



Human Interaction Guidelines

Version 0.31

© 2014 CrownPeak Technology, Inc. All rights reserved. No part of this document may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopy, recording, or any information storage and retrieval system, without permission from CrownPeak Technology.

Document History

Author/Editor	Date	Reason for Change	
Denise Duncan	9/2/2014	Draft	0.10
Justin Neely	9/14/2014	Revised Draft	0.20
Denise Duncan	9/14/2014	Accepted revisions, additional edits	0.30
Denise Duncan	9/22/2014	Changed names in sample folder	0.31

Table of Contents

Human Interaction Guidelines	1
Document History.....	2
Human Interaction Guidelines Overview	5
Benefits of these Guidelines	5
Content Organization Guidelines	6
Recommended folder structure	6
Definitions.....	6
CMS Site folders	6
Hosting Webserver	7
CMS Configuration	7
Managing Shared Files	7
Input Configuration Guidelines	9
Input Field Options.....	9
Using WYSIWYG CSS.....	9
Using WYSIWYG Library Files and Services Library	9
Using Default Field Values.....	10
Alt Tags for Images	10
Using Help Messaging	11
Using List Panels.....	12
Using Horizontal Panels.....	13
Removing Secure Pop-ups.....	14
Linking Images in CSS	15
Creating Conditional Output Code	15
Adding ICE Manually	17
Examples	17
When to Override ICE Using .Raw	18
How and when to configure a model.....	19
Optimizing Asset.Load.....	21
Using Filters and Lists	21
Disabling Dependencies	21
Reducing Dependencies	21
Disabling Preview Cache	22

Editing Publishing Properties22

Clearing Publishing Errors23

Notes and Additional Information 24

Notes24

Panels25

Basic form fields26

Selector and Upload Fields.....29

Drop-down Menus30

Display Only.....31

Human Interaction Guidelines Overview

What makes a great authoring experience in CrownPeak?

There are a few areas to focus on when trying to configure a great user experience in CrownPeak.

Content Organization

Enable users to easily find, create and edit content based on type and site

Input Configuration

Take advantage of various input template fields and best practices

Preview

Configure preview to match the published site. This usually only requires a few steps.

In Context Editing (ICE)

Enable users to edit content from preview by verifying the ICE configuration

Models

Enable users to create pages, sections, and sites based on site and role

Template performance

Make sure templates are configured to perform well for a small or large group of assets.

Goals

Configure CrownPeak and the content author's experience with a few goals in mind

- Simple: Make it easy to learn, easy to understand
- Useful: Make common, frequent tasks easy to repeat.
- Deep: Make sites and layouts modifiable so content editors have maximum flexibility

Benefits of these Guidelines

- Consistent user experiences
- Implementations designed and configured according to best practices that are easier to maintain, troubleshoot and update
- Happy customers

Content Organization Guidelines

These configuration guidelines to make it easy for content authors to share content and navigate through CrownPeak.

Recommended folder structure

Recommendation on folder structure, web server and package configuration. This provides the most flexibility to add additional sites, sub-domains or sites for device types

Definitions

- **Site collection:** sites related by domain name (i.e. locale, device) or common groupings (microsites, BU)
- **Domain:** *.domainname.*
- **Package:** Configured by Domain

CMS Site folders

Recommendation:

```
/Global/  
    /Assets  
    /Project1  
        /FileNames  
        /Library  
        /Templates  
/Site Collection 1/  
    /DomainRoot 1A/  
        /Project [R&D recommendation]  
/DomainRoot 1B  
    /Project [R&D recommendation]  
/Site Collection 2/  
    /DomainRoot 2A/  
    /DomainRoot 2B/
```

¹ Note: Projects will be available in November 2014 release.

