

Txp: article / .....	4
Txp: article custom / .....	5
Txp: article id / .....	7
Txp: article image / .....	8
Txp: author / .....	9
Txp: body / .....	9
Txp: breadcrumb / .....	10
Txp: category / .....	11
Txp: category list / .....	12
Txp: category1 / .....	13
Txp: category2 / .....	14
Txp: comment anchor / .....	15
Txp: comment email / .....	15
Txp: comment email input / .....	16
Txp: comment id / .....	18
Txp: comment message / .....	18
Txp: comment message input / .....	19
Txp: comment name / .....	21
Txp: comment name input / .....	21
Txp: comment permalink (1) .....	23
Txp: comment permalink (2) .....	24
Txp: comment preview / .....	24
Txp: comment remember / .....	26
Txp: comment submit / .....	28
Txp: comment time / .....	30
Txp: comment web / .....	30
Txp: comment web input / .....	31
Txp: comments / .....	33
Txp: comments count / .....	34
Txp: comments form / .....	35
Txp: comments help / .....	36
Txp: comments invite / .....	38
Txp: custom field / .....	39
Txp: css / .....	40
Txp: die / .....	41
Txp: else / .....	42
Txp: email / .....	43
Txp: excerpt / .....	43

Txp: feed link / .....	44
Txp: file download / .....	45
Txp: file download category / .....	46
Txp: file download created / .....	47
Txp: file download description / .....	47
Txp: file download downloads / .....	48
Txp: file download id / .....	49
Txp: file download link (1) .....	49
Txp: file download link (2) .....	50
Txp: file download list / .....	51
Txp: file download modified / .....	53
Txp: file download name / .....	53
Txp: file download size / .....	54
Txp: if article category .....	55
Txp: if article list .....	56
Txp: if article section .....	58
Txp: if category .....	58
Txp: if comments .....	60
Txp: if comments allowed .....	61
Txp: if comments disallowed .....	63
Txp: if custom field .....	64
Txp: if different .....	66
Txp: if excerpt .....	67
Txp: if first article .....	68
Txp: if individual article .....	68
Txp: if last article .....	69
Txp: if search .....	70
Txp: if section .....	71
Txp: image / .....	72
Txp: image display / .....	73
Txp: image index / .....	73
Txp: keywords / .....	74
Txp: lang / .....	75
Txp: link / .....	76
Txp: link description / .....	76
Txp: link feed link / .....	77
Txp: link to home .....	78
Txp: link to next .....	78

Txp: link to prev .....	79
Txp: linkdesc title / .....	80
Txp: linklist / .....	80
Txp: meta keywords / .....	82
Txp: meta author / .....	83
Txp: newer .....	83
Txp: next title / .....	84
Txp: older .....	85
Txp: output form / .....	85
Txp: page title / .....	86
Txp: password protect / .....	87
Txp: perm link .....	88
Txp: php .....	89
Txp: popup / .....	89
Txp: posted / .....	90
Txp: prev title / .....	91
Txp: recent articles / .....	91
Txp: recent comments / .....	93
Txp: related articles / .....	94
Txp: search input / .....	96
Txp: search result count / .....	97
Txp: search result date / .....	98
Txp: search result excerpt / .....	98
Txp: search result title / .....	99
Txp: search result url / .....	99
Txp: section / .....	100
Txp: section list / .....	101
Txp: sitename / .....	102
Txp: site slogan / .....	102
Txp: site url / .....	103
Txp: text / .....	104
Txp: thumbnail / .....	104
Txp: title / .....	105

# Txp:article /

## Classification

---

The basic `article` tag is a Single Tag and generally used in a [Page](#) template. The tag is used to output one or more articles depending on [Attributes](#) used. Default attributes will be used when nothing specific is assigned.

`article` is context-sensitive, which means it will grab articles from the currently viewed section and/or category. When used on the front page, article's context will include articles from all sections set to [display "On front page"](#).

(Here is a [comparison](#) of how `<txp:article />` and `<Txp:article_custom />` differ)

## Syntax

---

The `article` tag has the following syntactic structure...

```
<txp:article />
```

## Attributes

---

Tag will accept the following attributes, which can be used individually or in combination as needed (note: attributes are **case sensitive**):

`form="form name"`

Use specified form. Default is `default`. See [Form](#).

`listform="form name"`

Use specified form when page is displaying an article list.

`sortby="sort by"`

Sort retrieved articles. Available values: `ID` (article id#), `AuthorID` (author), `LastMod` (date last modified), `LastModID` (author of last modification), `Posted` (date posted), `Title`, `Category1`, `Category2`, `comments_count`, `Status`, `Section`, `Keywords`, `Image` (article image id#), and `custom_1` through `custom_10`. Please note that multiple attributes can be used, separated with a comma. You can also sort by [date then category](#) (details SQL calls that `sortby` accepts)

`sortdir="sort direction"`

Sort direction of retrieved articles. Available values: `asc` (first to last), or `desc` (last to first).

`limit="integer"`

The number of articles to display. Default is 10.

`offset="integer"`

The number of articles to skip. Default is 0.

`keyword="keywords"`

Restrict to articles with specified keyword(s). Comma-separated list.

`time="time"`

Restrict by time and date posted. Available values: `past`, `future`, or `any` (both `past` and `future`). Default is `past`.

`status="status"`

Restrict by status. Available values: `draft`, `hidden`, `pending`, `live`, `sticky`.

`allowoverride="integer"`

When set to `0`, disables override forms for the generated article list (does not effect individual articles). Available values: `1` or `0`.

`pgonly="integer"`

Do the article count, but do no display anything. Used when you want to show a search result count, or article navigation tags before the list of articles. Available values: `1` or `0`.

Note: `pgonly` tells the tag to do the article/pagination count but not display anything. I added it so you can use pagination tags above the article list, but search result count should work also. Just make sure that, other than `pgonly`, both article tags are identical."

### Examples

---

#### Example 1: Will display 5 articles using the article form Default

```
<txp:article form="default" limit="5" />
```

#### Example 2: Will display 5 articles starting with the third article in the sort order

```
<txp:article offset="2" limit="5" />
```

[Return to tag index](#)

# Txp:article custom /

## Classification

---

The `article_custom` tag is a [Single Tag](#) which provides a variety of custom options for article sorting, selecting, and display.

Textpattern will replace this tag with one or more articles.

Unlike [<Txp:article />](#), `article_custom` will always return an article list, and **is not context-sensitive**. This means that, while [<Txp:article />](#) can only see posts within the currently viewed section, `article_custom` can see all posts from all sections, unless you restrict it via the `section` attribute (see below).

Related Info: [Customizing txp:recent\\_articles](#) (Here is a [comparison](#) of how [<Txp:article />](#) and [<txp:article\\_custom />](#) differ)

## Syntax

---

The `article_custom` tag has the following syntactic structure...

```
<txp:article_custom />
```

## Attributes

---

Tag will accept the following attributes (case-sensitive):

`id="integer"`

The id# of any live article.

`form="form name"`

Use specified form. Default is default. See [Form](#).

`section="section name"`

Restrict to a specified section.

`category="category name"`

Restrict to a specified category.

`sortby="sort by"`

Sort retrieved articles. Available values: ID (article id#), AuthorID (author), LastMod (date last modified), LastModID (author of last modification), Posted (date posted), Title, Category1, Category2, comments\_count, Status, Section, Keywords, Image (article image id#), custom\_1 through custom\_10 and rand() ("random", including the brackets) . Please note that multiple attributes can be used, separated with a comma.

`sortdir="sort direction"`

Sort direction of retrieved articles. Available values: asc (first to last), or desc (last to first).

`limit="integer"`

The number of articles to display. Default is 10.

`offset="integer"`

The number of articles to skip. Default is 0

`excerpted="excerpt value"`

Restrict by excerpt value. Available values: y (containing an excerpt) or n (not containing an excerpt).

`author="author's name"`

Restrict to a specified author.

`month="yyyy-mm"`

Restrict to articles from the specified month.

`keywords="keyword(s)"`

Restrict to articles with specified keyword(s). Put multiple keywords in a comma-separated list.

`time="time"`

Restrict by time and date posted. Available values: past, future, or any (both past and future). Default is past.

`status="status"`

Restrict by status. Available values: draft, hidden, pending, live, sticky.

`allowoverride="integer"`

When set to 0, disables override forms for the generated article list(does not effect individual articles). Available values: 1 or 0.

The attribute `listform` no longer exists for `article_custom` (**only**, it still exists for [article](#)). Instead, make use of [conditional tags](#).

### Examples

---

#### Example 1: Display a list of articles publish during November of 2004

```
<txp:article_custom form="month_list" sortby="Section" sortdir="asc"
month="2004-10" />
```

#### Example 2: Display a list of articles that include "One" as a keyword

```
<txp:article_custom sortby="Posted" sortdir="desc" keywords="One" />
```

#### Example 3: Display a list of hyperlinked article titles by a specific author

```
<txp:article_custom form="author_list" author="Parkling" />
```

the `author_list` article form might go thus.

```
<p><txp:permalink><txp:title /></txp:permalink></p>
```

[Return to tag index](#)

# Txp:article id /

## Classification

---

The `article_id` tag is a [Single Tag](#) which returns the numeric "ID" of the article being displayed. This number will also be reflected as a part of the article permanent URL.

## Syntax

---

The `article_id` tag has the following syntactic structure...

```
<txp:article_id />
```

## Attributes

---

There are no attributes associated with `article_id`

## Examples

---

#### Example 1: Display a hyperlinked article ID number as part of an article form

```
*<txp:permalink><txp:article_id /></txp:permalink>
```

## Example 2: Display a hyperlinked article ID number if an individual article is being displayed

```
<txp:if_individual_article>
```

```
Article ID: <txp:permlink><txp:article_id /></txp:permlink>
```

```
</txp:if_individual_article>
```

# Txp:article image /

## Classification

---

The `article_image` tag is a [Single Tag](#) Textpattern will replace this tag with the IMG SRC html tag matching the numeric "ID" or URL assigned when the article is posted. Its context is in an article form or article content. The image to be associated with the tag is set under the content tab. Display advanced options and enter either the URL of the image, or the Textpattern ID (a number set by Textpattern at upload), Simply put: just the number, in the Article image field.

## Syntax

---

The `article_image` tag has the following syntactic structure...

```
<txp:article_image />
```

## Attributes

---

Tag will accept the following attributes (note: attributes are **case sensitive**):

```
style="style rule"
```

CSS style rule.

```
align="attribute"
```

IMG align attribute.

## Examples

---

### Example 1: Center align the article image.

```
<txp:article_image align="center" />
```

[Return to tag index](#)

# Txp:author /

## Classification

---

The `author` tag is a [Single Tag](#) which is used to return the name of the author of the currently displayed article. Its context is an article form.

## Syntax

---

The `author` tag has the following syntactic structure...

```
<txp:author />
```

## Attributes

---

Tag will accept the following attributes (note: attributes are **case sensitive**):

```
link="value"
```

Make text a link to author's posts. Available values: 1 or 0. Default is 0.

## Examples

---

### Example 1: Display the author's name, linked, as part of an article form

```
<p><txp:title /></p><div class="post">Posted By: <txp:author link="1" /> @  
<txp:posted />...
```

[Return to tag index](#)

# Txp:body /

## Classification

---

The `body` tag is a [Single Tag](#) which is used to return the text, or content, of the article being displayed. (The article itself)

## Syntax

---

The `body` tag has the following syntactic structure...

```
<txp:body />
```

## Attributes

---

This tag has no attributes.

## Examples

---

### Example 1: Display the body, or article text, as part of an article form

```
<p><txp:title /></p><div class="post"><p><txp:author /> @ <txp:posted /><br /><txp:body /></p></div>
```

[Return to tag index](#)

# Txp:breadcrumb /

## Classification

---

The `breadcrumb` tag is a [Single Tag](#) which is used to **create breadcrumb navigation**. Its context is page or column and it provides either hyperlinked navigation, or plain text positional display, anytime you are off the Home page.

## Syntax

---

The `breadcrumb` tag has the following syntactic structure...

```
<txp:breadcrumb />
```

## Attributes

---

Tag will accept the following attributes (note: attributes are **case sensitive**):

`label="label text"`

Uses the site's name if linked.

`link="value"`

Whether or not to hyperlink breadcrumbs. Available values: `y` or `n`. Default is `y`.

`sep="seperator text"`

Character to be used as the breadcrumb separator.

`wraptag="html tag"`

HTML tag to wrap breadcrumb text with. Default is `p`. See [Attributes Cross Reference](#).

`title="value"`

Whether content (like sections) is displayed with the raw name, or with the title. Available values: `y` or `n`. Default is `n`.

## Examples

---

### Example 1: Display a hyperlinked breadcrumb trail

```
<txp:breadcrumb wraptag="p" label="Navigation" sep="::" link="y" />
```

Provides hyperlinks to sections or categories in breadcrumb style, linking back to your home page.

(Breadcrumbs are not displayed on the **Home** page of your site)

### Example 2: Display a text only breadcrumb trail

```
<txp:breadcrumb wraptag="p" label="Navigation" sep=":" link="n" />
```

Provides a breadcrumb guide that reflects where a user is within the site's navigation.

[Return to tag index](#)

# Txp:category /

## Classification

---

The `category` tag is a [Single Tag](#) Textpattern will replace this tag with the name of the current category, if any, or the category as defined with the `name` attribute. Can be used in any context.

## Syntax

---

The `category` tag has the following syntactic structure...

```
<txp:category />
```

## Attributes

---

Tag will accept the following attributes (note: attributes are **case-sensitive**):

`title="integer"`

Display category name or title. Available values: 0 (category name) and 1 (category title). Default is 0.

`link="integer"`

Display as text or link. Available values: 0 (text) and 1 (link). Default is 0.

`wraptag="wraptag text"`

HTML tag to be used as the wraptag, without brackets. Default is empty.

`name="name text"`

Sets link to named category. Default is empty, which sets the link to current category.

`section="name text"`

Restricts category search to named section. Default is empty, which sets the link to current section.

## Examples

---

### Example 1: Displays the current category name

```
<txp:category />
```

### Example 2: Display hyperlinked category name when included in an article form

```
<txp:category link="1" />
```

**Example 3: Display hyperlinked category title when included in an article form.**

```
<txp:category link="1" title="1" />
```

**Example 4: Display a category link, by title, to the "articles" category. (defined with the "name" attribute)**

```
<txp:category link="1" title="1" wraptag="p" name="articles" />
```

[Return to tag index](#)

## Txp:category list /

### Classification

---

The `category_list` tag is a [Single Tag](#) which is used to produce a list of linked categories. Its context is page or column.

### Syntax

---

The `category_list` tag has the following syntactic structure...

```
<txp:category_list />
```

### Attributes

---

Tag will accept the following attributes (note: attributes are **case sensitive**):

`label="label text"`

Label for the top of the list. Default is unset.

`labeltag="labeltag text"`

Independent wraptag for label for the top of the list. Default is unset.

`break="html tag"`

HTML tag to be used for line breaks, without brackets. Default is `br`.

`wraptag="html tag"`

HTML tag to wrap the returned list with. Default is unset. See [Attributes Cross Reference](#).

`type="category type"`

Available values: `article`, `image`, `link`, `file`. Default is `article`.

`parent="category name"`

Will return the Parent category and its dependent categories.

`class="class name"`

CSS class attribute for wraptag, default is `category_list`.

