OmniUpdate®
Empower Web Excellence

Template Reference Topics

OU Campus v10

OmniUpdate, Inc.
1320 Flynn Road, Suite 100
Camarillo, CA 93012

OmniUpdate, Inc.
1320 Flynn Road, Suite 100
Camarillo, CA 93012
800.362.2605
805.484.9428 (fax)
www.omniupdate.com

Document Number: b-025
Publish Date: 4/8/2016

About OmniUpdate, Inc.

OmniUpdate® is the leading web content management system (CMS) provider for higher education. The company focuses on providing an exceptional product and customer experience to its OU Campus™ CMS users who manage more than 700 web and mobile sites in the U.S. and around the world. OU Campus is secure and scalable, server and platform independent, and seamlessly integrates with other enterprise campus systems. It provides college and university web developers, administrators, and marketers with the user-friendly tools and deployment flexibility they need to achieve excellence. For more information, visit .

About This Guide

Includes just the reference tables for page tagging and for page templating.

OU Campus Support

The Support site is available to everyone and users are encouraged to visit and browse the site for information. An institution's administrators are also available if the answer cannot be found on the Support site or further explanation and clarification is needed. Administrators may contact the OmniUpdate Support Team. Ways to access the OU Campus support documentation include:

- Support site: http://support.omniupdate.com/
- The help link in the main interface of OU Campus
- The WYSIWYG Help link
- Help links embedded in the system
- Text instructions are provide onscreen for specific fields and functionality
- OmniUpdate Community Network (OCN): http://ocn.omniupdate.com/

Conventions

Shorthand for navigation through the OU Campus CMS is indicated with a greater-than sign and bolded: > For example, Setup > Sites. Code snippets use Courier New and a shaded background.

Contents

Page Tagging Reference

Name	Description	Works With	Notes	Values	Example
bgcolor	Optional attribute to define the background color of the editing area. The value of color can be specified by name ("white") or hex ("#FFFFFF"). This applies a background color to the editable area. The bgcolor can be very helpful when there is an existing background color that you would like to match. This only affects the editable area when viewed within OmniUpdate.	Works with	• color name — Color name like green or red. • color Hex — Hexadecimal value for a color (#FF00CC).	• color name • color Hex	bgcolor="color name | color Hex"
border	Border color of editing area (Optional): Value of color can be a hex number (such as: "#FFFFFF") or the color name (such as: "white"). The border applies		color - Color name like green or red. color Hex - Hexadecimal value for a color (#FF00CC)	• color name — Color name like green or red. • color Hex — Hexadecimal value for a color (#FF00CC).	border="color name | color Hex"

Name	Description	Works With	Notes	Values	Example
	a colored rectangle surrounding the editable area. In the Gallena example above, the border color is set to purple in for the footer area and green in the main body text area.				
break	Break after button (Optional): The break adds a carriage return (br) after the editing button. This is completely cosmetic, but can be very useful depending on the size of the button being used.				break="break"
button	Editing Graphic or Button type (Optional): Numeric value (list of buttons currently available can be found here). If the Button is not used, the default button graphic is displayed: Edit Date			• button id — The id or name of a button. A list of buttons currently available can be found here. • hide — Hides the button. This is used in conjunction with the	button="button id | hide"

Name	Description	Works With	Notes	Values	Example
	New button graphics can also be added -- feel free to make a request.			MultiEdit tag, found here.	
button-class	Provides default styling using CSS rather than using the button attribute to call a .gif.	Supported by all tagging styles.		• Takes a class defined in the CSS. E.g., class="oucEd	<ouc:div label="sidecontent" group="Everyone" **button-class="oucEditButton"** button-text="Side Content" break="break">
button-color	Changes the color of a button styled with CSS (i.e., one that does not use the button attribute).	Supported by all tagging styles.		• Hexadecimal color. E.g., button-color="#D19D	<ouc:div label="maincontent" group="Everyone" button-class="oucEditButton" button-text="Main Content" **button-color="#D19DFF"** break="break">
button-text	Changes the color of a button styled with CSS (i.e., one that does not use the button attribute).	Supported by all tagging styles.		• String: E.g.,button-text="Side Content"	<ouc:div label="sidecontent" group="Everyone" button-class="oucEditButton" button-text="Side Content" break="break">
group	Editing group that can edit this area (Optional): The group identifies which group has access to edit the specific tagged area. If you do not include a			• Everyone - All users • group name — A group name as configured within OmniUpdate'	group="Everyone | group name"

Name	Description	Works With	Notes	Values	Example
	group there will be no editing button shown — even for administrators. Level-9 and 10 users will only see the editing buttons on tagged areas that have a valid group defined. All other editors will only see the editing buttons on tagged areas that have a valid group defined — assuming they are also a member of the specified group. It is often helpful to use the default group "Everyone" for the main body area of the page, then unique groups such as "Side Navbar Editors" for the side navigation area and so on. Because the group "Everyone" contains all users you'll be able to quickly assign page			group setup.	