Example

/Global/

/AdventGeneral

 /AdventGeneral.com

 /AdventGeneral.com+m (*publishing desktop and mobile site*)

 /AdventGeneral.m (*publishing just mobile site*)

 /AdventGeneral.uk

/AG Annuities

 /AGAnnuities.com

 /AGAnnuities.uk

Hosting Webserver

Recommendation:

/State/Site collection/Domain

Examples

- /Live/AdventGeneral/Global/
- /Live/AdventGeneral/AdventGeneral.com
- /Live/AdventGeneral/AdventGeneral.uk

CMS Configuration

Recommendation:

Models

/System/Models/Site collection/Desktop

/System/Models/Site collection/Mobile

Templates

/System/Templates/Site collection/Desktop

/System/Templates/Site collection/Mobile

Library (by naming conventions. Alternately, create project for site-specific projects)

/System/Library/Site collection/Desktop

/System/Library/Site collection/Mobile

Managing Shared Files

CrownPeak configuration should allow content authors and editors to define assets globally across all sites or locales.

Files that will be shared by more than one site should be stored in a Global folder available to all content authors for all sites.

Localized sites should have their own Asset folders for times when images, css or js files need to be localized as well. It is highly recommended that assets that do not need to be shared are also managed locally.

Input Configuration Guidelines

The input form is the content author's primary interaction with CrownPeak and their content. Here are guidelines for input and field configuration.

Input Field Options

Sample code for all input field types is available here -

https://connect.crownpeak.com/developers_and_partners/getting_started/documents/developers/code-samples/input.aspx

See addendum at the end of this document for screenshots.

Using WYSIWYG CSS

Our WYSIWYG editors - Ephox and TinyMCE - allow developers to provide specific CSS to the content authors in order to style their content. Best practice requires creating a dedicated WYSIWYG CSS to set up CSS styles to select from.

You can attach CSS to the WYSIWYG editors by setting the following parameter either locally in the input or globally in the library class or locally in a given template's input.aspx file:

```
fullParams.Stylesheet = "<cms path to the CSS file>";
```

You can also set a stylesheet that will be used by the WYSIWYG field for its preview using this parameter:

```
fullParams.PreviewStylesheet = "<cms path to the CSS file>";
```

Using WYSIWYG Library Files and Services Library

Using WYSIWYG controls requires developers to configure a long list of parameters. Best practice is to create a library class in the *System/Library* folder to contain these WYSIWYG configuration parameters. A sample library class, *WYSIWYGConfig.cs*, is included in the */System/Library* folder in the training instance.

Pre-configured parameters for compressed, medium, and full WYSIWYG configurations are available in the Services Library in the CrownPeak API. You can include them with this code.

Compressed

```
WysiwygParams wParams = ServicesInput.CompressedWYSIWYG();
```

Medium

```
WysiwygParams wParams = ServicesInput.MediumWYSIWYG();
```

Full

```
WysiwygParams wParams = ServicesInput.FullWYSIWYG();
```

Using Default Field Values

When creating new content from a Model, the content author will be prompted to set the initial asset label. The content of this field should be captured into a field by setting the default value to the `asset.label` property.

```
Input.ShowTextBox("Page Title", "page_title", width: 50, defaultValue: asset.Label,
helpMessage: "this is a help message");
```

Additionally, in the `post_input` template file you should update the asset's label property when the page title changes. Use this Services Library to check and update the label.

```
ServicesPostInput.SetAssetLabel(asset, context, "page_title", "Page Title");
```

Alt Tags for Images

Input templates with required images should also require an ALT tag for each image. Here is an example using a library method called `SetImage`.