## Examples

---

### Example 1: Display a linked category list with the label "Categories"

```
<txp:category_list label="Categories" wraptag="p" break="br" />
```

### Example 2: Display a styled category list

```
<txp:category_list break="li" wraptag="ul" />
```

Styles could go this way

```
category_list
{
  list-style-type:none;
}
```

[Return to tag index](#)

# Txp:category1 /

## Classification

---

The `<txp:category1 />` tag is a [Single Tag](#) which returns the **name** of a category to which a given article might be associated; in this case, it would be any `Cat1` category available. This tag must be used in an article [Form](#), you cannot use this tag directly in a page template (nor would it make much sense anyway).

**Note:** All categories you create in the [Organise Subtab](#) panel will equally be available in both the `Cat1` and the `Cat2` (`<txp:category2 />`) drop-down controls of the [Write Subtab](#) panel. The `Cat1` tag pulls category names for an article based on what category has been set in the `Cat1` dropdown control. (The `<txp:category2 />` tag behaves exactly the same way, but pulls category names from the `Cat2` dropdown control.)

## Syntax

---

The `category1` tag has the following syntactic structure...

```
<txp:category1 />
```

## Attributes

---

Tag will accept the following attributes (note: attributes are **case sensitive**):

`title=int`

Output category title, rather than name. Available values: 1 (yes) or 0 (no). Default is 0 (no).

`link=int`

Output category name as a hyperlink, which when clicked will search for other articles having the same category name. Available values: 1 (yes) or 0 (no). Default is 0 (no).

## Examples

---

### Example 1: Display the Cat1 category of the article being displayed in plain text

```
<txp:category1 />
```

### Example 2: Display the Cat1 category hyperlinked to search for articles of the same category

```
<txp:category1 link=1 />
```

If category1 is "General", this tag returns the following URL

<http://yoursite.com/category/general>

If you find you have URL mode or version conflicts with this, what follows will serve as well.

```
<a href="/subdirectory/index.php?c=<txp:category1 />"><txp:category1 /></a>
```

[Return to tag index](#)

# Txp:category2 /

## Classification

---

The category2 tag is a [Single Tag](#) which returns the category as defined in Cat2 of the article being displayed. Its context is an article form.

## Syntax

---

The category2 tag has the following syntactic structure...

```
<txp:category2 />
```

## Attributes

---

Tag will accept the following attributes (note: attributes are **case sensitive**):

title=int

Output category title, rather than name. Available values: 1 (yes) or 0 (no). Default is 0 (no).

link=int

Output category name as a hyperlink, which when clicked will search for other articles having the same category name. Available values: 1 (yes) or 0 (no). Default is 0 (no).

## Examples

---

### Example 1: Display the Cat2 category of the article being displayed in plain text

```
<txp:category2 />
```

### Example 2: Display the Cat2 category hyperlinked to search for articles of the same category

```
<txp:category2 link=1 />
```

If category2 is "General", this tag returns the following URL

```
http://yoursite.com/category/general
```

If you find you have URL mode or version conflicts with this, what follows will serve as well.

```
<a href="/subdirectory/index.php?c=<txp:category2 />"><txp:category2 /></a>
```

[Return to tag index](#)

## Txp:comment anchor /

### Classification

---

The `comment_anchor` tag is a [Single Tag](#). Textpattern will replace this tag with an empty anchor tag with an id attribute reflecting the comment ID. Used in a comments display form.

### Syntax

---

The `comment_anchor` tag has the following syntactic structure...

```
<txp:comment_anchor />
```

### Attributes

---

This tag has no attributes.

### Examples

---

Example 1: Tag output when part of a comment form displaying comment 000005

```
<a id="000005"></a>
```

[Return to tag index](#)

## Txp:comment email /

### Classification

---

The `comment_email` tag is a [Single Tag](#). Textpattern will replace this tag with the commenters email address, if entered at the time of posting. Used in a comments display form.

### Syntax

---

The `comment_email` tag has the following syntactic structure...

```
<txp:comment_email />
```

## Attributes

---

This tag has no attributes.

## Examples

---

### Example 1: Comments Display Form with linked email and comment id

```
<txp:comment_message /><br />
```

```
<small>&#8212 ;<a href="mailto:<txp:comment_email />">Email</a> &#160 ;&#160 ;
```

```
<txp:comment_permalink><txp:comment_id /></txp:comment_permalink></small>
```

[Return to tag index](#)

# Txp:comment\_email\_input /

This tag can be used in both Page templates and [Forms](#), which is why the link trail reflects both paths.

## Classification

---

The `comment_email_input` tag is a [Single Tag](#). Textpattern will replace this tag with a text entry field to except the users email address. Used in the comment input form.

## Syntax

---

The `comment_email_input` tag has the following syntactic structure...

```
<txp:comment_email_input />
```

## Attributes

---

This tag has no attributes.

## Examples

---

### Example 1: Comment Form

```
<table cellpadding="4" cellspacing="0" border="0">
  <tr>
    <td align="right" valign="top">
      <txp:text item="name" />
    </td>
    <td valign="top">
      <txp:comment_name_input />
    </td>
  </tr>
</table>
```

```
</td>
<td valign="top" align="left">
  <txp:comment_remember />
</td>
</tr>
<tr>
  <td align="right" valign="top">
    <txp:text item="email" />
  </td>
  <td valign="top" colspan="2">
    <txp:comment_email_input />
  </td>
</tr>
<tr>
  <td align="right" valign="top">
    http://
  </td>
  <td valign="top" colspan="2">
    <txp:comment_web_input />
  </td>
</tr>
<tr>
  <td valign="top" align="right">
    <txp:text item="message" />
  </td>
  <td valign="top" colspan="2">
    <txp:comment_message_input />
  </td>
</tr>
<tr>
  <td align="right" valign="top"> </td>
  <td valign="top" align="left">
    <txp:comments_help />
  </td>
  <td align="right" valign="top">
    <txp:comment_preview />
    <txp:comment_submit />
  </td>
</tr>
</table>
```

[Return to tag index](#)

# Txp:comment id /

## Classification

---

The `comment_id` tag is a [Single Tag](#). Textpattern will replace this tag with the comments internal id as assigned by Textpattern at the time of posting. Used in a comments display form.

## Syntax

---

The `comment_id` tag has the following syntactic structure...

```
<txp:comment_id />
```

## Attributes

---

This tag has no attributes.

## Examples

---

### Example 1: Comments Display Form with linked comment id

```
<txp:comment_message /><br />
```

```
<small>&#8212 ;<txp:comment_name /> &#160 ;&#160 ; <txp:comment_time />
&#160 ;&#160 ;
```

```
<txp:comment_permalink><txp:comment_id /></txp:comment_permalink></small>
```

[Return to tag index](#)

# Txp:comment message /

## Classification

---

The `comment_message` tag is a [Single Tag](#). Textpattern will replace this tag with the message text, or comment. Used in a comments display form.

## Syntax

---

The `comment_message` tag has the following syntactic structure...

```
<txp:comment_message />
```

## Attributes

---

This tag has no attributes.

## Examples

---

### Example 1: Comments Display Form

```
<txp:comment_message /><br />
```

```
<small>#212 ;<txp:comment_name /> #160 ;#160 ; <txp:comment_time />
#160 ;#160 ;
```

```
<txp:comment_permalink>#</txp:comment_permalink></small>
```

[Return to tag index](#)

# Txp:comment message input /

This tag can be used in both Page templates and [Forms](#), which is why the link trail reflects both paths.

## Classification

---

The `comment_message_input` tag is a [Single Tag](#). Textpattern will replace this tag with a text entry field to except the users message text. Used in the comment input form.

## Syntax

---

The `comment_message_input` tag has the following syntactic structure...

```
<txp:comment_message_input />
```

## Attributes

---

This tag has no attributes.

## Examples

---

### Example 1: Comment Form

```
<table cellpadding="4" cellspacing="0" border="0">
  <tr>
    <td align="right" valign="top">
      <txp:text item="name" />
    </td>
    <td valign="top">
      <txp:comment_name_input />
    </td>
  </tr>
</table>
```

```
</td>
<td valign="top" align="left">
  <txp:comment_remember />
</td>
</tr>
<tr>
  <td align="right" valign="top">
    <txp:text item="email" />
  </td>
  <td valign="top" colspan="2">
    <txp:comment_email_input />
  </td>
</tr>
<tr>
  <td align="right" valign="top">
    http://
  </td>
  <td valign="top" colspan="2">
    <txp:comment_web_input />
  </td>
</tr>
<tr>
  <td valign="top" align="right">
    <txp:text item="message" />
  </td>
  <td valign="top" colspan="2">
    <txp:comment_message_input />
  </td>
</tr>
<tr>
  <td align="right" valign="top"> </td>
  <td valign="top" align="left">
    <txp:comments_help />
  </td>
  <td align="right" valign="top">
    <txp:comment_preview />
    <txp:comment_submit />
  </td>
</tr>
</table>
```

[Return to tag index](#)

# Txp:comment name /

## Classification

---

The `comment_name` tag is a [Single Tag](#). Textpattern will replace this tag with a link using the commenters name as text. Commenters name and/or email address can set as a requirement in advanced preferences; the email address will be linked unless a site URL is entered. If a site URL has been entered it will be used for the link. Use of the `comment_name` or the choice of `<txp:comment_web />` or `<txp:comment_email />` in the comments display form should depend on your preference settings. Used in a comments display form.

## Syntax

---

The `comment_name` tag has the following syntactic structure...

```
<txp:comment_name />
```

## Attributes

---

This tag has no attributes.

## Examples

---

### Example 1: Comments Display Form

```
<txp:comment_message /><br />
```

```
<small>&#8212 ;<txp:comment_name /> &#160 ;&#160 ; <txp:comment_time />
&#160 ;&#160 ;
```

```
<txp:comment_permalink>#</txp:comment_permalink></small>
```

[Return to tag index](#)

# Txp:comment name input /

This tag can be used in both Page templates and [Forms](#), which is why the link trail reflects both paths.

## Classification

---

The `comment_name_input` tag is a [Single Tag](#). Textpattern will replace this tag with a text entry field to except the users name. Used in the comment input form.

## Syntax

---

The `comment_name_input` tag has the following syntactic structure...

```
<txp:comment_name_input />
```

## Attributes

---

This tag has no attributes.

## Examples

---

### Example 1: Comment Form

```
<table cellpadding="4" cellspacing="0" border="0">
  <tr>
    <td align="right" valign="top">
      <txp:text item="name" />
    </td>
    <td valign="top">
      <txp:comment_name_input />
    </td>
    <td valign="top" align="left">
      <txp:comment_remember />
    </td>
  </tr>
  <tr>
    <td align="right" valign="top">
      <txp:text item="email" />
    </td>
    <td valign="top" colspan="2">
      <txp:comment_email_input />
    </td>
  </tr>
  <tr>
    <td align="right" valign="top">
      http://
    </td>
    <td valign="top" colspan="2">
      <txp:comment_web_input />
    </td>
  </tr>
  <tr>
```

```
<td valign="top" align="right">
    <txp:text item="message" />
</td>
<td valign="top" colspan="2">
    <txp:comment_message_input />
</td>
</tr>
<tr>
    <td align="right" valign="top"> </td>
    <td valign="top" align="left">
        <txp:comments_help />
    </td>
    <td align="right" valign="top">
        <txp:comment_preview />
        <txp:comment_submit />
    </td>
</tr>
</table>
```

[Return to tag index](#)

# Txp:comment permlink (1)

## Classification

---

The `comment_permlink` tag is a [Container Tag](#) which is used to return the permanent link, of the article comment being displayed. The container tags wrap the text assigned to the link.

## Syntax

---

The `comment_permlink` tag has the following syntactic structure...

```
<txp:comment_permlink>...tag or text... </txp:permlink>
```

## Attributes

---

Tag will accept the following attributes (note: attributes are **case sensitive**):

`anchor="value"`

Whether to apply the comment's id to the hyperlink tag (id attribute), setting this comment permanent link as the comment page anchor. Available values: 0, 1. Default is 0.

## Examples

---

### Example 1: Display a link for the article comment being displayed

```
<txp:comment_permalink>#</txp:comment_permalink>
```

[Return to tag index](#)

# Txp:comment\_permalink (2)

## Classification

---

The `comment_permalink` tag is a [Container Tag](#) which is used to return the permanent link, of the article comment being displayed. The container tags wrap the text assigned to the link.

## Syntax

---

The `comment_permalink` tag has the following syntactic structure...

```
<txp:comment_permalink>...tag or text... </txp:permalink>
```

## Attributes

---

Tag will accept the following attributes (note: attributes are **case sensitive**):

`anchor="value"`

Whether to apply the comment's id to the hyperlink tag (id attribute), setting this comment permanent link as the comment page anchor. Available values: 0, 1. Default is 0.

## Examples

---

### Example 1: Display a link for the article comment being displayed

```
<txp:comment_permalink>#</txp:comment_permalink>
```

[Return to tag index](#)

# Txp:comment\_preview /

This tag can be used in both Page templates and [Forms](#), which is why the link trail reflects both paths.

## Classification

---

The `comment_preview` tag is a [Single Tag](#). Textpattern will replace this tag with a Preview button the user can use to preview the comment text. Used in the comment input form.

## Syntax

---

The `comment_preview` tag has the following syntactic structure...

```
<txp:comment_preview />
```

## Attributes

---

This tag has no attributes.

## Examples

---

### Example 1: Comment Form

```
<table cellpadding="4" cellspacing="0" border="0">
  <tr>
    <td align="right" valign="top">
      <txp:text item="name" />
    </td>
    <td valign="top">
      <txp:comment_name_input />
    </td>
    <td valign="top" align="left">
      <txp:comment_remember />
    </td>
  </tr>
  <tr>
    <td align="right" valign="top">
      <txp:text item="email" />
    </td>
    <td valign="top" colspan="2">
      <txp:comment_email_input />
    </td>
  </tr>
  <tr>
    <td align="right" valign="top">
      http://
    </td>
    <td valign="top" colspan="2">
      <txp:comment_web_input />
    </td>
  </tr>
  <tr>
```

```
<td valign="top" align="right">
    <txp:text item="message" />
</td>
<td valign="top" colspan="2">
    <txp:comment_message_input />
</td>
</tr>
<tr>
    <td align="right" valign="top"> </td>
    <td valign="top" align="left">
        <txp:comments_help />
    </td>
    <td align="right" valign="top">
        <txp:comment_preview />
        <txp:comment_submit />
    </td>
</tr>
</table>
```

[Return to tag index](#)

# Txp:comment remember /

This tag can be used in both Page templates and [Forms](#), which is why the link trail reflects both paths.

## Classification

---

The `comment_remember` tag is a [Single Tag](#). Textpattern will replace this tag with a check box input field. If checked the users details are remembered by the system the next time they open a comment form. Used in the comment input form.

## Syntax

---

The `comment_remember` tag has the following syntactic structure...

```
<txp:comment_remember />
```

## Attributes

---

This tag has no attributes.

## Examples

---

### Example 1: Comment Form

```
<table cellpadding="4" cellspacing="0" border="0">
  <tr>
    <td align="right" valign="top">
      <txp:text item="name" />
    </td>
    <td valign="top">
      <txp:comment_name_input />
    </td>
    <td valign="top" align="left">
      <txp:comment_remember />
    </td>
  </tr>
  <tr>
    <td align="right" valign="top">
      <txp:text item="email" />
    </td>
    <td valign="top" colspan="2">
      <txp:comment_email_input />
    </td>
  </tr>
  <tr>
    <td align="right" valign="top">
      http://
    </td>
    <td valign="top" colspan="2">
      <txp:comment_web_input />
    </td>
  </tr>
  <tr>
    <td valign="top" align="right">
      <txp:text item="message" />
    </td>
    <td valign="top" colspan="2">
      <txp:comment_message_input />
    </td>
  </tr>
  <tr>
```

```
<td align="right" valign="top"> </td>
<td valign="top" align="left">
  <txp:comments_help />
</td>
<td align="right" valign="top">
  <txp:comment_preview />
  <txp:comment_submit />
</td>
</tr>
</table>
```

[Return to tag index](#)

## Txp:comment submit /

This tag can be used in both Page templates and [Forms](#), which is why the link trail reflects both paths.

