**IActions**
The viewlet manager for the post actions.

Viewlets that wish to provide actions for the post should state that their **manager** is this interface:

```
<browser:viewlet
  name="gs-group-messages-post-hide-button"
  manager="gs.group.messages.post.base.interfaces.IActions"
  class=".button.HideButton"
  template="browser/templates/button.pt"
  permission="zope2.View"
  weight="90"
  title="Hide" />
```

**class** gs.group.messages.post.base.interfaces.**IBody**
The viewlet manager for the post body.

Viewlets that wish to provide views for the post body should state that their **manager** is this interface:

```
<browser:viewlet
  name="gs-group-messages-post-text"
  manager="gs.group.messages.post.base.interfaces.IBody"
  class=".body.PlainTextBody"
  template="browser/templates/body.pt"
  permission="zope2.View"
```

```
7     weight="10"
8     title="Text" />
```

CHAPTER 5

# Changelog

## 5.1 4.0.3 (2015-11-12)

- Following the updates to gs.content.js.sharebox
- Running gjslint over the JavaScript resources

## 5.2 4.0.2 (2015-10-30)

- Moving the `file_size_format` function to gs.group.messages.base

## 5.3 4.0.1 (2015-10-27)

- Removing a division that was reducing the reported size of files by an order of magnitude.

## 5.4 4.0.0 (2015-10-19)

- Adding documentation
- Renaming the product gs.group.messages.post.base
- Splitting gs.group.messages.post.hide, gs.group.messages.post.page, and gs.group.messages.post.text off this product
- Using viewlets to generate the post

## 5.5 3.5.0 (2015-09-21)

- Showing the *Rest of post* by default when viewing a single post
- [DE] Updating the product metadata, thanks to Cousin Clara
- Using `subject` rather than `Subject` in `mailto:` URIs

## 5.6 3.4.3 (2015-07-24)

- Hitting code with the PEP-8 stick

## 5.7 3.4.2 (2015-04-16)

- Switching from `<object>` to `<iframe>` for YouTube and Vimeo videos

## 5.8 3.4.1 (2015-03-11)

- [FR] Adding a French translation, thanks to Razique Mahroua

## 5.9 3.4.0 (2015-02-27)

- Naming the reStructuredText files as such
- Switching to GitHub as the canonical repository
- Adding internationalisation support
- Adding support for Transifex

## 5.10 3.3.0 (2014-06-11)

- Hiding the email address only when Anonymous can see the address
- Using the `GroupVisibility` code for displaying the privacy information

## 5.11 3.2.2 (2014-03-17)

- Following some new changes to the *Sharebox*
- Using `strict` mode in the JavaScript
- Switching to Unicode literals

## 5.12 3.2.1 (2013-12-06)

- Switching to the new *Loading* icon

## 5.13 3.2.0 (2013-11-15)

- Following the new *Sharebox* code
- Metadata update
- Fixing some more whitespace

## 5.14 3.1.2 (2013-07-31)

- Fixing some minor white-space

## 5.15 3.1.1 (2013-06-06)

- Minifying the JavaScript code

## 5.16 3.1.0 (2013-04-05)

- Adding icons for files, the navigation links, and breadcrumb trail

## 5.17 3.0.0 (2013-03-22)

- Updating the *Post* page for the new GroupServer user interface
  - Reformatting the files
  - Switching to Twitter Bootstrap for the *Hide* and *Share* interfaces
  - Adding a breadcrumb trail
  - Adding WAI-ARIA markup
- General PEP-8 code cleanup

## 5.18 2.0.1 (2012-10-10)

- Adding a post-date index

## 5.19 2.0.0 (2012-09-21)

- Switching to use the PostgreSQL full-text retrieval vector for searching

## 5.20 1.5.0 (2012-07-18)

- Refactoring the post code to allow `can_hide_post` to be overridden

## 5.21 1.4.0 (2012-06-28)

- Updating the `gs.database` code
- Updating SQLAlchemy
- Updating the cache code
- Fixing the hide-a-post query

## 5.22 1.3.0 (2012-05-16)

- Handling the new `youtu.be` URLs

## 5.23 1.2.0 (2011-09-28)

- Changing the name of the *Hide post* button, and the *Rest of post* button.

## 5.24 1.1.2 (2011-05-06)

- Fixing a problem with long URLs

## 5.25 1.1.1 (2011-04-29)

- Hiding the hide link when the member cannot hide the post
- Fixing some SQL problems

## 5.26 1.1.0 (2011-04-21)

- Adding a user-interface for hiding a post

## 5.27 1.0.1 (2011-04-05)

- Adding back-end support for hidden posts
- Improving the performance

## 5.28 1.0.0 (2011-02-21)

Initial version. Prior to the creation of this product the posts were rendered by `Products.XWFMailingListManager`.

The `gs.group.messages.post.base` product provides the code for displaying posts that have been made to a group, which as some strict *requirements*. It provides the `groupserver.Post` *content provider* that creates *the*

*post* with some *microformats*. This product does not, however, provide the body of the post. This is provided by other products that work with the *structure* provided by this product.

CHAPTER 6

# Indices and tables

- genindex
- modindex
- search

CHAPTER 7

# Resources

- Documentation: http://groupserver.readthedocs.org/projects/gsgroupmessagespostbase/
- Code repository: https://github.com/groupserver/gs.group.messages.post.base/
- Translations: https://www.transifex.com/groupserver/gs-group-messages-post/
- Questions and comments to http://groupserver.org/groups/development
- Report bugs at https://redmine.iopen.net/projects/groupserver

## G

gs.group.messages.post.base.interfaces.IActions (class in gs.group.messages.post.base), 10
gs.group.messages.post.base.interfaces.IBody (class in gs.group.messages.post.base), 10
gs.group.messages.post.base.interfaces.IPost (class in gs.group.messages.post.base), 10

## I

IPostContentProvider (class in gs.group.messages.post.base.interfaces), 5
isPublic (gs.group.messages.post.base.interfaces.IPostContentProvider attribute), 5

## P

position (gs.group.messages.post.base.interfaces.IPostContentProvider attribute), 5
post (gs.group.messages.post.base.interfaces.IPostContentProvider attribute), 5

## S

showPhoto (gs.group.messages.post.base.interfaces.IPostContentProvider attribute), 5
showRemainder (gs.group.messages.post.base.interfaces.IPostContentProvider attribute), 5

## T

topicName (gs.group.messages.post.base.interfaces.IPostContentProvider attribute), 5