InputHelper.cs

```
public static void SetImage(
    string _imagePath,
    string _field,
    string imageTitle = "Image",
    string altTitle = "Alt Text",
    string imageHelpMessage = "",
    string imagePopupMessage = "",
    string altDefaultValue = "",
    string altHelpMessage = "",
    string altPopupMessage = ""
)
{
    ShowAcquireParams sapImg = SetSAP(_imagePath);
    Input.ShowAcquireImage(imageTitle, _field + "_img", sapImg, helpMessage:
imageHelpMessage, popupMessage: imagePopupMessage);
    Input.ShowTextBox(altTitle, _field + "_alt", defaultValue: altDefaultValue,
helpMessage: altHelpMessage, popupMessage: altPopupMessage, width:50);
}
```

Input.aspx

```
R2_InputHelper.SetImage(R2_InputHelper.imageFolder, "two_column_image");
```

Output.aspx

```
GetImage giImage = new GetImage(asset, "two_column_image");
Out.Write(string.Format("<img src=\"{0}\" alt=\"{1}\">", giImage.ImagePath,
giImage.AltText));
```

Using Help Messaging

Help messages should be consistently applied in all template input files. You can include three types of help messaging

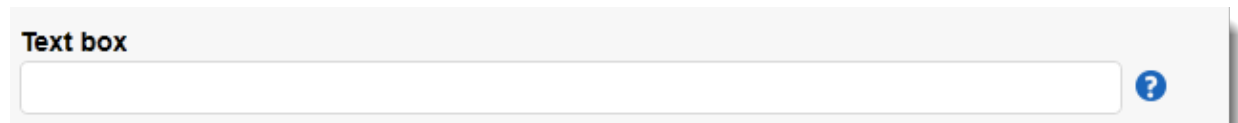
Help text can be added below input fields using this optional parameter.

```
Input.ShowTextBox("Page Title", "page_title", width: 50, defaultValue: asset.Label,
  helpMessage: "this is a help message");
```



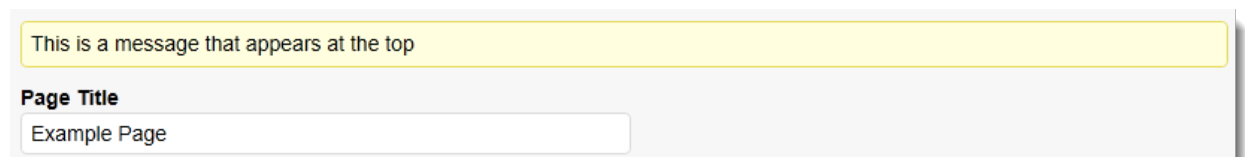
A help icon with a message that appears on rollover can be added next to a field by using the optional *popupMessage* parameter.

```
Input.ShowTextBox("Text box", "text box", popupMessage: "Popup help message is
  available for most Basic and Selector input fields");
```



Help messages highlighted in yellow can be added anywhere on an input form. Place this code wherever you'd like the message(s) to appear.

```
Input.ShowMessage("This is a message");
```



Using List Panels

INCORRECT: Not the right way add sorting

Sort Order
Collapse

Launcher Region

Select a sort priority for this folder. Note: folders with the same priority will default to alphabetical order. Debug: CurrentNode - Landing Page - Test ||| Property Key - Launcher RegionSortPriority

Massachusetts

Select a sort priority for this folder. Note: folders with the same priority will default to alphabetical order. Debug: CurrentNode - Landing Page - Test ||| Property Key - MassachusettsSortPriority

New York

Select a sort priority for this folder. Note: folders with the same priority will default to alphabetical order. Debug: CurrentNode - Landing Page - Test ||| Property Key - New YorkSortPriority

Rhode Island

Select a sort priority for this folder. Note: folders with the same priority will default to alphabetical order. Debug: CurrentNode - Landing Page - Test ||| Property Key - Rhode IslandSortPriority

CORRECT: A sortable list created with panels

List Panel: List Panel
Collapse

Name

Telephone

Name

Telephone

Name

Telephone

List panels give content authors the option to create lists with unlimited items that can be reordered.

Use this code to implement them.