### Classification

---

The `comment_submit` tag is a [Single Tag](#). Textpattern will replace this tag with a Submit button. Clicking the Submit button writes the comment information to the database. Used in the comment input form.

### Syntax

---

The `comment_submit` tag has the following syntactic structure...

```
<txp:comment_submit />
```

### Attributes

---

This tag has no attributes.

### Examples

---

#### Example 1: Comment Form

```
<table cellpadding="4" cellspacing="0" border="0">
  <tr>
    <td align="right" valign="top">
      <txp:text item="name" />
    </td>
    <td valign="top">
      <txp:comment_name_input />
    </td>
  </tr>
</table>
```

```
<td valign="top" align="left">
    <txp:comment_remember />
</td>
</tr>
<tr>
    <td align="right" valign="top">
        <txp:text item="email" />
    </td>
    <td valign="top" colspan="2">
        <txp:comment_email_input />
    </td>
</tr>
<tr>
    <td align="right" valign="top">
        http://
    </td>
    <td valign="top" colspan="2">
        <txp:comment_web_input />
    </td>
</tr>
<tr>
    <td valign="top" align="right">
        <txp:text item="message" />
    </td>
    <td valign="top" colspan="2">
        <txp:comment_message_input />
    </td>
</tr>
<tr>
    <td align="right" valign="top"> </td>
    <td valign="top" align="left">
        <txp:comments_help />
    </td>
    <td align="right" valign="top">
        <txp:comment_preview />
        <txp:comment_submit />
    </td>
</tr>
</table>
```

[Return to tag index](#)

# Txp:comment time /

## Classification

---

The `comment_time` tag is a [Single Tag](#). Textpattern will replace this tag with the time and date the comment was submitted. Used in a comments display [form](#).

## Syntax

---

The `comment_time` tag has the following syntactic structure...

```
<txp:comment_time />
```

## Attributes

---

This tag has no attributes.

## Examples

---

### Example 1: Comments Display Form

```
<txp:comment_message /><br />
```

```
<small>&#8212 ;<txp:comment_name /> &#160 ;&#160 ; <txp:comment_time />
&#160 ;&#160 ;
```

```
<txp:comment_permalink>#</txp:comment_permalink></small>
```

[Return to tag index](#)

# Txp:comment web /

## Classification

---

The `comment_web` tag is a [Single Tag](#). Textpattern will replace this tag with the commenters web address, if entered at the time of posting. Used in a comments display [form](#).

## Syntax

---

The `comment_web` tag has the following syntactic structure...

```
<txp:comment_web />
```

## Attributes

---

This tag has no attributes.

## Examples

---

### Example 1: Comments Display Form with linked website and comment id

```
<txp:comment_message /><br />
```

```
<small>&#8212 ;<a href="http://<txp:comment_web />"><txp:comment_web /></a>
&#160 ;&#160 ;
```

```
<txp:comment_permalink><txp:comment_id /></txp:comment_permalink></small>
```

[Return to tag index](#)

# Txp:comment web input /

This tag can be used in both Page templates and [Forms](#), which is why the link trail reflects both paths.

## Classification

---

The `comment_web_input` tag is a [Single Tag](#). Textpattern will replace this tag with a text entry field to except the commenters domain name. Function assumes `http://` for all URLs.

## Syntax

---

The `comment_web_input` tag has the following syntactic structure...

```
<txp:comment_web_input />
```

## Attributes

---

This tag has no attributes.

## Examples

---

### Example 1: Comment Form

```
<table cellpadding="4" cellspacing="0" border="0">
  <tr>
    <td align="right" valign="top">
      <txp:text item="name" />
    </td>
    <td valign="top">
      <txp:comment_name_input />
    </td>
    <td valign="top" align="left">
      <txp:comment_remember />
    </td>
  </tr>
</table>
```

```
</td>
</tr>
<tr>
  <td align="right" valign="top">
    <txp:text item="email" />
  </td>
  <td valign="top" colspan="2">
    <txp:comment_email_input />
  </td>
</tr>
<tr>
  <td align="right" valign="top">
    http://
  </td>
  <td valign="top" colspan="2">
    <txp:comment_web_input />
  </td>
</tr>
<tr>
  <td valign="top" align="right">
    <txp:text item="message" />
  </td>
  <td valign="top" colspan="2">
    <txp:comment_message_input />
  </td>
</tr>
<tr>
  <td align="right" valign="top"> </td>
  <td valign="top" align="left">
    <txp:comments_help />
  </td>
  <td align="right" valign="top">
    <txp:comment_preview />
    <txp:comment_submit />
  </td>
</tr>
</table>
```

[Return to tag index](#)

# Txp:comments /

## Classification

---

The `comments` tag is a [Single Tag](#). Textpattern will replace this tag with the comments associated with a particular article. Comments will be displayed for the present individual article as a default, or to the article set by the "id" attribute.

## Syntax

---

The `comments` tag has the following syntactic structure...

```
<txp:comments />
```

## Attributes

---

Tag will accept the following attributes (note: attributes are **case sensitive**):

`id="article id"`

The `id#` of the article you are adding comments to. Default is unset.

`class="value"`

See [Attributes Cross Reference: class](#). Default is `comments`.

`form="value"`

See [Attributes Cross Reference: form](#). Default is `comments`.

`wraptag="value"`

See [Attributes Cross Reference: wraptag](#). Default is unset.

`break="value"`

See [Attributes Cross Reference: break](#). Default is unset.

`breakclass="value"`

The CSS class to apply to the break tag. Default is unset.

## Examples

---

Example 1: Display comments, and give user's an indication of `Comments` status.

Comments for articles can be turned off or on at the authors discretion for any article that is published; by using the following scheme in an article form, you can still have the on/off control over comments while still giving users indication of comment status.

```
<txp:comments />
```

```
<txp:if_comments_allowed>
```

```
  <txp:comments_form />
```

```
</txp:if_comments_allowed>
```

```
<txp:if_comments_disallowed>
```

```
  <p>Comments are turned off for this article.</p>
```

```
</txp:if_comments_disallowed>
```

**Example 2: Column list for displaying id numbers for comments for article id 3, and a comment input form, if comments are currently allowed on article id 3.**

Tags

```
<txp:if_comments_allowed id="3">
<txp:comments id="3" form="lineitem" break="li" wraptag="ul"
breakclass="special" />
<txp:comments_form id="3" />
</txp:if_comments_allowed>
```

Form (lineitem) Type(comment)

```
<small><txp:comment_id /></small>
```

Styles could go this way

```
special
{
  display:list-item;
  list-style-type:none;
}
```

note: [<txp:comment\\_permalink>](#) will only link content correctly with an individual article displayed.

[Return to tag index](#)

## Txp:comments count /

### Classification

---

The `comments_count` tag is a [Single Tag](#) Textpattern will replace this tag with the number of comments associated with a particular article. Though `comments_count` can be used independently, it is also called by [<txp:comments\\_invite />](#) to append the comments count to the `comments_invite` link. Used in an article [form](#).

### Syntax

---

The `comments_count` tag has the following syntactic structure...

```
<txp:comments_count />
```

### Attributes

---

This tag has no attributes.

### Examples

---

**Example 1: Display comments invitation and comment count if any comments are associated with the current article.**

```
<txp:if_comments><p><txp:comments_invite /></p></txp:if_comments>
```

[Return to tag index](#)

# Txp:comments form /

## Classification

---

The `comments_form` tag is a [Single Tag](#). Textpattern will replace this tag with a comment form. Comments will be attached to present individual article as a default, or to the article set by the "id" attribute.

## Syntax

---

The `comments_form` tag has the following syntactic structure...

```
<txp:comments_form />
```

## Attributes

---

Tag will accept the following attributes (note: attributes are **case-sensitive**):

`id="integer"`

The `id#` assigned to the article you are adding comments to. Default is current article `id#`.

`form="form name"`

Use specified form. Default is `comment_form`. See [Form](#).

`wraptag="html tag"`

HTML tag to wrap the form in. Default is unset. See [Attributes Cross Reference](#).

`class="CSS class name"`

Default is `comments_form`.

## Examples

---

Example 1: Giving user's indication of `Comments` status.

Comments for articles can be turned off or on at the authors discretion for any article that is published; by using the following scheme in an article form, you can still have the on/off control over comments while still giving users indication of comment status.

```
<txp:if_comments_allowed>
  <txp:comments_form />
</txp:if_comments_allowed>
```

```
<txp:if_comments_disallowed>
  <p>Comments are turned off for this article.</p>
</txp:if_comments_disallowed>
```

**Example 2: Column list for displaying id numbers for comments for article id 3, and a comment input form, if comments are currently allowed on article id 3.**

Tags

```
<txp:if_comments_allowed id="3">
<txp:comments id="3" form="lineitem" break="li" wraptag="ul"
breakclass="special" />
<txp:comments_form id="3" />
</txp:if_comments_allowed>
```

Form (lineitem) Type(comment)

```
<small><txp:comment_id /></small>
```

Styles could go this way

```
special
{
  display:list-item;
  list-style-type:none;
}
```

Note: `<txp:comment_permalink>` will only link content correctly with an individual article displayed.

[Return to tag index](#)

## Txp:comments help /

This tag can be used in both Page templates and [Forms](#), which is why the link trail reflects both paths.

### Classification

---

The `comment_help` tag is a [Single Tag](#). Textpattern will replace this tag with a Textpattern help link.

### Syntax

---

The `comment_help` tag has the following syntactic structure...

```
<txp:comment_help />
```

### Attributes

---

This tag has no attributes.

### Examples

---

#### Example 1: Comment Form

```
<table cellpadding="4" cellspacing="0" border="0">
```

```
<tr>
  <td align="right" valign="top">
    <txp:text item="name" />
  </td>
<td valign="top">
  <txp:comment_name_input />
</td>
<td valign="top" align="left">
  <txp:comment_remember />
</td>
</tr>
<tr>
<td align="right" valign="top">
  <txp:text item="email" />
</td>
<td valign="top" colspan="2">
  <txp:comment_email_input />
</td>
</tr>
<tr>
<td align="right" valign="top">
  http://
</td>
<td valign="top" colspan="2">
  <txp:comment_web_input />
</td>
</tr>
<tr>
<td valign="top" align="right">
  <txp:text item="message" />
</td>
<td valign="top" colspan="2">
  <txp:comment_message_input />
</td>
</tr>
<tr>
<td align="right" valign="top"> </td>
<td valign="top" align="left">
  <txp:comments_help />
</td>
```

```
<td align="right" valign="top">
    <txp:comment_preview />
    <txp:comment_submit />
</td>
</tr>
</table>
```

[Return to tag index](#)

## Txp:comments invite /

This tag can be used in both Page templates and [Forms](#), which is why this link trail is indicative of both paths.

### Classification

---

The `comments_invite` tag is a [Single Tag](#) Textpattern will replace this tag with a link to an article comment form. Text used for the link will be the taken from the invitation field on the "write" screen. Used in an article form.

### Syntax

---

The `comments_invite` tag has the following syntactic structure...

```
<txp:comments_invite />
```

### Attributes

---

Tag will accept the following attributes (note: attributes are **case sensitive**):

`showcount="value"`

Whether to display comment count. Available values: 1 (yes), 0 (no). Default is 1 (no).

`textonly="value"`

Whether to display invite as text, rather than a hyperlink. Available values: 1 (yes), 0 (no). Default is 0.

`showalways="value"`

Whether to display invite on individual article page. Available values: 1 (yes), 0 (no). Default is 0.

`class="class name"`

CSS class, Default is `comments_invite`.

Info @ [Developers Weblog](#)

## Examples

---

**Example 1: Display comments invitation and comment count if any comments are associated with the current article.**

```
<txp:if_comments><p><txp:comments_invite /></p></txp:if_comments>
```

[Return to tag index](#)

# Txp:custom field /

## Classification

---

The `custom_field` tag is a [Single Tag](#) which is used to display the contents of a custom field.

Custom fields are useful if you publish content that has a consistent structure. They are defined in the "Advanced preferences" tab, under the "Custom" heading. You may have up to and including 10 fields. With a default Textpattern installation, you will just see two custom fields with the names "custom1" and "custom2", which you may rename or remove. Custom names may include letters (uppercase or lowercase), numbers and under\_scores, but no spaces or other special characters. To remove a custom field, simply clear its name. **Don't forget to save your changes.**

Once defined, like keywords, you may set values for your custom fields for each post (under "advanced options" in the "write" tab), and may display custom field information in any "article" type [form](#).

## Syntax

---

The syntax is as follows:

```
<txp:custom_field name="FieldName" />
```

This tag will be replaced with anything you write into the custom field named "FieldName".

## Attributes

---

```
name="fieldname"
```

Display custom field with the name `fieldname`.

## Examples

---

### Example 1: Book Reviews

You might, for example, publish book reviews for which you add the author, the title of the book, the publishing company and the year of publication. For this you could use the following setup:

```
Custom field 1 name: Author
```

```
Custom field 2 name: Title
```

```
Custom field 3 name: Publisher
```

```
Custom field 4 name: Year
```

article form:

```
<p><txp:custom_field name="Author" />: <txp:custom_field name="Title" /><br />
Published by <txp:custom_field name="Publisher" /> in <txp:custom_field
name="Year" />.</p>
```

html returned could be:

```
<p>J.R.R. Tolkien: The Lord of the Rings<br />
Published by HarperCollins in 2004.</p>
```

### Example 2: Linklog

[\[\[1\]\]](#): I use custom fields with article\_custom to create a linklog. This works well with [\[variation of Sencer's txp bookmarklet\]](#). 'scuse the dodgy CSS.

Custom field 1 name: Link

article form:

```
<div class="linklog-entry">
<div style="float: left;"><a href="<txp:custom_field name="Link" />"><txp:title
/></a></div>
<div style="float: right;"><txp:posted format="%d %d %Y" /></div><br>
<txp:excerpt />
</div>
```

html returned could be:

```
<div class="linklog-entry">
<div style="float: left;"><a
href="http://textpattern.com/">Textpattern</a></div>
<div style="float: right;"><txp:posted format="08 Aug 2005" /></div><br>
Textpattern is awesome.
</div>
```

## Closing Notes

---

Custom fields can save a lot of thinking and work and to make them even more powerful, there is a set of conditional tags to go with custom fields: the [txp:if\\_custom\\_field](#) tag. More explanation and examples on that page.

Related info: [<txp:if\\_custom\\_field>](#)

[Return to tag index](#)

# Txp:css /

## Classification

---

The `css` tag is a [Single Tag](#). This tag is used in the header of your document, where it will output the URL of the style sheet assigned in the [Sections Subtab](#).

## Syntax

---

```
<txp:css />
```

## Attributes

---

Tag will accept the following attributes (note: attributes are **case sensitive**):

`n="style name"`

Choose the name of a specific style. The style can be named and edited in the [Style Subtab](#).)