Name	Description	Works With	Notes	Values	Example
	access to any specific sub-group without changing the group tag in the code. Remember that only people assigned access to a page can access and edit that page — regardless of the "Group" assigned to selective areas. Note: to exclude individuals from gaining access to a specific area (such as a block of scripted code, image maps, or Java applets), tag the area and do not include the "Group". This exclusionary method is very useful for areas you want to ensure are "hands-off" to all but administrative (Level 9 and 10) users.				
hide					hide="yes | true | hide"

Name	Description	Works With	Notes	Values	Example
	When used in conjunction with the *path* parame this parameter allows for omission of included text. Omit parameter for default behavior.				
label (Required)	Name of tag (required, and must be unique per page). Never assign the same label name to multiple areas on the same page. Choose any name, but do not use any punctuation characters.			unique name or id — A unique-per-page identifier for a editable region.	label="unique name or id"
padding				pixel number — Padding in number of pixels	padding="pixel number
path	The path to the actual Include file: This must be an absolute link and begin with "/". Without the include tag you have a static area or inline area that occurs within the page itself.			staging absolute path — An absolute path relative to the root of the staging server.	path="staging absolute path"

Name	Description	Works With	Notes	Values	Example
position	Position of button (Optional): The position allows you to specify the position of the button relative to the selective content area itself. The default behavior of the button is to position itself at the top of the editable content (just above the content), but it can be useful to force the button to follow directly after the content. For instance, if the position's value is set to "bottom" and you've assigned no "break" value (see above) then the actual position of the button will be to the right following the content.				position="bottom | top"
wysiwyg-class	The attribute "wysiwyg-class" can be used with node style tagging (in either the ouc:div or ouc:editor)	ouc:div or ouc:editor tags		A defined CSS class: wysiwyg-class="ouc-editor"	<ouc:editor csspath="/_resources/ou/editor/maincontent.css" cssmenu="/_resources/ou/editor/

Name	Description	Works With	Notes	Values	Example
	tags to specify a class to be inserted into the <body> tag of the WYSIWYG Editor. In the case that both elements use the attribute, the classes are combined. In the case of any duplicated CSS classes, precedence is decided by what exists in the CSS not the ordering of the classes. In other words, <body class="dark left interior"> has the same behavior as <body class="interior left dark">.				styles.txt" wysiwyg-class="ouc-editor"/>

TCF Reference

The TCF reference page includes the syntax for and examples of the elements, attributes, and attribute values used by template control files. These include the following:

TCF-created directories inherit the parent directory access settings by default unless set in the <directory> tag. The TCF attributes for <directory> can be used to allow the user to override the default inherit for the directory being created. New pages by default inherit from the directory's access settings if created through a TCF. The TCF attributes for <template> can be used to allow the user to override the default inherit for the page being created.

As of release version 9.17.4, the following attributes were added for the <directory> element to allow the user to override the inherited value for access settings:

- approver
- publishers
- rss-feed
- toolbar
- template-group

Additionally, the following attributes were added for <template>:

- exclude-search
- exclude-sitemap

The following access settings inherit inherently and do not include an attribute for a TCF override. Access settings can be changed from the default inherit on a page-by-page basis.

- Enforce Approver
- URL Type

This change in functionality deprecates the use of `*inherit*`. It no longer needs to be set to force an inherit from the parent directory access settings. It can still be used, but is no longer necessary.

Include

Include files can be used within the TCF new page wizard files. This would be primarily helpful if there are common, repurposeable sections of TCF commands that may need to be used across many TCF files. Example: A selector list for the 50 states, or a selector list of countries.

These include files are external files which can be called by the TCF that needs to utilize that set of syntax content. There are a couple of important notes.

1. The include file must have an extension of .inc. It cannot be a .tcf extension as it will then be included in the New File template selecting page and will result in an error if chosen.
2. The include file must be in the same Templates directory as the the TCFs referencing the file/ performing the inclusion or an error will result.
3. The content of the include file must be a valid TCF command set.

Syntax

```
<include> </include>
```

Example

```
<include>filename.inc</include>
```

The Includes element has no attributes.

Variable List

Contains the child node called variable, which define variables to be used throughout the new page creation process.

Syntax

```
<variable-list> </variable-list>
```

Example

```
<variable-list>
<variable name="pagetitle" prompt="Title" alt="New document title">Untitled</
variable>
<variable name="template_path" display="no">/resources/xsl</variable>
</variable-list>
```

Attributes

This tag has no parameters.

Variable

The Variable node allows for the creation of variables to be used throughout the new page creation process. The variable tag is a child node of variable-list.

Syntax

```
<variable> </variable>
```

Example

```
<variable name="pagetitle" prompt="Title" alt="New document title">Untitled</
variable>
```

Name	Description	Works With	Notes	Values	Example
alt	Similar to the prompt parameter, alt gives more information about the input required in a text field. It's displayed below the text box, and usually offers an example or more information. This field is not required, and defaults to null/ not displayed.			string	alt="New document title"
dependency	Works with a file chooser and determines if a dependency tag can be added or not. With a value of s-tag can be used to pull content from PCF files on staging, which is an applied usage for mobile. The tag is encapsulated with double braces and prefaced with an s:, which is appended to the numerical reference.	type="filechoose for <parameter> or <variable> When the provided example is used in a TCF, the XSL can use a doc call to pull data and the concat to build the path: <xsl:copy-of select="doc(conc parameter[@nan tag']/text()))/ document/ content/ node()"/>		• s-tag: PCF returns the s-tag. Overwrites other attributes, such as "source". PCF is not expandable when s-tag is used; that is, only a staging file can be selected. Also, directory selection is disallowed. • dm-tag: Filechooser outputs the dependency tag.	<variable name="s-tag" type="filechooser" de tag" path="/" group="Ever a PCF:" alt="Select an existing PCF to get the S-tag associated with it."> </ variable>

Name	Description	Works With	Notes	Values	Example
	Note: Even if the Dependency Manager is disabled, s-tag is still valid.			• yes: Functions the same as the dm-tag value for the attribute. Included for backward compatibility. • no: Does not include a dependency tag. The root-relative path is added instead.	
display	Determines whether or not this particular variable is presented to the user for modification. If a variable is not displayed, the initial value defined between the <variable> tags defines the variable's contents. However, if the variable is displayed, its contents are considered a suggestion to the user, as they can change it freely.			• yes • no	display="yes"