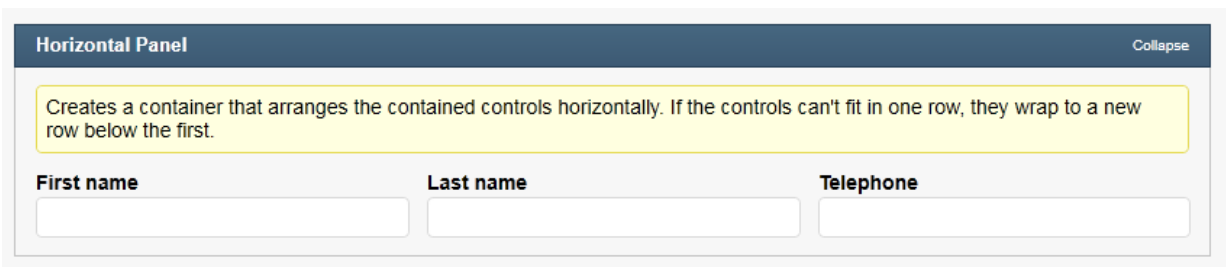
```
while (Input.NextPanel("panel_var_2", displayName: "List Panel"))
{
    Input.ShowTextBox("Name", "name_2");
    Input.ShowTextBox("Telephone", "telephone_3");
}
```

More information is available here

http://help.crownpeak.com/cmsapi/index.aspx?topic=html/M_CrownPeak_CMSAPI_Input_NextPanel.htm

https://connect.crownpeak.com/developers_and_partners/solutions/solutions_reference_guides/documents/design-patterns-best-practices/design_patterns/introductiontoclistpanelspdf

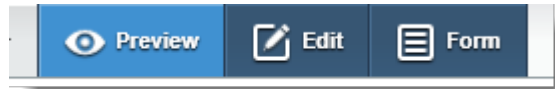
Using Horizontal Panels



To add more than one field to a line, use horizontal panels.

```
Input.StartHorizontalWrapContainer();
Input.ShowTextBox("First name", "fname_2");
Input.ShowTextBox("Last name", "lname_2");
Input.ShowTextBox("Telephone", "telephone_4");
Input.EndHorizontalWrapContainer();
```

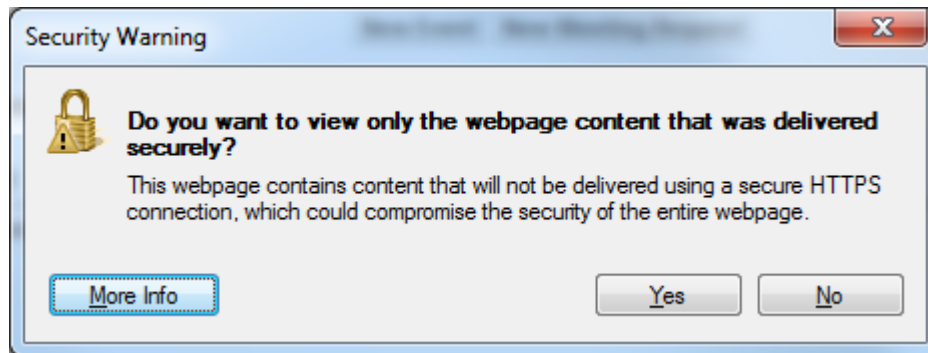
Preview Configuration Guidelines



Be sure to test and fully enable the Preview functionality in CrownPeak to give your content authors a way to preview their content in the site's look and feel without requiring publishing. Previews typically fail for simple reasons like failing to link to CSS or JS files using the correct syntax.

Removing Secure Pop-ups

"Where is this popup coming from and what can I do about it?"



The CrownPeak Preview uses HTTPS. Content in mixed mode can produce popups in Preview asking users whether or not they want to display "only secure content." This message could be produced by something as simple as an HTTP link out to a Google-hosted jQuery library.