## Examples

---

### Example 1: Output the URL to the section's default style sheet

```
<head>
<!-- additional tags... -->
<link rel="stylesheet" href="<txp:css />" type="text/css" />
<!-- more tags... -->
</head>
```

### Example 2: Output the URL to a named style sheet

```
<head>
<!-- additional tags... -->
<link rel="stylesheet" href="<txp:css n="stylename" />" type="text/css" />
<!-- more tags... -->
</head>
```

[Return to tag index](#)

# Txp:die /

## Classification

---

The die tag is a [Single Tag](#). Textpattern will terminate normal page rendition when it encounters this tag and return the given `status` to the user agent (browser, search engine crawler, feed aggregator). An error page will also be returned to the user agent.

The `status` can be displayed by the [<txp:error\\_status />](#) tag. A textual message can be associated with the error status and retrieved with the [<txp:error\\_message />](#) tag. See also: [Custom Error Pages](#).

## Syntax

---

The die tag has the following syntactic structure...

```
<txp:die />
```

## Attributes

---

Tag will accept the following attributes (note: attributes are **case sensitive**):

`msg="message"`

Textual representation of the error condition.

`status="number"`

Numerical representation of the error condition. Default is "503".

## Examples

---

### Example 1: Force "Not found" error

```
<txp:die status="404" />
```

[Return to tag index](#)

# Txp:else /

## Classification

---

The `else` tag is a [Single Tag](#).

The **else** tag is used between conditional tags to provide the means to assign default, or alternative behavior, when the condition set in the conditional tag is not met.

## Syntax

---

The `else` tag has the following syntactic structure...

```
<txp:if_conditional_tag />
...Content if true...
  <txp:else />
...Content if not true...
</txp:if_conditional_tag />
```

## Attributes

---

This tag has no attributes.

## Examples

---

### Example 1: Display the section name when no excerpt is associated with an article, display the excerpt if available.

```
<txp:if_excerpt>
And Furthermore · <txp:excerpt />
<txp:else />
<txp:section link="y" />
```

</txp:if\_excerpt>

[Return to tag index](#)

## Txp:email /

### Classification

---

The `email` tag is a [Single Tag](#) Textpattern will replace this tag with a `mailto:` email link, according to the attributes set.

### Syntax

---

The `email` tag has the following syntactic structure...

```
<txp:email />
```

### Attributes

---

Tag will accept the following attributes (note: attributes are **case sensitive**):

```
email="email address"
```

Any valid email.

```
linktext="text"
```

Displayed link text.

```
title="text"
```

href title attribute.

### Examples

---

#### Example 1: Display the section name when included in an article form

```
<txp:email email="rsilletti@gmail.com" linktext="Contact" title="Send an Email" />
```

[Return to tag index](#)

## Txp:excerpt /

### Classification

---

The `excerpt` tag is a [Single Tag](#) which is used to return the excerpt text, if any, associated with the article being displayed.

This tag is used in an article [form](#).

### Syntax

---

The `excerpt` tag has the following syntactic structure...

```
<txp:excerpt />
```

## Attributes

---

This tag has no attributes.

## Examples

---

### Example 1: Display the excerpt text, or a default link, as part of an article form using conditional tags

```
<txp:if_excerpt>
<txp:excerpt />
<txp:else />
Section <a href="/subdirname/index.php?s=<txp:section />"><txp:section /></a>
</txp:if_excerpt>
```

This example will display the section name linked as **/"subdirectory if any"/index.php?s="section name"** to provide a link to return the articles from the same section as the article being displayed.

Related info: <[txp:if\\_excerpt](#)>

[Return to tag index](#)

# Txp:feed link /

## Classification

---

The `feed_link` tag is a [Single Tag](#) Textpattern will replace this tag with an XHTML link to the sites articles RSS feed. Use in Page or Column.

## Syntax

---

The `feed_link` tag has the following syntactic structure...

```
<txp:feed_link />
```

## Attributes

---

Tag will accept the following attributes (note: attributes are **case sensitive**):

`label="label text"`

Label for feed link.

`break="tag text"`

HTML line breaks.

`wraptag="wraptag text"`

Character to be used as the wraptag without brackets.

`category="category name"`

Selected from available link categories.

limit="integer"

Number of links shown in feed.

section="section name"

Selected from available sections.

title="title text"

XML feed title.

flavor="rss or atom"

## Examples

---

**Example 1: Display an article feed link called "XML" with a flavor attribute set to "rss" with category and section criteria.**

```
<txp:feed_link label="XML" category="General" section="about" flavor="rss"
wraptag="p" />
```

[Return to tag index](#)

# Txp:file download /

## Classification

---

The `file_download` tag is a [Single Tag](#). Textpattern will replace this tag with a file download [form](#). Page Column.

## Syntax

---

The `file_download` tag has the following syntactic structure...

```
<txp:file_download />
```

## Attributes

---

Tag will accept the following attributes (note: attributes are **case-sensitive**):

form="form name"

Use specified form. Default is `files`. See [Form](#).

id="integer"

File id of the file to link to. Default is unset, nothing is returned.

filename="filename"

Filename of the file to link to. Default is unset, nothing is returned.

## Examples

---

**Example 1: Display a download form for the file to be downloaded**

```
<txp:file_download form="files" id="1" />
```

### Default "files" form

```
<txp:text item="file" />:
<txp:file_download_link>
<txp:file_download_name /> [<txp:file_download_size format="auto" decimals="2"
/>]
</txp:file_download_link>
<br />
<txp:text item="category" />: <txp:file_download_category /><br />
<txp:text item="download" />: <txp:file_download_downloads />
```

[Return to tag index](#)

# Txp:file download category /

## Classification

---

The `file_download_category` tag is a [Single Tag](#). Textpattern will replace this tag with the category of the file to download. Should be used in a download [form](#).

## Syntax

---

The `file_download_category` tag has the following syntactic structure...

```
<txp:file_download_category />
```

## Attributes

---

This tag has no attributes.

## Examples

---

### Example 1: Display a category name following "category:"

```
<txp:text item="category" />: <txp:file_download_category />
```

### Default "files" form

```
<txp:text item="file" />:
<txp:file_download_link>
<txp:file_download_name /> [<txp:file_download_size format="auto" decimals="2"
/>]
</txp:file_download_link>
<br />
<txp:text item="category" />: <txp:file_download_category /><br />
<txp:text item="download" />: <txp:file_download_downloads />
```

[Return to tag index](#)

# Txp:file download created /

## Classification

---

The `file_download_created` tag is a [Single Tag](#). Textpattern will replace this tag with the upload date of the file to download. Should be used in a download [form](#).

## Syntax

---

The `file_download_created` tag has the following syntactic structure...

```
<txp:file_download_created />
```

## Attributes

---

Tag will accept the following attributes (note: attributes are **case sensitive**):

```
format="character format"
```

character formatting usable by `strftime()`, uses the Archive date format as the default.

## Examples

---

### Example 1: Display formatted file upload date

```
<txp:file_download_created format="%c" />
```

### Default "files" form

```
<txp:text item="file" />:
<txp:file_download_link>
<txp:file_download_name /> [<txp:file_download_size format="auto" decimals="2"
/>]
</txp:file_download_link>
<br />
<txp:text item="category" />: <txp:file_download_category /><br />
<txp:text item="download" />: <txp:file_download_downloads />
```

[Return to tag index](#)

# Txp:file download description /

## Classification

---

The `file_download_description` tag is a [Single Tag](#). Textpattern will replace this tag with the description of the file to download, as defined when uploaded. Should be used in a download [form](#).

## Syntax

---

The `file_download_description` tag has the following syntactic structure...

```
<txp:file_download_description />
```

## Attributes

---

This tag has no attributes.

## Examples

---

**Example 1: Display a file s description following "description:"**

```
<txp:text item="description" />: <txp:file_download_description />
```

[Return to tag index](#)

# Txp:file download downloads /

## Classification

---

The `file_download_downloads` tag is a [Single Tag](#). Textpattern will replace this tag with the number of times a file has been downloaded. Should be used in a download [form](#).

## Syntax

---

The `file_download_downloads` tag has the following syntactic structure...

```
<txp:file_download_downloads />
```

## Attributes

---

This tag has no attributes.

## Examples

---

**Example 1: Display the number of downloads following "downloads:"**

```
<txp:text item="downloads" />: <txp:file_download_downloads />
```

### Default "files" form

```
<txp:text item="file" />:
<txp:file_download_link>
<txp:file_download_name /> [<txp:file_download_size format="auto" decimals="2" />]
</txp:file_download_link>
<br />
```

```
<txp:text item="category" />: <txp:file_download_category /><br />
<txp:text item="download" />: <txp:file_download_downloads />
```

[Return to tag index](#)

## Txp:file download id /

### Classification

---

The `file_download_id` tag is a [Single Tag](#). Textpattern will replace this tag with the internal ID number of the file to be downloaded. Should be used in a download [form](#).

### Syntax

---

The `file_download_id` tag has the following syntactic structure...

```
<txp:file_download_id />
```

### Attributes

---

This tag has no attributes.

### Examples

---

#### Example 1: Display a file id following "File number:"

```
<txp:text item="File number" />: <txp:file_download_id />
```

#### Default "files" form

```
<txp:text item="file" />:
<txp:file_download_link>
<txp:file_download_name /> [<txp:file_download_size format="auto" decimals="2" />]
</txp:file_download_link>
<br />
<txp:text item="category" />: <txp:file_download_category /><br />
<txp:text item="download" />: <txp:file_download_downloads />
```

[Return to tag index](#)

## Txp:file download link (1)

### Classification

---

The `file_download_link` tag is a [Container Tag](#), Textpattern will replace this tag with a download link. The container tags wrap the text or tag assigned to the link.

## Syntax

---

The `file_download_link` tag has the following syntactic structure...

```
<txp:file_download_link> ...your content here... </txp:file_download_link>
```

## Attributes

---

Tag will accept the following attributes (note: attributes are **case sensitive**):

`filename="Text "`

name of file to download

`id="Integer"`

id of the file to download

Note: "id" takes precedence over "filename", if neither is defined nothing is returned.

## Examples

---

### Example 1: File download link id #1

```
<txp:file_download_link id="1">
<txp:file_download_name /> [<txp:file_download_size format="auto" decimals="2"
/>]
</txp:file_download_link>
```

### Default "files" form

```
<txp:text item="file" />:
<txp:file_download_link>
<txp:file_download_name /> [<txp:file_download_size format="auto" decimals="2"
/>]
</txp:file_download_link>
<br />
<txp:text item="category" />: <txp:file_download_category /><br />
<txp:text item="download" />: <txp:file_download_downloads />
```

[Return to tag index](#)

# Txp:file download link (2)

## Classification

---

The `file_download_link` tag is a [Container Tag](#), Textpattern will replace this tag with a download link. The container tags wrap the text or tag assigned to the link.

## Syntax

---

The `file_download_link` tag has the following syntactic structure...

```
<txp:file_download_link> ...your content here... </txp:file_download_link>
```

## Attributes

---

Tag will accept the following attributes (note: attributes are **case sensitive**):

filename="Text "

name of file to download

id="Integer"

id of the file to download

Note: "id" takes precedence over "filename", if neither is defined nothing is returned.

## Examples

---

### Example 1: File download link id #1

```
<txp:file_download_link id="1">
<txp:file_download_name /> [<txp:file_download_size format="auto" decimals="2"
/>]
</txp:file_download_link>
```

### Default "files" form

```
<txp:text item="file" />:
<txp:file_download_link>
<txp:file_download_name /> [<txp:file_download_size format="auto" decimals="2"
/>]
</txp:file_download_link>
<br />
<txp:text item="category" />: <txp:file_download_category /><br />
<txp:text item="download" />: <txp:file_download_downloads />
```

[Return to tag index](#)

# Txp:file download list /

## Classification

---

The `file_download_list` tag is a [Single Tag](#) which is used to produce a list of download links according to the attributes set in the tag criteria.

## Syntax

---

The `file_download_list` tag has the following syntactic structure...

```
<txp:file_download_list />
```

## Attributes

---

Tag will accept the following attributes (note: attributes are **case-sensitive**):

`category="category name"`

Restrict to files from specified category. Default is unset (unrestricted).

`limit="integer"`

The number of articles to display. Default is 10.

`sort="sort by"`

Sort the list by specified option. Available values: `id`, `filename`, `category`, `description`, `downloads` or `rand()`. Default is `filename`. See [Attributes Cross Reference](#).

`form="form name"`

Use specified form. Default is `files`. See [Form](#).

`label="text label"`

Label for the top of the list. See [Attributes Cross Reference](#).

`labeltag="labeltag text"`

Independent wraptag for label for the top of the list. Default is empty.

`wraptag="html tag"`

HTML tag to wrap the list in. Default is unset. See [Attributes Cross Reference](#).

`class="CSS class name"`

Default is `file_download_list`.

`break="html tag"`

Default is `br`. See [Attributes Cross Reference](#).

## Examples

---

### Example 1: Display a styled download link list of ten downloads

```
<txp:file_download_list limit="10" break="li" wraptag="ul" sort="downloads asc" />
```

Styles could go this way

```
file_download_list
{
  list-style-type:none;
}
```

[Return to tag index](#)

# Txp:file download modified /

## Classification

---

The `file_download_modified` tag is a [Single Tag](#). Textpattern will replace this tag with the last modified date of the file to download. Should be used in a download [form](#).

## Syntax

---

The `file_download_modified` tag has the following syntactic structure...

```
<txp:file_download_modified />
```

## Attributes

---

Tag will accept the following attributes (note: attributes are **case sensitive**):

```
format="character format"
```

character formatting usable by `strftime()`, uses the Archive date format as the default.

## Examples

---

### Example 1: Display formatted file modified date.

```
<txp:file_download_modified format="%c" />
```

### Default "files" form

```
<txp:text item="file" />:
<txp:file_download_link>
<txp:file_download_name /> [<txp:file_download_size format="auto" decimals="2"
/>]
</txp:file_download_link>
<br />
<txp:text item="category" />: <txp:file_download_category /><br />
<txp:text item="download" />: <txp:file_download_downloads />
```

[Return to tag index](#)

# Txp:file download name /

## Classification

---

The `file_download_name` tag is a [Single Tag](#). Textpattern will replace this tag with the name of the file to download. Should be used in a download [form](#) or with download link tags.

## Syntax

---

The `file_download_name` tag has the following syntactic structure...

```
<txp:file_download_name />
```

## Attributes

---

This tag has no attributes.

## Examples

---

### Example 1: Display the name of a file, linked to download

```
<txp:file_download_link filename="info">
<txp:file_download_name /> [<txp:file_download_size format="auto" decimals="2"
/>]
</txp:file_download_link>
```

### Default "files" form

```
<txp:text item="file" />:
<txp:file_download_link>
<txp:file_download_name /> [<txp:file_download_size format="auto" decimals="2"
/>]
</txp:file_download_link>
<br />
<txp:text item="category" />: <txp:file_download_category /><br />
<txp:text item="download" />: <txp:file_download_downloads />
```

Note: "id" takes precedence over "filename", if neither is defined nothing is returned. Ref: [<txp:file\\_download\\_link>](#)

Retrieved from "[http://textpattern.net/wiki/index.php?title=Txp:file\\_download\\_name/](http://textpattern.net/wiki/index.php?title=Txp:file_download_name/)"

[Return to tag index](#)

# Txp:file download size /

## Classification

---

The `file_download_size` tag is a [Single Tag](#). Textpattern will replace this tag with the formatted file size of the file to be downloaded. Should be used in a download [form](#).