Name	Description	Works With	Notes	Values	Example
editor	Applies only to type="text", which is the default. The use of this parameter on a variable node whose type is "text" will result in an input field being turned into a mini-WYSIWYG editor that can be used by the user to define the value of a variable. Must be used in conjunction with the "rows" parameter in order to define the number of rows for the element.	type="text" rows="10"		• yes -- Enables a mini-WYSIWYG editor for input field • no -- Removes the mini-WYSIWYG editor. Multi-line field is still presented. The entire attribute can be removed if no editor is needed.	<variable name="body"rows="15" type="text" editor="yes" prompt="Body" alt="Enter in the body of your article."></variable>
filter	Applies only to type="filechoose The use of this parameter on a variable node with a type of "filechooser" allows for the restriction of a user within a specified folder.	type="filechoose		file extension without a period (string)	<variable name="leftnav_include" prompt="Left Nav" alt="Choose your left nav" type="filechooser" filter="html" >/includes/navs/default.html</variable>
filtered-tags	Applies only if type="tag" is declared for the variable. Specify a list of tags and/	type="tag"		comma-separated strings of text	filtered-tags="athletics,baseball,s

Name	Description	Works With	Notes	Values	Example
	or collections that users will either be allowed to choose from in the field, or disallowed from choosing.				
filtered-tags-type	Specifies whether the filtered tags listed for the variable are the only tags available to choose from, or whether those tags are not allowed to be added to the variable.	used in conjunction with the filtered-tags attribute. This attribute is ignored if filtered-tags is not declared.		• allow -- allows only the filtered tags/ collections of tags defined in the filtered-tags attribute to be used. • disallow -- disallow all filtered tags/ collections of tags defined in the filtered-tags attribute	filtered-tags="athletics,baseball,s filtered-tags-type="allow"
format	Optional helper attribute for the date-time picker and separate date and time pickers. Currently, saved for future values, the default of type="datetime" is format="iso", which produces the following	type="datetime", type="date", type="time"		• iso	<variable name="datetime" type="datetime" format="iso" group="Everyone" prompt="Date & Time" alt="Please add a date and time." section="Date Time Example"></ variable>

Name	Description	Works With	Notes	Values	Example
	variation of ISO 8601: 12/14/2016 10:45:29 AM				
lockout	Applies only to type="filechoose and path="/some/path". The use of this parameter on a variable node whose type is "filechooser" allows for the restriction of a user within a specified folder defined by the path parameter.	type="filechoose path="/some/path"		• yes -- Restrict user to directory specified in the "path" attribute • no -- Don't restrict user to directory specified in the "path" attribute	<variable name="leftnav_include" prompt="Left Nav" alt="Choose your left nav" type="filechooser" path="/includes/navs" lockout="yes" filter="html" >/includes/navs/default.html</variable>
maxlength	Defines a specific number of characters the variable may contain. Anything over and the user will be alerted that the field can contain no more than "number" characters once the create button is pressed.		Number of characters including spaces	number	<variable name="description" prompt="Description" alt="Add a description for the page. Maximum length is 150 characters." type="text" rows="5" maxlength="150"></variable>
name	Required. The name of the variable to be used when echoing its contents into		Required.	variable name (string)	<variable name="description"></variable>

Name	Description	Works With	Notes	Values	Example
	the newly-created pages. It is used by the system as an identifier for the variable, and therefore is not seen by the end-user. This parameter is, by nature, required to be defined for each variable.				
output	Determines whether or not this variable will output xml-friendly <option> elements or a comma-separated value. This attribute is used only in conjunction with checkbox, select, and radio variable types. The XML type is mainly used for enhanced page properties.			• csv • xml	output="csv"
path	Applies only to type="filechoose The use of this parameter on a variable node with `type="fil` define the default directory for the filechooser.			staging absolute path (string) -- An absolute path relative to the root of the staging server	<variable name="leftnav_include" prompt="Left Nav" alt="Choose your left nav" type="filechooser" path="/includes/navs"> </variable>

Name	Description	Works With	Notes	Values	Example
	Users entering the filechooser will be taken to this directory.				
prompt	The question posed to the user. It is displayed on the left side of the user's input text field. A colon (:) is automatically added to the end of this prompt. This parameter is not required, and defaults to null/not displayed.			Text string	<variable name="keywords" type="text" prompt="Keywords" alt="Enter keywords for your page."> </variable>
rows	Can be applied in conjunction with editor="yes" or type="text". The use of this parameter on a variable node with `editor="yes"` or with `type="text"` defines the vertical size of the resulting field in rows of text.	editor="yes" type="text"	rows="10"	number (integer)	<variable name="description" rows="5" type="text" prompt="Summary" alt="Describe your new page in about 40 words."></ variable> <variable name="body" editor="yes" rows="15" type="text" prompt="Body" alt="Enter in the body of your article."></ variable>
section	Utilizing "section" will insert a horizontal rule above			• (space between the quotes): Including	<variable name="summary" prompt="Summary" alt="Provide an introductory

Name	Description	Works With	Notes	Values	Example
	the variable field in order to create sections during the new page creation. Entering in optional content will create a section header after the horizontal rule.			a space between quotes will create a horizontal rule above the variable in which it is entered. • optional text (string): Entering in optional text between the quotes will create a header beneath the horizontal text.	text for the body of the page." section="Body" ></variable>
source	Used in conjunction with type="filechoose Selects the default server when users enter the filechooser to browse for a file. If omitted from a variable, the default value is "source="produc	type="filechoose	Will not select alternative production targets or auxiliary sites.	Text string that matches the name of the server in OU Campus (e.g., "staging" or "production"). Text is case-sensitive.	<variable name="banner" prompt="Banner Image" alt="Select an image to appear in the banner header." type="filechooser" source="staging" dependency="dm-tag"> </ variable>
tags	if using type="tag," any tags added to this attribute will appear pre-filled in the tags field for this variable. Users will still	type="tag", type="asset"		for type="asset": To filter by site, syntax is site:sitename. To filter by type of Asset, syntax is:	type="tag" tags="tag1,tag2,tag3" type="asset" tags="enrollment,type:Pla Text"