Remedies include:

- Using //
 - First, confirm the external site supports both https and http.
 - If so, change "http://" to just "//"
 - // will inherit protocol from the current page
- Importing the files causing the error, if possible
 - See items below for steps to configure links.
- Using context.isPublishing blocks to hide code that will not be used in CrownPeak, like server-side code
- Troubleshooting specific elements: Testing can be done in Volte using tools like Firebug or Fiddler. Here are links to some debugging tips available on Connect.
 - Debugging with Firebug Lite - <https://connect.crownpeak.com/blogs/developers/debugginginvolteusingfirebuglite>
 - Debugging front end code - <https://connect.crownpeak.com/blogs/developers/debuggingjavascriptfrontendcodeinvoltepreview>
 - [Link to CSS and JS so Preview renders correctly](#)

CSS and JS can be added to the CMS two ways: imported as uploads or added using developer templates. These files are then linked in navigation wrappers. If these files aren't linked correctly, using the CMS API to complete path information, it will cause them to fail in Preview. In some cases they may still function properly when pages are published (i.e., when relative path structure happens to match that on the published site), but they are not configured correctly for management by the CMS.

For correct configuration

1. Confirm that CSS and JS are editable by adding them to the CMS as assets and linking them to developer templates. The best practice is described in detail here - <https://connect.crownpeak.com/documents/best-practices/developertemplatedesignpatterns.pdf>
2. Create links in navigation wrappers using `ServicesOutput.RenderScriptLink` Or `ServicesOutput.RenderCssLink`
`ServicesOutput.RenderCSSLink(cssID, "test", "test")`
`ServicesOutput.RenderScriptLink(scriptID);`
3. Test the Preview in CrownPeak

Linking Images in CSS

Image links in CSS files should be updated to use CrownPeak link syntax. If not, the image will not display in preview, although the page will work when published.

To link images correctly in CSS

1. Confirm that image is loaded in the CMS. Depending on where it is used, it should either be in the global assets folder or the assets folder for the specific site or locale.
2. Confirm that CSS is editable by adding it to the instance and linking it to a developer template (see above). The best practice is available here - <https://connect.crownpeak.com/documents/best-practices/developertemplatedesignpatterns.pdf>
3. Replace hard-coded URL paths with `Asset.Load` which will get and render the link to the image.
`<%= Asset.Load(imgFolderPath + "dropdown-arrow@2x.png").GetLink() %>`

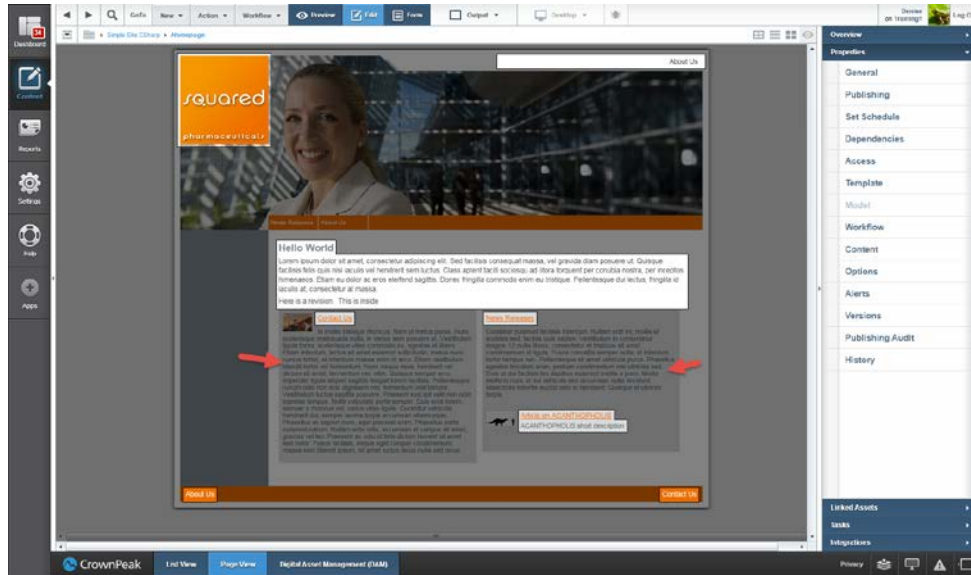
Creating Conditional Output Code

In output template files, some fields may be left blank but if they are not made conditional in the output they can render broken HTML. A common use case for this is an optional image that will appear in the page based on the content the author chooses to include.