## Syntax

---

The `file_download_size` tag has the following syntactic structure...

```
<txp:file_download_size />
```

## Attributes

---

Tag will accept the following attributes (note: attributes are **case sensitive**):

format="numbering style"

select from dropdown options in Tag Builder Asst

decimals="number of decimal places"

## Examples

---

### Example 1: Display formatted file size.

```
<txp:file_download_size format="kb" decimals="2" />
```

### Default "files" form

```
<txp:text item="file" />:
```

```
<txp:file_download_link>
```

```
<txp:file_download_name /> [<txp:file_download_size format="auto" decimals="2" />]
```

```
</txp:file_download_link>
```

```
<br />
```

```
<txp:text item="category" />: <txp:file_download_category /><br />
```

```
<txp:text item="download" />: <txp:file_download_downloads />
```

[Return to tag index](#)

# Txp:if article category

## Classification

---

The `if_article_category` tag is a [Conditional Tag](#), and therefore a [Container Tag](#), as all conditional tags are container tags. This tag checks the category name associated with a particular article and its category number (Category1 or Category2) if the category name is matched to the proper category level (cat1 or cat2), it will execute the contained statements. Should be used in an Article form (for Page or Column, use [Txp:if\\_category](#))

## Syntax

---

The `if_article_category` tag has the following syntactic structure...

```
<txp:if_article_category> ...your content here...<txp:else />...default option... </txp:if_article_category>
```

## Attributes

---

Tag will accept the following attributes (note: attributes are **case sensitive**):

name="Text "

Category Name

```
number="1 or 2"
```

```
match Category in Cat1 or Cat2
```

## Examples

---

### Example 1: Display category name if "Prose" as Cat1 is assigned to article

```
<txp:if_article_category name="Prose" number="1">
<p><txp:category1 /></p>
</txp:if_article_category>
```

### Example 2: Display welcome text if category and category level match, show default link otherwise

```
<txp:if_article_category name="Prose" number="1">
<p>Fun With Prose</p>
<txp:else />
<p><a href://" .index.php">Home</a></p>
</txp:if_article_category>
```

Example 3: Display a list of links to articles of the same category as the displayed category as an addendum to the article [form](#)

Place what follows in an article [form](#), a set of conditionals for each category you want to look for.

```
<txp:if_article_category name="yourcategory" number="1">
<ul>
<txp:article_custom form="sub" category="yourcategory" sortby="Posted"
sortdir="asc" />
</ul>
</txp:if_article_category>
```

And then your article form (in this case called 'sub') for listing links to other articles can be something like this:

```
<li><txp:permlink><txp:title /></txp:permlink></li>
```

[Return to tag index](#)

# Txp:if article list

## Classification

---

The `if_article_list` tag is a [Conditional Tag](#), and therefore a [Container Tag](#), as all conditional tags are container tags. The tag will execute the contained statement if an article list is being displayed.

## Syntax

---

The `if_article_list` tag has the following syntactic structure...

```
<txp:if_article_list> ...your content here...<txp:else />...default option...  
</txp:if_article_list>
```

## Attributes

---

This tag has no attributes.

## Examples

---

**Example 1: Excerpt from RC5 default page template, selects next-prev or older-newer navigation method.**

```
<txp:article />  
<txp:if_individual_article>  
<p>  
<txp:link_to_prev><txp:prev_title /></txp:link_to_prev>  
<txp:link_to_next><txp:next_title /></txp:link_to_next>  
</p>  
</txp:if_individual_article>  
<txp:if_article_list>  
<p>  
<txp:older>Previous</txp:older>  
<txp:newer>Next</txp:newer>  
</p>  
</txp:if_article_list>
```

Example 2: Use `<txp:else />` to display a logo when not displaying an article list, and the Site's Name when displaying an article list.

```
<txp:if_article_list>  
<p>  
<txp:sitename />  
</p>  
<txp:else />  
<p>  
</img>  
</p>  
</txp:if_article_list>
```

[Return to tag index](#)

# Txp:if article section

## Classification

---

The `if_article_section` tag is a [Conditional Tag](#), and therefore a [Container Tag](#), as all conditional tags are container tags. This tag checks the section name associated with a particular article, if the section name matches it will execute the contained statements. Should be used in an Article form.

## Syntax

---

The `if_article_section` tag has the following syntactic structure...

```
<txp:if_article_section> ...your content here...<txp:else />...default option...</txp:if_article_section>
```

## Attributes

---

Tag will accept the following attributes (note: attributes are **case sensitive**):

`name="text"`

Section name.

## Examples

---

### Example 1: Display section name if article is assigned to section "Poetry"

```
<txp:if_article_section name="Poetry">
<p><txp:section /></p>
</txp:if_article_section>
```

### Example 2: Display welcome text if section match, show default link otherwise

```
<txp:if_article_section name="Poetry">
<p>Fun With Poetry</p>
<txp:else />
<p><a href://" .index.php">Home</a></p>
</txp:if_article_section>
```

[Return to tag index](#)

# Txp:if category

## Classification

---

The `if_category` tag is a [Conditional Tag](#), and therefore a [Container Tag](#), as all conditional tags are container tags. Textpattern will replace this tag with the result of the **contained statement** when the name attribute matches a category search value / or (?c=**Category**, an

article list by category).

Should be used in a Page or Column (for an Article Form, use [Txp:if\\_article\\_category](#)).

## Syntax

---

The `if_category` tag has the following syntactic structure...

```
<txp:if_category> ...your content here...<txp:else />...default option...
</txp:if_category>
```

## Attributes

---

Tag will accept the following attributes (note: attributes are **case sensitive**). List multiple category names with comma separators (v4.0.2):

```
name="Category(,another category,...)"
    Category Name(s)
```

## Examples

---

### Example 1: Display category name if the article list is of category(Cat1) "Prose"

```
<txp:if_category name="Prose">
<p><txp:category1 /></p>
</txp:if_category>
```

### Example 2: Display a category list, and an article list of the category being displayed

Given the defined article categories: Prose, Poetry, and Opinions.

```
<txp:category_list label="Category Navigation" wraptag="p" class="list" />
<txp:if_category name="Prose">
<txp:recent_articles label="Prose" limit="25" break="br" wraptag="p"
class="list" category="Prose" />
</txp:if_category>
<txp:if_category name="Poetry">
<txp:recent_articles label="Poetry" limit="25" break="br" wraptag="p"
class="list" category="Poetry" />
</txp:if_category>
<txp:if_category name="Opinions">
<txp:recent_articles label="Opinions" limit="25" break="br" wraptag="p"
class="list" category="Opinions" />
</txp:if_category>
```

Styles could go this way:

```
p.list
{
font-family: Verdana, "Lucida Grande", Tahoma, Helvetica;
font-size: 11px;
color:#333;
```

```
margin-left: 10px;
border-left: 3px solid #ccc;
}
p.list a
{
color:#333;
margin-left:15px;
}
p.list a:hover
}
border-bottom: 1px dashed #333;
}
```

**Example 1: Display category name if the article list is of category(Cat1) "Prose", otherwise display the site name as a default**

```
<txp:if_category name="Prose">
<p><txp:category1 /></p>
<txp:else />
<h3><txp:sitename /></h3>
</txp:if_category>
```

[Return to tag index](#)

# Txp:if comments

## Classification

---

The `if_comments` tag is a [Conditional Tag](#), and therefore a [Container Tag](#), as all conditional tags are container tags. This tag checks to see if there are one or more comments associated with a particular article. Should be used in an article form.

## Syntax

---

The `if_comments` tag has the following syntactic structure...

```
<txp:if_comments> ...your content here...<txp:else />...default option...
</txp:if_comments>
```

## Attributes

---

This tag has no attributes.

## Examples

---

**Example 1: Display the number of comments associated with a particular article, otherwise display default text.**

```
<txp:if_comments>
<p><txp:comments_count /> Comments</p>
<txp:else />
<p>No Current Comments</p>
</txp:if_comments>
```

**Example 2: Display the number of comments associated with a particular article with no default option.**

```
<txp:if_comments>
<p><txp:comments_count /> Comments</p>
</txp:if_comments>
```

[Return to tag index](#)

# Txp:if comments allowed

## Classification

---

The `if_comments_allowed` tag is a [Conditional Tag](#), and therefore a [Container Tag](#), as all conditional tags are container tags. The tag will execute the contained statement if comments are allowed for a given article.

## Syntax

---

The `if_comments_allowed` tag has the following syntactic structure...

```
<txp:if_comments_allowed> ...Content... </txp:if_comments_allowed>
```

## Attributes

---

Tag will accept the following attributes (note: attributes are **case sensitive**):

```
id="The id assigned to the article your looking for comments status for"
    Article ID
```

## Notes on Context

This tag can be used in pages/columns, or in comment forms; though it will allow you to use an `id` attribute in a comment form, the default behavior (no attribute) will ensure consistency in comment/article matching when viewing an individual article. When used in a page/column context it will test the article identified by the attribute for status and act, or not, according to that status. It will not pass the `id` attribute to the contained statement, such as `<txp:comments />` or `<txp:comments_form />`, they must be added as attributes to the contained tag.

## Examples

---

This tag is mainly used in combination with the `<txp:if_comments_disallowed>` ...  
`</txp:if_comments_disallowed>`.

### Example 1: Giving user's indication of Comments status.

Comments for articles can be turned off or on at the authors discretion for any article that is published; by using the following scheme in an article form, you can still have the on/off control over comments while still giving users indication of comment status.

```
<txp:if_comments_allowed>
  <txp:comments_form />
</txp:if_comments_allowed>
```

```
<txp:if_comments_disallowed>
  <p>Comments are turned off for this article.</p>
</txp:if_comments_disallowed>
```

### Example 2: Column list for displaying id numbers for comments for article id 3 if comments are currently allowed on article id 3.

Tags

```
<txp:if_comments_allowed id="3">
<txp:comments id="3" form="lineitem" break="li" wraptag="ul"
breakclass="special" />
<txp:else />
<p>Comments closed</p>
</txp:if_comments_allowed>
```

Form (lineitem) Type(comment)

```
<txp:comment_id />
```

Styles could go this way

special

```
{
  display:list-item;
  list-style-type:none;
}
```

note: `<txp:comment_permalink>` will only link content correctly with an individual article displayed.

[Return to tag index](#)

# Txp:if comments disallowed

## Classification

---

The `if_comments_disallowed` tag is a [Conditional Tag](#), and therefore a [Container Tag](#), as all conditional tags are container tags. The tag will execute the contained statement if comments are allowed for a given article.

## Syntax

---

The `if_comments_allowed` tag has the following syntactic structure...

```
<txp:if_comments_disallowed> ...Content... </txp:if_comments_disallowed>
```

## Attributes

---

Tag will accept the following attributes (note: attributes are **case sensitive**):

```
id="The id assigned to the article your looking for comments status for"  
    Article ID
```

## Notes on Context

This tag can be used in pages/columns or in comment forms; though it will allow you to use an id attribute in a comment form, the default behavior (no attribute) will ensure consistency in comment/article matching when viewing an individual article. When used in a page/column context it will test the article identified by the attribute for status and act, or not, according to that status. It will not pass the id attribute to the contained statement, such as [<txp:comments />](#) or [<txp:comments\\_form />](#), they must be added as attributes to the contained tag.

## Examples

---

This tag is mainly used in combination with the `<txp:if_comments_allowed> ... </txp:if_comments_allowed>`.

### Example 1: Giving user's indication of Comments status.

Comments for articles can be turned off or on at the authors discretion for any article that is published; by using the following scheme in an article form, you can still have the on/off control over comments while still giving users indication of comment status.

```
<txp:if_comments_allowed>  
    <txp:comments_form />  
</txp:if_comments_allowed>
```

```
<txp:if_comments_disallowed>  
    <p>Comments are turned off for this article.</p>  
</txp:if_comments_disallowed>
```

**Example 2: Column list for displaying links to comments for article id 3, using comment id as text, if comments are currently not allowed on article id 3.**

Tags

```
<txp:if_comments_disallowed id="3">
<txp:comments id="3" form="lineitem" break="li" wraptag="ul"
breakclass="special" />
</txp:if_comments_disallowed>
```

Form (lineitem) Type(comment)

```
<small><txp:comment_permalink><txp:comment_id /></txp:comment_permalink></small>
```

Styles could go this way

```
.special
{
  display:list-item;
  list-style-type:none;
}
```

[Return to tag index](#)

# Txp:if custom field

## Classification

---

The `if_custom_field` tag is a [Conditional Tag](#), and therefore a [Container Tag](#), as all conditional tags are container tags. The function of the tag is to check to see if there are one or more custom fields associated with a particular article, and if so, to display those custom fields. (The contents of a custom field can be displayed with the `<txp:custom_field />` tag.)

## Syntax

---

The `if_custom_field` tag has the following syntactic structure...

```
<txp:if_custom_field> ...your content here...<txp:else />...default option...
</txp:if_custom_field>
```

## Attributes

---

Tag will accept the following attributes (note: attributes are **case sensitive**):

- **name**="add name here" (allows you to add the custom name to your field)
- **val**="add content here" (allows you to add the content you want revealed in the field)

## Examples

---

### Example 1: Display contents of custom fields

A basic example of how to display the contents of one or more custom fields is over at the [explanation page of the <txp:custom\\_field /> tag](#)

### Example 2: Display contents of custom fields and create behavior if field has no content

The purpose of the conditional tag for custom fields is to check if a custom field has any content and display it depending on the result of that check. A more elaborate function is to check if a custom field has a specific content but we'll look at that in example 3. Let's first look at a basic example for the use of a conditional with custom fields.

Say, you are publishing book reviews on your site and you use custom fields to enter the author, title, publisher and year of publication ([see example](#)).

Some of the books have a subtitle, others don't. You can take care of that with `if_custom_field`. You set up the conditional to check if the custom field you named "subtitle" holds any content and if it does it will be displayed. If it's empty, the field won't turn up on the page. Here's the relevant part of the code:

```
<txp:if_custom_field name="subtitle">
  <txp:custom_field name="subtitle" />
</txp:if_custom_field>
```

If the custom field is empty, nothing will be displayed.

The whole set of custom fields could look like this:

```
<p><txp:custom_field name="author" />: <txp:custom_field name="title" /> <br />
  <txp:if_custom_field name="subtitle">
    <txp:custom_field name="subtitle" />
  </txp:if_custom_field> <br />
published by <txp:custom_field name="publisher" /> in <txp:custom_field
name="year" />.</p>
```

For a **book that has a subtitle**, this will put out the following:

```
<p>Stephen Covey: The Seven Habits of Highly Effective People<br />
Powerful Lessons in Personal Change <br />
published by Simon & Schuster in 2002.</p>
```

For a **book without a subtitle**, this will be shown:

```
<p>J.R.R. Tolkien: The Lord of the Rings <br />
published by HarperCollins in 2004.</p>
```

### Example 3: Display content if a custom field has a certain value

A completely different use might be a mood indicator. Say, you defined a custom field "mood" in which you enter a word to indicate your mood while writing an article. You enter either "happy" or "sad".

With the articles, you want to display a smiley for each of the moods you enter. The code in your article form could look like this:

```
<txp:if_custom_field name="mood" val="happy">
```

```

</txp:if_custom_field>
```

```
<txp:if_custom_field name="mood" val="sad">
  
</txp:if_custom_field>
```

### Example 4: Display content if a custom field contains data or display something else if it doesn't

Say, you publish music reviews and you've set up some custom fields for the band name, the album title and the band's website. But not all bands have a website and you want to display a standard message if a band hasn't got a website.

This could be the code:

```
...
<txp:if_custom_field name="website">
  <txp:custom_field name="website" />
<txp:else />
  <p>Unfortunately, this band hasn't got a website.</p>
</txp:if_custom_field>
...
```

[Return to tag index](#)

# Txp:if different

## Classification

---

The `if_different` tag is a [Conditional Tag](#), and therefore a [Container Tag](#), as all conditional tags are container tags. Textpattern will execute the contained statement when the value of the contained statement differs from the preceding value for the contained statement. Can be used in article, link, comment, and file forms

## Syntax

---

The `if_different` tag has the following syntactic structure...

```
<txp:if_different> ...your content here...<txp:else />...default option...
</txp:if_different>
```

## Attributes

---

This tag has no attributes.