Name	Description	Works With	Notes	Values	Example
	be able to delete these pre-filled tags while filling out the TCF. If using type="asset," this attribute will delimit the asset chooser to only show assets that have the listed tags.			type:Web Content type:Plain Text type:Source Code type:Image Gallery type:Managed Form All values are comma-separated. for type="tag": Strings of text, comma-separated.	
type	Form field type. If omitted from a variable element, type="text" is the default. Checkbox, select, or radio variables can maintain their form field type in page properties if their <option> tagging is preserved in the PCF. To do this, specify the output method as xml (output="xml"). Variable output encoding must be turned off in the TMPL file to avoid changing the <option>	output="xml" encoding="none"		• text -- Variable is simple text • select -- Variable is defined by user by choosing from a select-pulldown. Variable will be the value of the user-selected option. • filechooser -- Variable is defined by user by navigating to and selecting a file. Variable will be an	type="text" <variable name="description" rows="5" type="text" prompt="Summary" alt="Describe your new page in about 40 words."></variable> <variable name="datetime" type="datetime" format="iso" group="Everyone" prompt="Date & Time" alt="Please add a date and time." section="Date Time Example"></variable>

Name	Description	Works With	Notes	Values	Example
	nodes into plain text; to do this, use an encoding="none" attribute in the appropriate TMPL echo statement.			absolute path relative to the root of the production server. • checkbox -- Variable is defined by user by toggling the value of a checkbox. Variable will be the value of the user-selected option. • radio -- Variable is defined by user by choosing from a set of radio buttons. Variable will be the value of the user-selected option. • tag -- Presents the user with a field, into which they can enter tags for the new content. • asset -- Will present the user with an Asset Chooser.	

Name	Description	Works With	Notes	Values	Example
				• datetime -- Presents the user with a date-time picker. The variable gets passed in the following format: 12/14 10:45:29 AM and the optional format attribute is available • date -- Presents the user with a date picker. The display of the date can be altered with the format attribute. • time -- Presents the user with a time picker. The display of the time can be altered with the format attribute.	

Option

The Option tag can be used to allow for user input at the time of new page creation. Option is used within the <variable> node. Within the Option element, the Type attribute is used to specify the method of user interaction. For example, a type of select produces a drop-down menu. The possible choice of values is

defined by <option> and the value attribute with the data. The user's choice for the value is echoed onto the page. The possible values for the Type attribute are text, select, filechooser, checkbox, radio, and asset.

Syntax

```
<option> </option>
```

Example

```
<variable name="color" type="select" prompt="Choose" alt="Choose a header
color">
<option value="blue">Blue Text</option>
</variable>
```

Name	Description	Works With	Notes	Values	Example
value	The value attribute can define specific choice for users. Input can be a string for selection or true/false.	The variable element; Type attribute.		string	value="Undergraduate"
selected	Used to indicate selection choice; for example, with a variable/ type of radio or checkbox.	The variable element; Type attribute.		• true • false	selected="false"

Template List

Template List contains child nodes which define templates to be used throughout the new page creation process.

Syntax

```
<template-list> </template-list>
```

Example

```
<template-list>
  <template
    prompt-prefix="New Document"
    group="everyone"
    filename="z-breadcrumb"
    destination="{directory/}{dirname}"
    display-group="yes"
    preferred-redirect="yes"
    publish="no"
    extension="pcf"
    >z-breadcrumb.tmpl</template>
  <template
    group="everyone"
    filename="index"
    display-group="no"
    display-destination="no"
    display-filename="no"
    display-overwrite="no"
    preferred-redirect="yes"
    publish="no"
    extension="pcf"
    destination="{directory/}{dirname}"
    force-destination="yes"
    >newpcf.tmpl</template>
</template-list>
```

Attributes

This element does not contain attributes.

Template

Defines a template file (TMPL) and relevant parameters to be used during new page creation process. Files created from a TCF by default inherit parent directory values unless set with new values in the <template> tag.

Syntax

```
<template> </template>
```

Example

```
<template
prompt-prefix="New Document"
group="everyone"
filename="untitled"
display-group="yes"
rss-feed="*inherit*"
preferred-redirect="yes"
publish="no"
extension="pcf"
destination="{directory/}{dirname}"
>{selected_template}</template>
```

Name	Description	Works With	Notes	Values	Example
approver	Allows certain pages to have an assigned approver. This is helpful for different sections of a site which require that different people approve content. A page approver overrides an approver set on the user.			• None -- No approver is set for the page. • string user name -- User name as defined within the OU Campus CMS.	approver="None"
autonav	Used in conjunction with the Navigation tag.	Navigation		navigation name (string)	autonav="leftnav"
destination	Will set a location for new files created other than the current folder when Create New Page was clicked.			absolute path (string)	destination="{dirname}"

Name	Description	Works With	Notes	Values	Example
directedit	Utilize "directedit" to repress the creation of a DirectEdit com.omniupdate tag. This will override the settings of "Standard" and "Transparent" in the Site settings. The purpose of this attribute is when files other than pages are being created, such as .css, .inc, or .doc files. It is also useful when creating templates from templates (using a TCF to create a new TMPL).			• yes • no	directedit="no"
display-filename	Determines whether a user can fill in a filename for a new file that is being created. This is desirable except when creating a resource file for a new folder; for example, a navigation file.			• yes • no	display-filename="yes"
display-group	Displays a drop down		When including a	• yes • no	display group="yes"