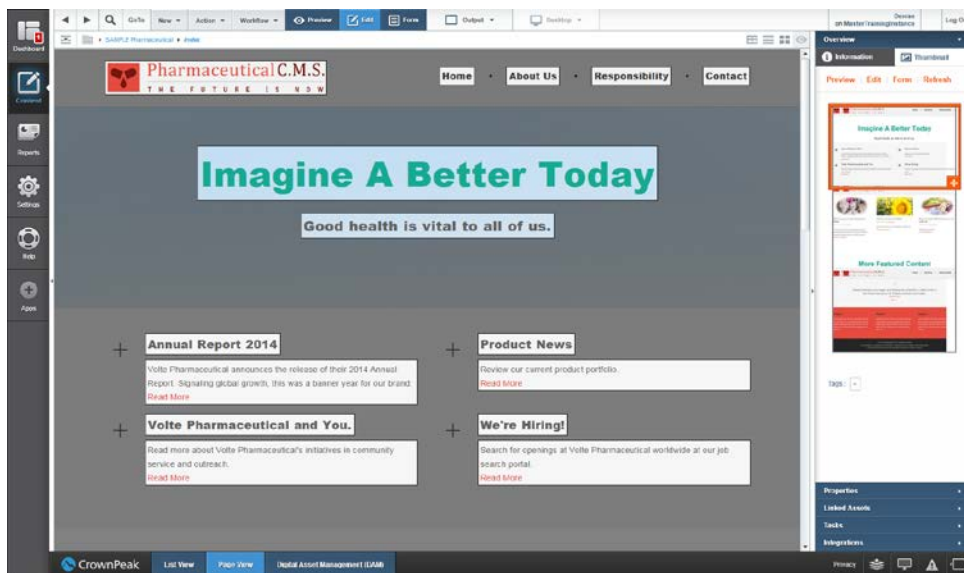
In this case you can configure this manually by using an "if statement". You can also use this services library method to manage it - `ServicesOutput.RenderConditionalString` or `ServicesOutput.RenderConditionalField`.

In Context Editing (ICE) Configuration Guidelines

INCORRECT: Some of the fields are not ICE enabled



CORRECT: Every content field can be edited from preview



ICE enables content authors to edit their content from Preview mode. ICE is applied automatically to fields using WYSIWYG, text, and image inputs. ICE can be added to other field types manually or disabled manually where it's been automatically applied.

Adding ICE Manually

ICE is included automatically for image, WYSIWYG, and text fields. You will need to apply ICE manually in the following situations

- For all other types of fields
- For fields in panels
- For fields from other assets that don't have an input template file
- For overriding the automatic mode where raw output is required

To add ICE, you must decorate your output with a call to `Util.Editable`. Note that `fieldName` must be a name of a field in your input template, not `post_input`. See API documentation here - http://help.crownpeak.com/cmsapi/index.aspx?topic=html/M_CrownPeak_CMSAPI_Util_Editable.htm

Examples

Fields from the same template

```
<span <%= Util.Editable("fieldname") %>><%= asset["fieldname"] %></span>
```

Fields in panels

```
string fieldName = "acquired_asset";
foreach (PanelEntry panel in asset.GetPanels("asset_list"))
{
    var otherAsset = Asset.Load(panel[fieldName]);
    Out.WriteLine("<li {0}><a href=\"{1}\">{2}</a></li>", Util.Editable(fieldName, panel),
otherAsset.GetLink(), otherAsset.Label);
}
var otherAsset = Asset.Load(asset[fieldName]);
```

Fields from other assets

```
string fieldName = "acquired_asset";
var otherAsset = Asset.Load(asset[fieldName]);
Out.WriteLine("<li {0}><a href=\"{1}\">{2}</a></li>", Util.Editable(fieldName),
otherAsset.GetLink(), otherAsset.Label);
```

For additional information on ICE configuration, please see this article on Connect - <https://connect.crownpeak.com/blogs/usersandadministrators/implementingincontexteditingice>

When to Override ICE Using .Raw

To make ICE to work, CrownPeak adds markup to the preview output. Developers can use .Raw to return the raw value from content field and override the behavior that adds custom javascript tags around it.