## Examples

---

### Example 1: Display posting time per article once per day.

#### [Weblog](#)

```
<txp:if_different>
<!-- heading - only once per day -->
<h3><txp:posted format="%d %B %Y" /></h3>
</txp:if_different>
```

[Return to tag index](#)

# Txp:if excerpt

## Classification

---

The `if_excerpt` tag is a [Conditional Tag](#), and therefore a [Container Tag](#), as all conditional tags are container tags. The tag will execute the contained statement if an excerpt is associated with the article being displayed.

## Syntax

---

The `if_excerpt` tag has the following syntactic structure...

```
<txp:if_excerpt> ...your content here...<txp:else />...default option...
</txp:if_excerpt>
```

## Attributes

---

This tag has no attributes.

## Examples

---

### Example 1: Display the excerpt text, or a default link, as part of an article form using conditional tags

```
<txp:if_excerpt>
<txp:excerpt />
<txp:else />
Section <a href="/subdirname/index.php?s=<txp:section />"><txp:section /></a>
</txp:if_excerpt>
```

This example will display the section name linked as **/"subdirectory if any"/index.php?s="section name"** to provide a link to return the articles from the same section as the article being displayed.

Related info: [<txp:excerpt />](#)

[Return to tag index](#)

# Txp:if first article

## Classification

---

The `if_first_article` tag is a [Conditional Tag](#), and therefore a [Container Tag](#), as all conditional tags are container tags. Textpattern will execute the contained statement from within an article [form](#) when the displayed article is the first in the currently displayed list. It will display in both single article and article list modes. Should be used in an article [form](#)

## Syntax

---

The `if_first_article` tag has the following syntactic structure...

```
<txp:if_first_article> ...your content here...<txp:else />...default option...
</txp:if_first_article>
```

## Attributes

---

This tag has no attributes.

## Examples

---

**Example 1: Add a linked section by title to the header of the first article in an article list.**

```
<h3><txp:permlink><txp:title /></txp:permlink> · <txp:posted /> by <txp:author
/>
<txp:if_first_article>
· Section: <txp:section linked="1" title="1" />
</txp:if_first_article></h3>
<txp:body />
<txp:comments_invite wraptag="p" />
```

[Return to tag index](#)

# Txp:if individual article

## Classification

---

The `if_individual_article` tag is a [Conditional Tag](#), and therefore a [Container Tag](#), as all conditional tags are container tags. The tag will execute the contained statement if an individual article is being displayed.

## Syntax

---

The `if_individual_article` tag has the following syntactic structure...

```
<txp:if_individual_article> ...your content here...<txp:else />...default
option... </txp:if_individual_article>
```

## Attributes

---

This tag has no attributes.

## Examples

---

**Example 1: Excerpt from RC5 default page template, selects next-prev or older-newer navigation method.**

```
<txp:article />
<txp:if_individual_article>
<p>
<txp:link_to_prev><txp:prev_title /></txp:link_to_prev>
<txp:link_to_next><txp:next_title /></txp:link_to_next>
</p>
</txp:if_individual_article>
<txp:if_article_list>
<p>
<txp:older>Previous</txp:older>
<txp:newer>Next</txp:newer>
</p>
</txp:if_article_list>
```

**Example 2: Use [<txp:else />](#) to display the site's name when showing a single article, and a logo when not displaying a single article.**

```
<txp:if_individual_article>
<p>
<txp:sitename />
</p>
<txp:else />
<p>
</img>
</p>
</txp:if_individual_article>
```

[Return to tag index](#)

# Txp:if last article

## Classification

---

The `if_last_article` tag is a [Conditional Tag](#), and therefore a [Container Tag](#), as all conditional tags are container tags. Textpattern will execute the contained statement from

within an article [form](#) when the displayed article is the last in the currently displayed list. It will display in both single article and article list modes. Should be used in an article [form](#)

## Syntax

---

The `if_last_article` tag has the following syntactic structure...

```
<txp:if_last_article> ...your content here...<txp:else />...default option...
</txp:if_last_article>
```

## Attributes

---

This tag has no attributes.

## Examples

---

### Example 1: Add an image after the last article in an article list.

```
<h3><txp:permalink><txp:title /></txp:permalink> · <txp:posted /> by <txp:author
/></h3>
<txp:body />
<txp:comments_invite wraptag="p" />
<txp:if_last_article>
</img>
</txp:if_last_article>
```

[Return to tag index](#)

# Txp:if search

## Classification

---

The `if_search` tag is a [Conditional Tag](#), and thus a [Container Tag](#). The tag will execute the contained statement if the called page is the result of a search.

## Syntax

---

The `if_search` tag has the following syntactic structure...

```
<txp:if_search> ...your content here...<txp:else />...default option...
</txp:if_search>
```

## Attributes

---

This tag has no attributes.

### Example 1: Select a stylesheet named "search" when search results are being displayed, and a stylesheet determined by the active section for normal page display

```
<txp:if_search>
```

```
<link rel="stylesheet" href="<txp:css n="search" />" type="text/css" />
<txp:else />
<link rel="stylesheet" href="<txp:css />" type="text/css" />
</txp:if_search>
```

[Return to tag index](#)

## Txp:if section

### Classification

---

The `if_section` tag is a [Conditional Tag](#), and thus a [Container Tag](#). The tag will execute the contained statement if the called page is part of the section specified with the `name` attribute.

### Syntax

---

The `if_section` tag has the following syntactic structure...

```
<txp:if_section> ...your content here...<txp:else />...default option...
</txp:if_section>
```

### Attributes

---

Tag will accept the following attributes (note: attributes are **case sensitive**). List multiple section names with comma separators (v4.0.2):

```
name="section name(,section name 2,...)"
```

Section Name(s)

Note: to select the default section [=home page], use **name=""**. To select the default section and other sections, use **name=",otherSection1(,otherSection2,...)"**. Note the initial comma!

### Example 1: Conditionally display text for a section

```
<txp:if_section name="about">
<p>danger, ego pages ahead!</p>
<txp:else />
<p>nothing. just nothing. any ideas? anybody?</p>
</txp:if_section>
```

[Return to tag index](#)

# Txp:image /

## Classification

---

The `image` tag is a [Single Tag](#) Textpattern will replace this tag with the `IMG SRC` html tag matching the numeric "ID" assigned by Textpattern when the image was uploaded via the TXP "images" screen. Its context is article, page or column.

## Syntax

---

The `image` tag has the following syntactic structure...

```
<txp:image />
```

## Attributes

---

Tag will accept the following attributes (note: attributes are **case sensitive**):

`id="The id assigned at upload"`

Can be found on the Images tab.

`name="image name"`

Image Name as shown on the Image tab.

`style="style rule"`

CSS style rule.

`align="attribute text"`

IMG align attribute.

`class="class name here"`

CSS class attribute).

## Examples

---

### Example 1: Display image id# 23

```
<txp:image id="23" />
```

### Example 2: Display a styled image selection

```
<txp:image id="3" class="content" />
```

Styles could go this way

```
img.content
{
  background: #fff;
  border: 1px solid #ccc;
  display: block;
  margin: -5px 5px 5px -5px;
  padding: 4px;
```

```
position: relative;
}
```

[Return to tag index](#)

# Txp:image display /

## Classification

---

The `image_display` tag is a [Single Tag](#) that is intended to be used in tandem with `image_index`

## Syntax

---

The `image_display` tag has the following syntactic structure...

```
<txp:image_index /><txp:image_display />
```

## Use

---

The `image_index` tag returns the html `<IMG>` links to all thumbnails associated with the image category called, the assumption is that there will be a parent image available. These thumbnail images will be hyperlink with the textpattern URL to the parent image. The `image_display` tag will then display the parent image of the thumbnail when the thumbnail is clicked.

## Example 1: Create a category list of image categories with `category_list`

---

```
<txp:category_list type="image" limit="10" break="li" wraptag="ol" />
```

[Return to tag index](#)

# Txp:image index /

## Classification

---

The `image_index` tag is a [Single Tag](#) that is intended to be used in tandem with `image_display`

## Syntax

---

The `image_index` tag has the following syntactic structure...

```
<txp:image_index /><txp:image_display />
```

## Use

---

The `image_index` tag returns the html `<IMG>` links to all thumbnails associated with the image category called, the assumption is that there will be a parent image available. These thumbnail

images will be hyperlink with the textpattern URL to the parent image. The `image_display` tag will display the parent image of the thumbnail when the thumbnail is clicked.

## Attributes

---

Tag will accept the following attributes (note: attributes are **case sensitive**):

`label="label text"`

Label for the top of the list. Default is empty.

`labeltag="labeltag text"`

Independent wraptag for label for the top of the list. Default is empty.

`parent=""`

in development.

`break="tag text"`

HTML tag to be used for line breaks, without brackets. Default is empty.

`wraptag="wraptag text"`

HTML tag to be used as the wraptag, without brackets. Default is empty.

`c=""`

`doSlash($c)` // Keep the option to override categories due to backward compatibility. :

`type="category type"`

Present choices are Article, Image, Link, and File.

`class="class name"`

CSS class attribute, default is `image_index`.

## Example 1: Create a category list of image categories with `category_list`

---

```
<txp:category_list type="image" limit="10" break="li" wraptag="ol" />
```

[Return to tag index](#)

# Txp:keywords /

## Classification

---

The `keywords` tag is a [Single\\_Tag](#) Textpattern will replace this tag with the keywords associated with the article being displayed. Could be used in an article [form](#).

## Syntax

---

The `keywords` tag has the following syntactic structure...

```
<txp:keywords />
```

## Attributes

---

This tag takes no attributes.

## Examples

---

**Example 1: Display the the keywords associated with an article when included in an article form**

```
<h3><txp:title /> <span class="permalink"><txp:permalink>::</txp:permalink></span>
<span class="date"><txp:posted /></span></h3>
<txp:body /><hr><txp:keywords />
```

[Return to tag index](#)

# Txp:lang /

## Classification

---

The lang tag is a [Single\\_Tag](#). Textpattern will replace this tag with the 2-letter code of the language which is set as the site's language preference on the [Preferences Subtab](#), according to [\[1766\]](#).

## Syntax

---

The lang tag has the following syntactic structure...

```
<txp:lang />
```

## Attributes

---

This tag has no attributes.

## Examples

---

**Example 1: Define a document's language inside the <head> tag**

```
<head>
<html lang=<txp:lang />>
<!-- other markup -->
</head>
```

[Return to tag index](#)

# Txp:link /

## Classification

---

The `link` tag is a [Single Tag](#) which is used to return an XHTML hyperlink, as defined under the "links" tab, using the **Title** field as the links text. Its context is a links [form](#).

## Syntax

---

The `link` tag has the following syntactic structure...

```
<txp:link />
```

## Attributes

---

This tag has no attributes.

## Examples

---

### Example 1: Display a link and its description field contents

```
<p><txp:link /><br /><txp:link_description /></p>
```

Used in "link" forms with [<txp:linklist />](#)

[Return to tag index](#)

# Txp:link description /

## Classification

---

The `link_description` tag is a [Single Tag](#) which is used to return the text from the **Description** field as defined under the "links" tab. Its context is a links [form](#).

## Syntax

---

The `link_description` tag has the following syntactic structure...

```
<txp:link_description />
```

## Attributes

---

This tag has no attributes.

## Examples

---

### Example 1: Display a link and its description field contents

```
<p><txp:link /><br /><txp:link_description /></p>
```

Used in "link" forms with [<txp:linklist />](#)

[Return to tag index](#)

# Txp:link feed link /

## Classification

---

The `link_feed_link` tag is a [Single Tag](#) Textpattern will replace this tag with an XHTML link to the sites links RSS feed. Use in Page or Column.

## Syntax

---

The `link_feed_link` tag has the following syntactic structure...

```
<txp:link_feed_link />
```

## Attributes

---

Tag will accept the following attributes (note: attributes are **case sensitive**):

`label="label text"`

Label for feed link.

`break="tag text"`

HTML line breaks.

`wraptag="wraptag text"`

Character to be used as the wraptag without brackets.

`category="category name"`

Selected from available link categories.

`title="title text"`

XML feed title.

`flavor="rss or atom"`

## Examples

---

### Example 1: Display a feed link called "Commerce Links" with a flavor attribute set to "atom".

```
<txp:link_feed_link label="Commerce Links" flavor="atom" />
```

[Return to tag index](#)

# Txp:link to home

## Classification

---

The `link_to_home` tag is a [Container Tag](#), . The `link_to_home` tag returns a link to the site's home page.

## Syntax

---

The `link_to_home` tag has the following syntactic structure...

```
<txp:link_to_home> ...Text or Tag... </txp:link_to_home>
```

## Attributes

---

Tag will accept the following attributes (note: attributes are **case sensitive**):

`class="class name"`

CSS class attribute, default is `link_to_home`.

## Examples

---

**Example 1: Home page link with Site's name .**

```
<txp:link_to_home><txp:sitename /></txp:link_to_home>
```

[Return to tag index](#)

# Txp:link to next

## Classification

---

The `link_to_next` tag is a [Container Tag](#), .

Textpattern will replace this tag with the permanent URL of the next article by posting date. The container tags wrap the text or tag assigned to the link. Use in Page or Column.

## Syntax

---

The `link_to_next` tag has the following syntactic structure...

```
<txp:link_to_next> ...Text or Tag... </txp:link_to_next>
```

## Attributes

---

Tag will accept the following attributes (note: attributes are **case sensitive**):

`showalways="1"`

Show wrapped value even when no next article exists.

## Examples

---

**Example 1: Display a link to the next article using the article title.**

```
<txp:link_to_next><txp:next_title /></txp:link_to_next>
```

**Example 2: Use "showalways" to enable link\_to\_next to display "Next" in plain text even when no next article exists.**

```
<txp:link_to_next showalways="1">Next</txp:link_to_next>
```

note: showalways will enable link\_to\_next to show what is wrapped in the tags;

<[txp:next\\_title /](#)> returns nothing if there is no next title, so nothing is displayed. Use text or the returned value that you need displayed.

[Return to tag index](#)

# Txp:link to prev

## Classification

---

The link\_to\_prev tag is a [Container Tag](#), .

Textpattern will replace this tag with the permanent URL of the previous article by posting date. The container tags wrap the text or tag assigned to the link. Use in Page or Column.

## Syntax

---

The link\_to\_prev tag has the following syntactic structure...

```
<txp:link_to_prev> ...Text or Tag... </txp:link_to_prev>
```

## Attributes

---

Tag will accept the following attributes (note: attributes are **case sensitive**):

showalways="1"

Show wrapped value even when no next article exists.

## Examples

---

**Example 1: Display a link to the previous article using the article title.**

```
<txp:link_to_prev><txp:prev_title /></txp:link_to_prev>
```

**Example 2: Display a link to the previous article using "Previous" as text, show unlinked "Previous" as text if nothing previous exists.**

```
<txp:link_to_prev showalways="1">Previous</txp:link_to_prev>
```

note: showalways will enable link\_to\_prev to show what is wrapped in the tags;

<[txp:prev\\_title /](#)> returns nothing if there is no previous title, so nothing is displayed. Use text or the returned value that you need displayed.