Name	Description	Works With	Notes	Values	Example
	list of groups that the user belongs to in the new page screen and allows the user to assign new pages to one of those groups. If the display group is set to "no", the user has no control over which group the new page is assigned.		<template> node in a TCF and the attribute display-group="no" is explicitly defined and the group attribute is NOT explicitly defined, then the group access settings should inherit from the folder. If the display-group attribute is omitted or explicitly defined as yes, then the resulting drop-down defaults to Everyone instead of inherit. In this case, to ensure the drop-down defaults to Inherit, the group attribute must be explicitly defined to use *inherit*. Note that the use of *inherit* is considered deprecated, but is valid usage in this case.		
display-overwrite	Determines whether users			• yes • no	display-overwrite="yes"

Name	Description	Works With	Notes	Values	Example
	can overwrite existing files. By setting the value to "no", users cannot overwrite any existing pages.				
exclude-sitemap	Attribute can be used to override the default of directory inherit for excluding from sitemap.			• yes • no	exclude-sitemap="no"
exclude-search	Attribute can be used to override the default of directory inherit for excluding from search.			• yes • no	exclude-search="no"
extension	The default file extension of the resulting file. Use in conjunction with filename attribute to specify file filename of resulting file. The following example produces a file named index.pcf	filename attribute		file extension (string)	filename="index" extension="pcf"
filename	The default name of the resulting file without file extension. Use in conjunction with extension	extension attribute	The user should not leave a filename field blank. If the filename before the	file name (string)	filename="index" extension="pcf"

Name	Description	Works With	Notes	Values	Example
	parameter to specify file extension of resulting file. The following example produces a file named index.pcf		extension is blank, the system provides an error notification. Whe the TCF form is saved, and it would create a blank filename, then the message is: "Please enter a file name."		
filename-alt	The default description for the filename prompt. The following example replaces the default description that shows up below the filename box.			description (string)	filename-alt="Please enter a filename"
force-destination	If yes, creates folders or directories if they do not exist already and must be used with the destination parameter.			• yes • no	force-destination="no"
group	Determines the editing group that is assigned to new files.		When including a <template> node in a TCF and the attribute display-group="no" is explicitly	• Everyone -- The "everyone" group. This means anyone can edit the file. • string group name --	group="Everyone"

Name	Description	Works With	Notes	Values	Example
			defined and the group attribute is NOT explicitly defined, then the group access settings should inherit from the folder. If the display-group attribute is omitted or explicitly defined as yes, then the resulting drop-down defaults to Everyone instead of inherit. In this case, to ensure the drop-down defaults to Inherit, the group attribute must be explicitly defined to use *inherit*. Note that the use of *inherit* is considered deprecated, but is valid usage in this case.	Group name as defined within the OmniUpdate system.	
name	Uniquely identifies template. If prompt-prefix is omitted, the value of this parameter will be used.	prompt-prefix		unique identifier (string)	name="defaultpage"

Name	Description	Works With	Notes	Values	Example
overwrite	If yes, creates a new page of the chosen filename after erasing any existing file of the same name.			• yes • no	overwrite="no"
preferred-redirect	More than one file can be created using a TCF. This parameter specifies that the page created with this particular TMPL file will be the one to which the user is redirected for immediate editing, once the pages have been created. The default for this parameter is "no." If this parameter is not set to "yes" anywhere in a TCF, the user will be presented with a report of files created, and whether they were published. This is useful when creating multiple files at once; for example, when creating a new section			• yes • no	preferred-redirect="yes"

Name	Description	Works With	Notes	Values	Example
	as it allows for to render the index.pcf in edit mode once the entire section has been created.				
prompt-prefix	Because the TCF format allows for multiple files to be created and referenced, the prompt-prefix parameter helps to identify the file to which a question refers. OU Campus adds the prompt-prefix to each question being asked. As an example, if prompt-prefix is "New file," the user may be asked "New file overwrite if exists?"			string	prompt-prefix="New file"
publish	Normally, when a file is created from a TCF, it is created on the staging server, awaiting a publish command from the user. However, there are times when the TCF author would like one or more		Can be used with the targets attribute.	• yes • no	publish="yes"

Name	Description	Works With	Notes	Values	Example
	of the pages created by the TCF to be automatically published to the production server upon creation. This parameter automatically publishes the specified page when it is created. This is very helpful for navigation files when creating new sections.				
publishers	Allows certain pages to have assigned publishing groups that override the approval process.			• None -- Indicates that no Publishers group is assigned. • string group name -- Group name as defined within the OU Campus system.	publishers="None"
rss-author	Allows the author of an RSS feed item to be attached to a different user.			string	rss-author="{some_variable}"
rss-description	Allows for the population of the RSS item's description node with			• string description -- Allows for a customized	rss-description="*inherit*"

Name	Description	Works With	Notes	Values	Example
	content provided by a TCF variable.			description to be added for each RSS item.	
rss-extra	Allows for the population of the RSS item node with additional XML.			• string xml nodes -- A string of XML nodes to be added to each RSS item.	rss-extra="*inherit*"
rss-feed	Determines the feed to which the newly created page will belong. By default, new pages do not have an RSS feed assigned to them and this property allows the page to automatically be assigned to an RSS feed.			• string feed path -- Absolute path relative to production server root as defined within the OU Campus System.	rss-feed="*inherit*"
rss-link	Allows the RSS link to be provided by a TCF variable.			• string url - Specifies the link associated with each RSS item.	rss-link="{var_name}"
rss-media	Allows for the population of the RSS item node with additional XML.			• Delimited List - A delimited string defining the number of Media RSS items and the value of each item parameter.	rss-media="{mtitle1}^^{murl1} {mtitle2}^^{murl2}^^{mdes {mtitle3}^^{murl3}^^{mdes