[Return to tag index](#)

# Txp:linkdesctitle /

## Classification

---

The `linkdesctitle` tag is a [Single Tag](#) which is used to return an XHTML hyperlink, as defined under the "links" tab, using the **Title** field contents as the links text. The **Description** field contents will be displayed as a title attribute. Its context is a links [form](#).

## Syntax

---

The `linkdesctitle` tag has the following syntactic structure...

```
<txp:linkdesctitle />
```

## Attributes

---

This tag has no attributes.

## Examples

---

### Example 1: Display a link and its Title field contents

```
<p><txp:linkdesctitle /></p>
```

Link, title=Description

Used in "link" forms with [<txp:linklist />](#)

[Return to tag index](#)

# Txp:linklist /

## Classification

---

The `linklist` tag is a [Single Tag](#) which is used to produce a list of links from a predefined list. Its context is page or column. Links can be input and assigned to categories under the [links tab](#).

## Syntax

---

The `linklist` tag has the following syntactic structure...

```
<txp:linklist />
```

## Attributes

---

Tag will accept the following attributes (note: attributes are **case sensitive**):

`form="form name"`

Selected from available link forms. Default is `plainlinks`.

`label="label text"`

Label for the top of the list. Default is empty.

`labeltag="labeltag text"`

Independent `wraptag` for label for the top of the list. Default is empty.

`limit="integer"`

Maximum number of listed links. Default is 10.

`break="tag text"`

HTML tag to be used for line breaks, without brackets. Default is empty.

`wraptag="wraptag text"`

HTML tag to be used as the `wraptag`, without brackets. Default is empty.

`category="category name"`

Selected from available link categories.

`sort="sort option"`

(see below for details)

`class="class name"`

CSS class attribute, default is `linklist`.

### Attribute Semantics

- **label:** This string will be prepended to the link list. When using semantic markup ("`wraptag`" being either `ol` or `ul`), `label` will be the **first** list item.
- **break:** This string will be used to separate list items. Suggested values include `br` and `hr` for presentational markup, or `li` if you preferred semantic markup. Textpattern cares for the correct nesting of tags in either case. NB: Choosing `li` as the break string requires `wraptag` to be either `ul` or `ol` and vice versa.
- **wraptag:** This string will be used to enclose the whole list. Suggested values include `p` for presentational markup, `ol` or `ul` for semantic markup. NB: Appending additional HTML attributes like `class`, `style` or `start` is not supported.
- **sort:** This attribute's value will be used as a SQL `order by` clause and can thus contain every table row of the `txp_link` database table followed by SQL sort order indications `asc` and `desc` respectively. Valid values include `url`, `linkname desc`, `description asc`, `date asc`, `rand()` and others.

### Examples

---

#### Example 1: Display a list of the 10 links at the top of the sort order from the General category

```
<txp:linklist form="Links" category="General" limit="10" sort="linksort"
wraptag="p" />
```

### Example 2: Display an ordered list of links selected by category with conditional tags

This example uses the displayed page's category as the criterion for choosing the linklist's category.

```
<txp:if_category name="100">
<txp:linklist label="First Floor" category="First" wraptag="ol" break="li" />
</txp:if_category>
<txp:if_category name="200">
<txp:linklist label="Second Floor" category="Second" wraptag="ol" break="li" />
</txp:if_category>
```

### Example3: Links Form (default "Links")

```
<txp:link />
.. <txp:link_description />
.. <txp:linkdesctitle /> ..
```

The form is repeated for each link provided by `<txp:linklist />`

[Return to tag index](#)

# Txp:meta keywords /

## Classification

---

The `meta_keywords` tag is a [Single Tag](#) Textpattern will replace this tag with an HTML meta tag as follows:

```
<meta name="keywords" content="keywords as set in your article's keywords input area" />
```

Used in the head of an individual article page template.

## Syntax

---

The `meta_keywords` tag has the following syntactic structure...

```
<txp:meta_keywords />
```

## Attributes

---

This tag takes no attributes.

## Examples

---

**Example 1: Create the meta tag as returned when using "sauce,caramel,sugar" as keywords when saving your article.**

**Tag:** In the head of an individual article page.

```
<txp:meta_keywords />
```

Output:

```
<meta name="keywords" content="sauce,caramel,sugar" />
```

Note: Tag returns nothing if no keywords are set for an article.

[Return to tag index](#)

## Txp:meta author /

### Classification

---

The `meta_author` tag is a [Single Tag](#) Textpattern will replace this tag with an HTML meta tag as follows:

```
<meta name="author" content="Article authors name" />
```

Used in the head of an individual article page template.

### Syntax

---

The `meta_author` tag has the following syntactic structure...

```
<txp:meta_author />
```

### Attributes

---

This tag takes no attributes.

### Examples

---

**Example 1: Create the meta tag showing "sysop" as the posting author.**

**Tag:** In the head of an individual article page.

```
<txp:meta_author />
```

Output:

```
<meta name="author" content="sysop" />
```

[Return to tag index](#)

## Txp:newer

### Classification

---

The `newer` tag is a [Container Tag](#), . Should be used in a page after an article tag.

### Behavior

Textpattern will replace this tag with an XHTML link to the next list of articles in the sort order. The container tags wrap the text or tag assigned to the link. An article list consists of the assigned number of articles set by the article tag. If there are no articles available having **Newer** status (articles ranked higher, or newer, in the present sort criteria than the present

top of page article) `<txp:newer>` will not display. It is normally seen used in tandem with `<txp:older>`

Given a `<txp:article limit ="5" />` tag on the page in question. `<txp:newer>` will page up 5 articles at a time from the oldest post forward in time to the most recently posted article.

### Syntax

---

The `newer` tag has the following syntactic structure...

```
<txp:newer> ...tag or text... </txp:newer>
```

### Attributes

---

This tag has no attributes.

### Examples

---

**Example 1: Display pagination link with the text "Newer".**

```
<txp:newer>Newer</txp:newer>
```

[Return to tag index](#)

# Txp:next title /

### Classification

---

The `next_title` tag is a [Single Tag](#). Textpattern will replace this tag with the name (text) of the next article in the sort order. The container tag `<txp:link_to_next>` wraps the text or tag and assigns the link. Should be used in Page or Column.

### Syntax

---

The `next_title` tag has the following syntactic structure...

```
<txp:next_title />
```

### Attributes

---

This tag takes no attributes.

### Examples

---

**Example 1: Display a link to the next article when displaying individual articles."**

```
<txp:link_to_next><txp:next_title /></txp:link_to_next>
```

[Return to tag index](#)

# Txp:older

## Classification

---

The `older` tag is a [Container Tag](#), . Should be used in a page after an article tag.

## Behavior

Textpattern will replace this tag with an XHTML link to the next list of articles in the sort order. The container tags wrap the text or tag assigned to the link. An article list consists of the assigned number of articles set by the article tag. If there are no articles available having **Older** status (articles ranked lower, or later, in the present sort criteria than the present bottom of page article) `<txp:older>` will not display. It is normally seen used in tandem with [<txp:newer>](#)

Given a `<txp:article limit ="5" />` tag on the page in question. `<txp:older>` will page down 5 articles at a time from the most recent post back in time to the oldest.

## Syntax

---

The `older` tag has the following syntactic structure...

```
<txp:older> ...tag or text... </txp:older>
```

## Attributes

---

This tag has no attributes.

## Examples

---

**Example 1: Display pagination link with the text "Older".**

```
<txp:older>Older</txp:older>
```

[Return to tag index](#)

# Txp:output form /

## Classification

---

The `output_form` tag is a [Single Tag](#). Textpattern will replace this tag with the content resulting from the form called by the tag.

## Syntax

---

The `output_form` tag has the following syntactic structure...

```
<txp:output_form />
```

## Attributes

---

Tag will accept the following attributes (note: attributes are **case-sensitive**):

`form="form name"`

Use specified form. Default is unset, nothing is output. See [Form](#).

## Examples

---

### Example 1: Display static text at the head of a column

```
<txp:output_form form="headtext" />
```

[Return to tag index](#)

# Txp:page title /

## Classification

---

The `page_title` tag is a [Single Tag](#).

### Behavior

Textpattern will replace this tag with text depending on what context it is being used in. Results appear as follows:

#### Article List:

Your Site Name

#### Articles by Category:

Your Site Name: Category

#### Search Results page:

Your Site Name: "Search results": Search term

#### Single Article page:

Your Site Name: Article name

#### Comments display:

"Comments on": Article name

## Syntax

---

The `page_title` tag has the following syntactic structure...

```
<txp:page_title />
```

## Attributes

---

Tag will accept the following attributes (note: attributes are **case sensitive**):

`separator="separator characters"`

Default is " : "

## Examples

---

### Example 1: Show page titles with right pointing angle quotes (») as a separator

```
<txp:page_title separator=" &raquo; " />
```

[Return to tag index](#)

# Txp:password protect /

## Classification

---

The `password_protect` tag is a [Single Tag](#). When Textpattern encounters the password protect tag it causes the user to be prompted for username and password, if these match the attributes set in the tag, the user is allowed access to the site. The `<txp:password_protect login="user" pass="password" />` can go anywhere, from page template, to article and forms.

## Syntax

---

The `password_protect` tag has the following syntactic structure...

```
<txp:password_protect login="user" pass="password" />
```

## Attributes

---

Tag will accept the following attributes (note: attributes are **case sensitive**):

```
login="Username Text"
```

```
pass="Password Text"
```

## Examples

---

### Example 1: Cause Textpattern to prompt the user for a username (theuser) and password (thepassword)

```
<txp:password_protect login="theuser" pass="thepassword" />
```

Note: It is not adequate to protect a single section. This is not due to the tag itself, but rather because of how Textpattern handles urls. By changing the url an article can be rendered with a different section template, which would mean that the tag in the protected section would not be rendered and could not protect the article - only page requests that would be rendered in that section would be protected.

[Return to tag index](#)

# Txp:permlink

## Classification

---

The `permlink` can be used as a [Container Tag](#), or a [Single Tag](#), to return the permanent url of the article being displayed: if used as a container tag, the HTML required to output a hyperlink; if used as a single tag, only the url itself is returned. This tag is used in an article form.

## Syntax

---

The `permlink` tag has the following syntactic structure...

```
<txp:permlink>...your content here... </txp:permlink> or <txp:permlink />
```

## Attributes

---

This tag has no attributes.

## Container Example

---

### Code

```
<txp:permlink><txp:title /></txp:permlink>
```

### Outputs something like

```
<a href="http://yoursite/index.php?id=2">article title</a>
```

## Single Example

---

### Code

```
<txp:permlink />
```

### Outputs something like

```
http://yoursite/index.php?id=2
```

## Customizing Permanent Links

---

By default `permlink` returns only a very basic link, which doesn't allow for customizing the link title, or adding a CSS class, etc. Using the tag in its [Single Tag](#) capacity opens up a lot more possibilities.

For example, to have the permanent link have an HTML title attribute of the article's title, and also apply a class to it named "orange":

```
<a href="<txp:permlink />" title="<txp:title />" class="orange"><txp:title /></a>
```

[Return to tag index](#)

# Txp:php

## Classification

---

The `php` tag is a [Container Tag](#), . The `php` tag is Textpatterns variation of `<?php "content" ?>`.

This allows flexible administrative usage control, to control use of PHP in pages, article bodies, or both. The access settings are available under the preferences / advanced preferences tab.

## Syntax

---

The `php` tag has the following syntactic structure...

```
<txp:php> ...php code here (no <?php ?>)... </txp:php>
```

## Attributes

---

This tag has no attributes.

## Examples

---

### Example 1: Retrieve PHP server library information.

```
<txp:php>
    phpinfo();
</txp:php>
```

[Return to tag index](#)

# Txp:popup /

## Classification

---

The `popup` tag is a [Single Tag](#) Textpattern will replace this tag with a popup selector for browsing by section ( `s` ) or category ( `c` ). Should be used in a Page or Column.

## Syntax

---

The `popup` tag has the following syntactic structure...

```
<txp:popup />
```

## Attributes

---

Tag will accept the following attributes (note: attributes are **case sensitive**):

```
label="label text"
```

Label for the top of the list.

type="s" or "c"

Section or Category.

wraptag="wraptag text"

Character to be used as the wraptag without brackets.

## Examples

---

### Example 1: Display a category browse popup selector.

```
<txp:popup type="c" wraptag="p" />
```

### Example 2: Display a popup selector labeled "Browse".

```
<txp:popup label="Browse" type="c" wraptag="p" />
```

[Return to tag index](#)

# Txp:posted /

## Classification

---

The `posted` tag is a [Single Tag](#) which is used to return the publish date of the article being displayed. The format is determined by the settings specified in the Date Format, or Archive Date Format, fields on the "admin" - "preferences" screen. Its context is an article form.

## Syntax

---

The `posted` tag has the following syntactic structure...

```
<txp:posted />
```

## Attributes

---

Tag will accept the following attributes (note: attributes are **case-sensitive**):

```
format="desired format"
```

Override default date format, as set in preferences. Available values: any valid [strftime\(\)](#) string values, or `since`

## Examples

---

### Example 1: Display "since" format date setting

```
<p>Posted: <txp:posted format="since" /></p>
```

would result in

```
<p>Posted: 29 Days ago</p>
```

## Example 2: Display custom format date setting using strftime() format

```
<p>Posted: <txp:posted format="%b %d, %Y" /></p>
```

would result in:

```
<p>Posted: May 28, 2005</p>
```

[Return to tag index](#)

# Txp:prev title /

## Classification

---

The `prev_title` tag is a [Single Tag](#). Textpattern will replace this tag with the name (text) of the previous article in the sort order. The container tag `<txp:link_to_prev>` wraps the text or tag and assigns the link.

## Syntax

---

The `prev_title` tag has the following syntactic structure...

```
<txp:prev_title />
```

## Attributes

---

This tag takes no attributes.

## Examples

---

### Example 1: Display a link to the previous article when displaying individual articles."

```
<txp:link_to_prev><txp:prev_title /></txp:link_to_prev>
```

[Return to tag index](#)

# Txp:recent articles /

## Classification

---

The `recent_articles` tag is a [Single Tag](#) which is used to produce a list of permanent links to recent articles by title. Its context is page or column and it provides hyperlinked navigation by article list.

## Syntax

---

The `recent_articles` tag has the following syntactic structure...

```
<txp:recent_articles />
```

## Attributes

---

Tag will accept the following attributes (note: attributes are **case sensitive**):

`label="label text"`

Label for the top of the list. Default is empty.

`labeltag="labeltag text"`

Independent wraptag for label for the top of the list. Default is empty.

`limit="integer"`

Maximum number of listed articles. Default is 10.

`break="tag text"`

HTML tag to be used for line breaks, without brackets. Default is empty.

`wraptag="wraptag text"`

HTML tag to be used as the wraptag, without brackets. Default is empty.

`category="category name"`

Selected from available article categories.

`sort="sort option"`

(see below for details)

`sortdir="desc or asc"`

asc first first :: desc first last.

`class="class name"`

CSS class attribute, default is `recent_articles`.

## Attribute Semantics

- **label**: This string will be prepended to the link list. When using semantic markup ("`wraptag`" being either "`ol`" or "`ul`"), "`label`" will be the **first** list item.
- **break**: This string will be used to separate list items. Suggested values include "`br`" and "`hr`" for presentational markup, or "`li`" if you preferred semantic markup. Textpattern cares for the correct nesting of tags in either case. NB: Choosing `li` as the break string requires **wraptag** to be either `ul` or `ol` and vice versa.
- **wraptag**: This string will be used to enclose the whole list. Suggested values include "`p`" for presentational markup, "`ol`" or "`ul`" for semantic markup. NB: Appending additional HTML attributes like `class`, `style` or `start` is not supported.
- **sort**: This attribute's value will be used as a SQL `order by` clause and can thus contain every table row of the `txp_link` database table followed by SQL sort order indications `asc` and `desc` respectively. Valid values include "`url`", "`linkname desc`", "`description asc`", "`date asc`", "`rand()`" and others.

## Examples

---

### Example 1: Display a list of the 5 most recently posted articles

```
<txp:recent_articles label="Latest and Greatest" limit="5" />
```

### Example 2: Display a list of the 10 most recent articles meeting a complex criteria

```
<txp:recent_articles label="Latest" limit="10" break="br" wraptag="p"
category="Code" sortby="Section" sortdir="desc" />
```

### Example 3: Display a styled recent article list

```
<txp:recent_articles label="Recently" limit="10" break="li" wraptag="ul" />
```

Styles could go this way

```
recent_articles
{
  list-style-type:none;
}
```

[Return to tag index](#)

# Txp:recent comments /

## Classification

---

The `recent_comments` tag is a [Single Tag](#) Textpattern will replace this tag with a list of permanent links to recent comments. This list will be displayed with the format.

**Users Name(Article Name).**

Its context is page or column.

## Syntax

---

The `recent_comments` tag has the following syntactic structure...

```
<txp:recent_comments />
```

## Attributes

---

Tag will accept the following attributes (note: attributes are **case sensitive**):

`label="label text"`

Label for the top of the list. Default is empty.

`labeltag="labeltag text"`

Independent wraptag for label for the top of the list. Default is empty.

`limit="integer"`

Maximum number of listed comments. Default is 10.

`break="tag text"`

HTML tag to be used for line breaks, without brackets. Default is empty.

`wraptag="wraptag text"`

HTML tag to be used as the wraptag, without brackets. Default is empty.

`class="class name"`

CSS class attribute, default is `recent_comments`.

### Attribute Semantics

- **label:** This string will be prepended to the comments list. When using semantic markup ("`wraptag`" being either "`ol`" or "`ul`"), "`label`" will be the **first** list item.
- **break:** This string will be used to separate list items. Suggested values include "`br`" and "`hr`" for presentational markup, or "`li`" if you preferred semantic markup. Textpattern cares for the correct nesting of tags in either case. NB: Choosing `li` as the break string requires **wraptag** to be either `ul` or `ol` and vice versa.
- **wraptag:** This string will be used to enclose the whole list. Suggested values include "`p`" for presentational markup, "`ol`" or "`ul`" for semantic markup. NB: Appending additional HTML attributes like `class`, `style` or `start` is not supported.

### Examples

---

#### Example 1: Display a list of the 25 most recent comments with the label Recent Comments

```
<txp:recent_comments label="Recent Comments" limit="25" wraptag="p" break="br" />
```

#### Example 2: Display the default number of recent comments as an unordered list

```
<txp:recent_comments label="Recent Comments" wraptag="ul" break="li" />
```

Style for default class could go this way:

```
recent_comments
{
    list-style-type:none;
}
```

[Return to tag index](#)

## Txp:related articles /

### Classification

---

The `related_articles` tag is a [Single Tag](#) which is used to produce a list of permanent links to related articles by title.

Its context is page or column and it provides hyperlinked navigation by article list.

Related matches are selected as follows:

If a match to Cat1 or Cat2 of the individual article being displayed is found in either Cat1 or Cat2 of any other article in the database, it will cause that article to be listed as related.

If Cat1 of the individual article being displayed is left blank and Cat2 is not blank, then all other articles are selected as being related. If both Categories are left blank, then no articles are selected.

### Syntax

---

The `related_articles` tag has the following syntactic structure...

```
<txp:related_articles />
```

### Attributes

---

Tag will accept the following attributes (note: attributes are **case sensitive**):

`label="label text"`

Label for the top of the list. Default is empty.

`labeltag="labeltag text"`

Independent wraptag for label for the top of the list. Default is empty.

`limit="integer"`

Maximum number of listed articles. Default is 10.

`break="tag text"`

HTML tag to be used for line breaks, without brackets. Default is empty.

`wraptag="wraptag text"`

HTML tag to be used as the wraptag, without brackets. Default is empty.

`class="class name"`

CSS class attribute, default is `related_articles`.

### Examples

---

#### Example 1: Display a list of the 5 most recent related articles

```
<txp:related_articles label="Related" limit="5" />
```

#### Example 2: Display a styled related article list

```
<txp:related_articles label="Related" limit="10" break="li" wraptag="ul" />
```

Styles could go this way

```
related_articles
{
  list-style-type:none;
}
```

[Return to tag index](#)

# Txp:search input /

## Classification

---

The `search_input` tag is a [Single Tag](#). This tag will provide a text entry field for search parameters and an optional button to initiate the search.

## Syntax

---

The **search\_input** tag has the following syntactic structure...

```
<txp:search_input />
```

## Attributes

---

Tag will accept the following attributes (note: attributes are **case-sensitive**):

`section="section name"`

Restrict search results to articles from specified section. Default is unset, searches all sections.

`label="text label"`

Label for the top of the field. Default is `Search`. See [Attributes Cross Reference](#).

`button=button text"`

Creates and labels a button to initiate the search. Default is unset, no button is created.

`size="integer"`

Set the button size. Default is 15.

`wraptag="html tag"`

HTML tag to wrap the search field with. Default is `p`. See [Attributes Cross Reference](#).

`form="form name"`

Use specified form. Default is `search_input`. See [Form](#).

## Examples

---

### Example 1: Display a search input form

```
<txp:search_input label="Search" button="Search" size="15" wraptag="p" />
```

Textpattern, as of this writing, will use a user defined form named `search_results`, or an internally defined default form if no search result form is defined by this name by the user.

[Return to tag index](#)

# Txp:search result count /

## Classification

---

The `search_result_count` tag is a [Single Tag](#). This tag returns the number of article returned by the article tag. Use `<txp:if_search>` to screen for search results. Use in Page or Column(after the `<txp:article />` tag).

## Syntax

---

The `search_result_count` tag has the following syntactic structure...

```
<txp:search_result_count />
```

## Attributes

---

Tag will accept the following attributes (note: attributes are **case sensitive**):

```
text="alternate text"
```

Default is "articles found".

## Examples

---

**Example 1: Display the number of articles returned as follows: "5 articles found" when five articles are returned.**

Note: The `<txp:if_search>` conditional tag is required to recognize actual search results, without them the number of articles is returned by default.

```
<txp:search_result_count />
```

**Example 2: Display a search result count when a search is made as follows: "3 Hits" when three matches are returned.**

```
<txp:if_search>
<txp:search_result_count text="Hits" />
</txp:if_search />
```

**Example 3: Top display a search result count when a search is made as follows: "3 Hits" when three matches are returned.**

```
<txp:if_search>
<txp:article pgonly="y" category="Current" />
<txp:search_result_count text="Hits" />
<txp:article category="Current" />
</txp:if_search />
```

Note: The "pgonly" attribute sets the article tag to return pagination statistics without rendering the article list. Care must be taken to remain consistent with article tag attributes to keep statistics accurate.

[Return to tag index](#)

# Txp:search result date /

## Classification

---

The **search\_result\_date** tag is a [Single Tag](#). This tag will provide the article posted date as returned by the search function.

## Syntax

---

```
<txp:search_result_date />
```

## Attributes

---

This tag has no attributes.

### Example 1: Displays the posting date of an article in the search results form

```
<h3><txp:permlink><txp:title /></txp:permlink></h3>
<p><txp:search_result_date /><br/>
<small><txp:permlink><txp:permlink /></txp:permlink> .
<txp:posted /></small></p>
```

[Return to tag index](#)

# Txp:search result excerpt /

## Classification

---

The **search\_result\_excerpt** tag is a [Single Tag](#). The tag will show the occurrence of the search term with some surrounding context.

## Syntax

---

```
<txp:search_result_excerpt />
```

## Attributes

---

This tag has no attributes.

### Example 1: Display the search excerpt of an article in the search results form

```
<h3><txp:permlink><txp:title /></txp:permlink></h3>
<p><txp:search_result_excerpt /><br/>
<small><txp:permlink><txp:permlink /></txp:permlink> .
<txp:posted /></small></p>
```

[Return to tag index](#)

# Txp:search result title /

## Classification

---

The **search\_result\_title** tag is a [Single Tag](#). This tag will provide a hyperlinked title to an article as returned by the search function.

## Syntax

---

```
<txp:search_result_title />
```

## Attributes

---

This tag has no attributes.

### Example 1: Display a hyperlinked title to the article in the search results form

```
<h3><txp:search_result_title /> <txp:search_result_date /></h3>
```

[Return to tag index](#)

# Txp:search result url /

## Classification

---

The **search\_result\_url** tag is a [Single Tag](#). This tag will provide a hyperlinked url to an article as returned by the search function.

## Syntax

---

```
<txp:search_result_url />
```

## Attributes

---

This tag has no attributes.

### Example 1: Display a hyperlinked url to the article in the search results form

```
<h3><txp:permlink><txp:title /></txp:permlink></h3>
```

```
<p><txp:search_result_url /></p>
```

[Return to tag index](#)

# Txp:section /

## Classification

---

The `section` tag is a [Single Tag](#) Textpattern will replace this tag with the name of the current section when in page template, or the name of the article section when in article form, or the section as defined with the name attribute. Can be used in any context.

## Syntax

---

The `section` tag has the following syntactic structure...

```
<txp:section />
```

## Attributes

---

Tag will accept the following attributes (note: attributes are **case-sensitive**):

```
title="integer"
```

Display section name or title. Available values: 0 (section name) and 1 (section title). Default is 0.

```
link="integer"
```

Display as text or link. Available values: 0 (text) and 1 (link). Default is 0.

```
wraptag="wraptag text"
```

HTML tag to be used as the wraptag, without brackets. Default is empty.

```
name="name text"
```

Sets link to named section. Default is empty, which sets the link to current section.

## Examples

---

### Example 1: Displays the current section name

```
<txp:section />
```

### Example 2: Display hyperlinked section name when included in an article form

```
<txp:section link="1" />
```

### Example 3: Display hyperlinked section title when included in an article form.

```
<txp:section link="1" title="1" />
```

### Example 4: Display a section link, by title, to the archive section. (defined with the "name" attribute)

```
<txp:section link="1" title="1" wraptag="p" name="archive" />
```

[Return to tag index](#)

# Txp:section list /

## Classification

---

The `section_list` tag is a [Single Tag](#) which is used to produce a list of linked sections. Its context is page or column.

## Syntax

---

The `section_list` tag has the following syntactic structure...

```
<txp:section_list />
```

## Attributes

---

Tag will accept the following attributes (note: attributes are **case sensitive**):

`label="label text"`

Label for the top of the list. Default is empty.

`labeltag="labeltag text"`

Independent wraptag for label for the top of the list. Default is empty.

`break="tag text"`

HTML tag to be used for line breaks, without brackets. Default is empty.

`wraptag="wraptag text"`

HTML tag to be used as the wraptag, without brackets. Default is empty.

`class="class name"`

CSS class attribute, default is `section_list`.

`include_default="1 or 0"`

Include "default" section in section list. Default is empty.

## Attribute Semantics

- **label:** This string will be prepended to the link list. When using semantic markup ("`wraptag`" being either "`ol`" or "`ul`"), "`label`" will be the **first** list item.
- **break:** This string will be used to separate list items. Suggested values include "`br`" and "`hr`" for presentational markup, or "`li`" if you preferred semantic markup. Textpattern cares for the correct nesting of tags in either case. NB: Choosing `li` as the break string requires **wraptag** to be either `ul` or `ol` and vice versa.
- **wraptag:** This string will be used to enclose the whole list. Suggested values include "`p`" for presentational markup, "`ol`" or "`ul`" for semantic markup. NB: Appending additional HTML attributes like `class`, `style` or `start` is not supported.

## Examples

---

### Example 1: Display a linked section list with the label "Sections"

```
<txp:section_list label="Sections" wraptag="p" break="br" />
```

### Example 2: Display a styled section list

```
<txp:section_list break="li" wraptag="ul" />
```

Styles could go this way

```
section_list
{
  list-style-type:none;
}
```

[Return to tag index](#)

# Txp:sitename /

## Classification

---

The sitename tag is a [Single Tag](#) that returns the site's name as defined under the preferences tab.

## Syntax

---

The sitename tag has the following syntactic structure...

```
<txp:sitename />
```

## Attributes

---

This tag has no attributes

## Examples

---

### Example 1: Display the site's name as defined under the preferences tab

```
<txp:sitename />
```

[Return to tag index](#)

# Txp:site slogan /

## Classification

---

The site\_slogan tag is a [Single Tag](#) which is used to return the site slogan as entered in the site preferences. Can be used in any context.

## Syntax

---

The `site_slogan` tag has the following syntactic structure...

```
<txp:site_slogan />
```

## Attributes

---

This tag has no attributes.

"Textpattern Help" describes the `site_slogan` as follows:

This is a brief (255 chars max) tagline or description of your site.

This will be used as the description of your site wherever required, such as in your XML feeds.

## Examples

---

### Example 1: Display site slogan

```
<p><txp:site_slogan /></p>
```

[Return to tag index](#)

# Txp:site url /

## Classification

---

The `site_url` tag is a [Single Tag](#) which returns the full URL of the site as text.

## Syntax

---

The `site_url` tag has the following syntactic structure...

```
<txp:site_url />
```

## Attributes

---

This tag has no attributes.

Examples

### Example 1: Display a hyperlink to download a text file

```
<a href="<txp:site_url />download.txt">Download</a>
```

[Return to tag index](#)

# Txp:text /

## Classification

---

The `text` tag is a [Single Tag](#) which is used to display static text as defined in the tags attribute (item)

## Syntax

---

The `text` tag has the following syntactic structure...

```
<txp:text />
```

## Attributes

---

Tag will accept the following attributes (note: attributes are **case sensitive**):

```
item="text"
```

Static text to display.

## Examples

---

### Example 1: Display static text as set by the item attribute

```
<txp:text item="hometext" />
```

[Return to tag index](#)

# Txp:thumbnail /

## Classification

---

The `thumbnail` tag is a [Single Tag](#) Textpattern will replace this tag with the IMG SRC html tag matching the thumbnail image of the numeric "ID" assigned by Textpattern when the parent image was uploaded via the TXP "images" screen. Its context is article, page or column.

## Syntax

---

The `thumbnail` tag has the following syntactic structure...

```
<txp:thumbnail />
```

## Attributes

---

Tag will accept the following attributes (note: attributes are **case sensitive**):

```
id="The id assigned at upload"
```

Can be found on the Images tab.

```
name="image name"
```

Image Name as shown on the Image tab.

style="style rule"

CSS style rule.

align="HTML attribute"

IMG align attribute.

thumbnail="needed()".

poplink="1 or 0"

Set to true, image will be rendered in a popup window.

## Examples

---

### Example 1: Display the thumbnail for image id# 23

```
<txp:thumbnail id="23" />
```

[Return to tag index](#)

# Txp:title /

## Classification

---

The `title` tag is a [Single Tag](#) which is used to return the title of the article being displayed. This tag is used in an article form.

## Syntax

---

The `title` tag has the following syntactic structure...

```
<txp:title />
```

## Attributes

---

This tag has no attributes.

## Examples

---

### Example 1: Display an article title as part of an article form

```
<p><txp:title /></p><div class="post"><p><txp:author /> @ <txp:posted /><br /><txp:body /></p></div>
```

### Example 2: Display a hyperlinked title for the article being displayed

```
<txp:permlink><txp:title /></txp:permlink>
```

Related info: `<txp:permlink >`