Name	Description	Works With	Notes	Values	Example
				The double-het (^^) delimiter separates each Media RSS parameter, and each double-bar (||) separates each Media RSS item.	
rss-pubdate	Allows for the population of the RSS item's pubdate node with content provided by a TCF variable.			• string date -- A customized date that can be added in manually.	rss-pubdate="[auto]"
rss-tags	Allows for tags to be added to the RSS item.		Typically, tags will be declared in a variable in the TCF, the value of which will be passed into this attribute.	• list of tags, comma-separated.	rss-tags="athletics,baseball,c
rss-title	Allows for the population of the RSS item's title node with content provided by a TCF variable.			• string title -- Specifies the title of each RSS item.	rss-title="*inherit*"
tags	Allows for tags to be added to the new page.		Typically, tags will be declared in a variable in the TCF, the value of which will be passed into this attribute.	• list of tags, comma-separated.	tags="{tags},athletics,bas

Name	Description	Works With	Notes	Values	Example
			Tags can also manually added into this attribute.		
targets	Used in conjunction with publish to specify publish target(s) for file auto publish when it is created via the template structure. If not used, production server is the default publish target.		publish="yes"	• server names delimited with a comma (string)	publish="yes" target="dev,site_name"
toolbar	Determines the toolbar used for a particular page. This overrides the user's own toolbar assignment and is helpful when creating navigation files and the full toolbar should never be used on a page, regardless of the user editing the file. The toolbar group can also be found in the editor tag and the directory tag. The hierarchy is users, page and then editor tag for the			• Default • string toolbar group name -- Toolbar Group name as defined within OU Campus system.	toolbar="Default"

Name	Description	Works With	Notes	Values	Example
	toolbar that will be available to an editing area. This parameter is only necessary to override toolbar groups for pages, irrespective of user assigned toolbars.				

Directory List

Directory List contains the Parent and Directory child nodes and is used to create an empty directory in a new section. For example: /artdepartment/images.

Syntax

```
<directory-list>
<parent path="{directory/}{sectionname}">
<directory name="images">images</directory>
</parent>
</directory-list>
```

Example

```
<directory-list>
<parent name="current_folder" force-lowercase="no" path=".">
<directory name="images" prompt-prefix="New Section" group="*inherit*"
rss-feed="*inherit*" force-lowercase="yes" display-group="yes" template-
group="*inherit*" publish="yes">{dirname}/images</directory>
</parent>
</directory-list>
```

Parent

Parent is a child node of Directory-List, and contains the Directory node.

Syntax

```
<parent></parent>
```

Example

```
<directory-list>
<parent name="current_folder" force-lowercase="no" path=".">
<directory name="images" prompt-prefix="New Section" group="*inherit*"
rss-feed="*inherit*" force-lowercase="yes" display-group="yes" template-
group="*inherit*" publish="yes">{dirname}/images</directory>
</parent>
</directory-list>
```

Name	Description	Works With	Notes	Values	Example
name				string	
force-lowercase				• yes • no	force-lowercase="no"
path (required)				• . • /path_name(/string)	

Directory

Directory is a child node of Parent, which is contained within Directory-List and used in the creation of a new section with an empty directory.

Syntax

```
<directory name="images">images</directory>
```

Example

```
<directory-list>
<parent name="name_path" force-lowercase="yes" path="/images">
<directory name="images_path" force-lowercase="yes" group="*inherit*"
publish="yes">{directory/}{dirname}/rotator</directory>
</parent>
</directory-list>
```

Name	Description	Works With	Notes	Values	Example
approver	Attribute can be used to override the default of directory			• None -- No user is defined as an	approver="Maddy"

Name	Description	Works With	Notes	Values	Example
	inherit for the approver. An approver is a user, not a group.			approver at this level. • string user name -- User name as defined within the OU Campus system.	
exclude-search	Attribute can be used to override the default of directory inherit for excluding from search.			• yes • no	exclude-search ="no"
exclude-sitemap	Attribute can be used to override the default of directory inherit for excluding from sitemap.			• yes • no	exclude-sitemap ="yes"
filtered-tags	Used to set filtered tags for the new directory.			• strings of text for each tag, comma-separated	filtered-tags="athletics,baseball,f
filtered-tags-type	This attribute modifies whether the filtered tags listed in the same directory node are allowed or disallowed.	only functions if the filtered-tags attribute is defined		• allow • disallow	filtered-tags-type="allow"
fixed-tags	Used to set fixed tags for the new directory.			• strings of text for each tag, comma-separated	fixed-tags="athletics,gallena"

Name	Description	Works With	Notes	Values	Example
force-lowercase	Used with name to create lower case names.			• yes • no	force-lowercase="yes"
group	Determines the editing group that is assigned to new files. If not set, then inherits group from parent directory.			• Everyone -- The "Everyone" group. This means anyone can edit the file. • string group name -- Group name as defined within the OU Campus system.	group="Everyone"
name	The name for the directory to be created.	force-lowercase		• string name	name="images"
publish	If yes, pushes a file to the production server for publishing prior to editing. This is very helpful for navigation files when creating new sections.	Can be used with the targets attribute.		• yes • no • target	publish="yes"
publishers	Attribute can be used to override the default of directory inherit for the publisher group. The			• None -- Indicates no Publisher group is set.	publishers="None"

Name	Description	Works With	Notes	Values	Example
	Publisher group has the ability to bypass the approval process and publish immediately.			• Everyone -- The "Everyone" group. This means anyone can override the approver setting. • string group name -- Group name as defined within the OU Campus system.	
prompt-prefix	Because the TCF format allows for multiple files to be created and referenced, the prompt-prefix parameter helps to identify the file to which a question refers. OU Campus adds the prompt-prefix to each question being asked. As an example, if prompt-prefix is "New file," the user may be asked "New file overwrite if exists?"			string	prompt-prefix="New Section"
rss-feed	Determines the feed to which				rss-feed="*inherit*"

Name	Description	Works With	Notes	Values	Example
	the newly created page will belong. By default, new pages do not have an RSS feed assigned to them and this property allows the page to automatically be assigned to an RSS feed.			• string feed path -- Absolute path relative to production server root as defined within the OU Campus System.	
display-group	Displays a drop down list of groups that the user belongs to in the new page screen and allows the user to assign new pages to one of those groups. If the display group is set to "no", the user has no control over which group the new page is assigned.			• yes • no	display-group="yes"
template-group	By default when not specified, makes new folders created by a TCF inherit the template group from the parent folder. This is desirable when creating new folders in a section of			• None -- In this context None specifies that all templates in the templates directory are available for page creation.	template-group="None"

Name	Description	Works With	Notes	Values	Example
	the site which has specific templates. When a template-group is specified it limits the available templates to those in the group.			• string template-group -- template-group name as defined in the OU Campus CMS and used to limit a directory using only specific templates.	
toolbar	Attribute can be used to override the default of directory inherit setting for the WYSIWYG toolbar. This parameter is only necessary to override toolbar groups for pages, irrespective of user assigned toolbars.			• Default -- Default toolbar for the WYSIWYG Editor. • string toolbar name -- Name of toolbar as defined within the OU Campus system by an administrator	toolbar="Default"

Use with Predefined Variables

Directory can take a value that uses a variable and/or a string to specify a path location.

Examples

```
{directory/}{dirname}/rotator
```

```
{dirname}/images
```

Navigation List

Contains child nodes which define navigation to be used throughout the new page creation process.

Syntax

```
<navigation-list> </navigation-list>
```

Example

```
<navigation-list>
<navigation name="leftnav" path="subnav.inc" group="everyone" publish="yes">
<li>
<a href="{ox_autonav:shorturl}">
{navtitle}
</a>
</li>
</navigation>
</navigation-list>
```

This tag has no attributes.

Navigation

The Navigation element is a child node of the navigation-list element. It contains the structure of a new navigation item to be rendered and appended to the file specified via the path attribute.

Syntax

```
<navigation> </navigation>
```

Example

```
<navigation name="leftnav" path="subnav.inc" group="everyone" publish="yes">
<li>
<a href="{ox_autonav:shorturl}">
{navtitle}
</a>
</li>
</navigation>
```

Name	Description	Works With	Notes	Values	Example
group	Determines the editing group that is assigned to new navigation files. The "*inherit*" value assigns			• *inherit* -- Inherits group from directory access privileges. • everyone -- The	group="*inherit*"

Name	Description	Works With	Notes	Values	Example
	the new navigation file to the same group as that of the folder where the new navigation file is located.			"everyone" group. This means anyone can edit the file. • string group name -- Group name as defined within the OmniUpdate system.	
name	Navigation identifier used in correspondence with the template tag's autonav attribute.	The Template element's autonav attribute.		string	name="leftnav"
path	Path to navigation file into which the new navigation item should be rendered.			string	path="subnav.inc"
publish	If yes, pushes a file to the production server for publishing prior to editing.			• yes • no	publish="no"

TMPL Reference

The TMPL file is an XML file that typically includes XML, XHTML, OU Campus tagging, and the Echo Variable and Set Variable elements. Predefined variables can be used in *.tmpl* files to echo, or print, values relating to, for example, directory names, includes, site name, and user-provided input. This allows the TCF to pass a value and the TMPL to use it in their combined effort to manufacture a PCF. These predefined system variables are particularly handy to dynamically create a path to the root of the site or any other area of the site. They can be useful for utilizing recursive includes for, say, a breadcrumb trail.

At a general level, a TMPL is simply a PCF with echo variables in place of hard-coded content. This page will cover echo variables and other elements unique to a TMPL file (as opposed to elements such as OU Campus and Page Properties tags that are present in both TMPLs and PCFs). For more information about OU Campus tags found in PCFs and TMPLs, please visit the following pages:

- Publish Control Files (PCF)
- Properties Tags Overview

The major headings on this page are:

Echo Variable Tag Overview

The value of a variable gets echoed or printed into a new file being created. Note that there is *not* a space between the first comment tag and the percent sign.

Syntax

```
<!--%echo var="current" -->
```

Example

```
<meta name="keywords" content="<!--%echo var="current" -->">
```

Name	Description	Works With	Notes	Values	Example
encoding	This attribute provides the ability to turn encoding off.	<variable> that includes output="xml"	Example from the TCF: <variable name="quotes"	• "none"	encoding="none"

Name	Description	Works With	Notes	Values	Example
	If this attribute is not set, encoding is assumed to be on. If the variable, defined in the TCF or otherwise, has the output attribute set to "xml" then encoding is required to be set to none in the TMPL for the feature to return the proper value.		type="select" group="Everyone prompt="Student Quotes" alt="Do you want to display the student quotes" output="xml"> <option value="true" selected="true"> option> <option value="false" selected="false"> option> </variable> Corresponding example from the TMPL: <parameter name="quotes" type="select" group="Everyone prompt="Student Quotes" alt="Do you want to display the student quotes"> **<!--%echo var="quotes"** encoding="none" --> </parameter>		
var	The name of the variable which should be printed into the new document. Variable can be defined by a TCF or be	encoding="none if applicable		• system_varia • tcf_defined_v	var="variable name"

Name	Description	Works With	Notes	Values	Example
	a predefined variable.				

More about Encoding and Output

When creating user-input elements in the TCF or page parameters, the attribute "type" is used within the <variable> node to specify the type of element. Some types can include options. It's the value of the option specified by the user that is used to determine something about the page content or page configuration. Types that can include options are the checkbox, select (pick list), and radio (radio buttons).

These variable types (checkbox, select, radio) can maintain the form field type (rather than getting converted to text) if <option> tagging is preserved in the PCF. To do this, specify the output method as xml (output="xml") in the TCF variable definition. The corresponding node in the TMPL must include in its echo var statement the attribute to turn off output encoding to avoid changing the <option> nodes into plain text. To do this, use encoding="none" in the appropriate TMPL echo statement.

This outputs all of the options instead of just the value from a TCF variable to a TMPL.

The combination of output="xml" and encoding="none" will output all of the option nodes where the echo statement is within a TMPL file.

Variable Reference

Current Date Variable

The current date variable outputs the current date and time at the time of page creation.

Syntax

```
<!--%echo var="currentDate" -->
```

Example Usage

```
<parameter name="datetime" prompt="Date created" alt="Date created.">
<!--%echo var="currentDate" --></parameter>
```

Example Output

```
<parameter name="datetime" prompt="Date created" alt="Date created.">Wednesday,
November 27, 2013 8:59:12 AM PST</parameter>
```

Common Tree Variable

Prints the name of the current directory.

Syntax

```
<!--%echo var="commontree" -->
```

Example

```
<a href="/<!--%echo var="commontree" -->/index.html">
```

Current Variable

Prints a path to the current directory from the root of the site.

Syntax

```
<!--%echo var="current" -->
```

Example

```
<meta name="keywords" content="<!--%echo var="current" -->">
```

Directory Variable

Prints a path to the current directory from the root of the site, followed by a slash.

Syntax

```
<!--%echo var="directory/" -->
```

Example

```
<a href="/<!--%echo var="directory/" -->index.html">
```

Includes Variable

Prints the path from the site root to the parent directory and adds a trailing "includes" directory (i.e., "currentdir/includes").

Syntax

```
<!--%echo var="includes" -->
```

Example

```
<a href="<!--%echo var="includes" -->/leftnav.html">
```

Parent Variable

Prints the path from the site root to the directory that contains the current directory (i.e., one directory "above" the current location).

Syntax

```
<!--%echo var="parent" -->
```

Example

```
<a href="/<!--%echo var="parent" -->index.html">
```

Relative Root Variable

Prints the relative path from the current directory to the staging root (i.e., "../../currentdir/"). This variable is particularly useful when used in PCFs or utilized in XSL transformations.

Syntax

```
<!--%echo var="relativeRoot/" -->
```

Example

```
<config><!--%echo var="relativeRoot/" -->resources/config/standard.xml</config>
```

Site Name Variable

Prints the name of the current site (determined by an administrator).

Syntax

```
<!--%echo var="sitename" -->
```

Example

```
This page is part of the <!--%echo var="sitename" --> site.
```

Set Variable Tag

Defines a new variable within a TMPL.

Syntax

```
<!--%set -->
```

Example

```
<meta name="keywords" content="<!--%set var="metakey" -->">
```

Name	Description	Works With	Notes	Values	Example
var (Required)	The name of the variable which should be printed into the new document. Variable can be defined by a TCF or be a predefined variable.				var="variable name"

More about Defining Variables

Group Tag

To set editing access for a page created by a new page template:

```
<!-- com.omniupdate.meta group="Everyone" -->
```

Note: "Everyone" is a standard group available in all accounts. Any other account-specific group name may also be used.

Filename Tag

To set a default file name for new pages created by templates:

```
<!-- com.omniupdate.meta filename="index" -->
```

Extension Tag

To set a default extension to a file name for new pages created by templates:

```
<!-- com.omniupdate.meta group="Secondary Nav" filename="rightnavlinks"
extension="inc" -->
```

Echoing Variables

User-Defined Echo Tags

To echo any user-defined variable as defined above, simply use the following format:

```
<title><!--%echo var="title" --></title>
```

Meta Echo Tags

To echo any of the default meta fields from a non-TCF defined new page form, use the following format:

```
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<meta name="keywords" content="<!--%echo var="metakey" -->">
<meta name="description" content="<!--%echo var="metadesc"-->">
```

Title Echo Tag

Tthe following format can be used to include the page title within the body of the page:

```
<!--%echo var="title" -->
```

Other System-Defined Echo Tags

How to write a path within the same directory:

1. First, declare a variable to use as follows:

```
<!--%set var="myvarname" directory="current" file="_section-path.html"-->
```

2. Second, use the new variable as follows:

```
<!-- com.omniupdate.div label="breadcrumb-preview" group="z-nav-section-name"
path="<!--%echo var="myvarname" -->" button="125" position="bottom" -->
```

To write a path up one directory:

1. First, declare a variable to use as follows:

```
<!--%set var="myvarname" directory="parent" file="_section-path.html"-->
```

2. Second, use the new variable as follows:

```
 <!-- com.omniupdate.div label="breadcrumb-preview" group="z-nav-section-name"
path="<!--%echo var="myvarname" -->" button="125" position="bottom" -->
```