Some examples

```
string myValue = asset.Raw["my_field"];
```

CORRECT

```
if(myValue == "foo") { //code }
```

```
if (myPanel.Raw["my_field"] == "foo") { //code }
```

CORRECT

```
string myValue = asset["my_field"];
```

```
if(myValue.Raw == "foo") { //code }
```

INCORRECT

When to use .Raw

- When using if statements to test for content
- When type-casting date and time expressions in the output
- When outputting HTML attributes, CSS, JavaScript, JSON code based on content fields (for example, class or ids are common)

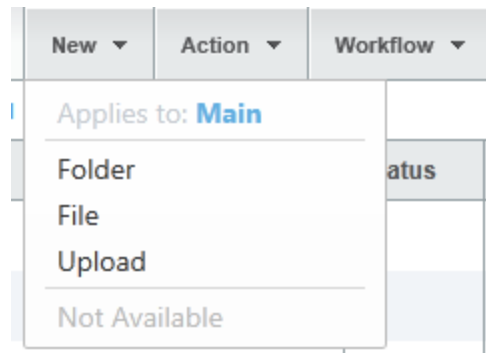
Do not use .Raw if you want ICE enabled for text, image, or WYSIWYG fields.

For more information, please review this article on Connect - <https://connect.crownpeak.com/blogs/developers/understandingraw>

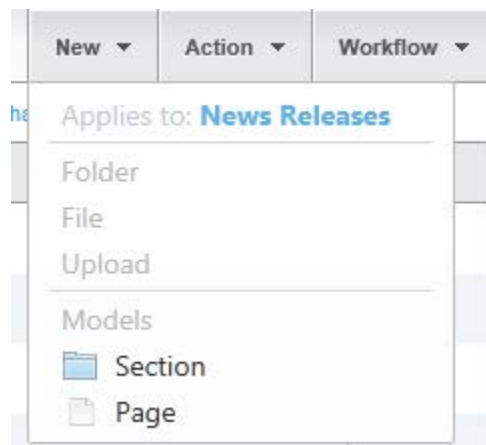
Model Configuration Guidelines

How and when to configure a model

INCORRECT: This configuration will allow developers to create intrinsic content types, but content authors are not able to create new content.




CORRECT: Intrinsic content types are disabled for content authors and models are available to create folders and pages



Models enable your content authors to create new content within a range of assigned options. You create models so users can create site pages with specific workflow, template, and access properties assigned. A model that only allows users to create pages is called a Simple Model.

The screenshot shows a web application interface with a breadcrumb trail at the top: System > Models > Model Examples > Simple. Below the breadcrumb is a table with the following data:

Type	Label	Status	Changed By	Change...	Size
	Simple Page	Draft	Denise Duncan	2/14/2013 8:12:34 AM	

You can also create models that enable users to create folders, as well as pages. There are two best practice approaches to creating more complex models:

- Intermediate Model
 - o The model includes a folder that has its own model assigned, to allow new pages. This configuration enables your user to create a page, a folder, and pages within the folder.
- Recursive Model
 - o This model includes a folder with the current model assigned as its model. This allows the user to create a page, a folder, and pages and folders within that folder.

Models are required wherever content authors will need to add new content or sections. Most site folders require a model. Some configuration folders can leverage models as well.

More information on models is available here -

https://connect.crownpeak.com/training/template_development/training_modules/documents_template_development_training/template-development-modules/6modelconfigurationpdf

Optimizing Template Performance

If you have a template that takes more time than the others to render for preview or publish, take a look at these guidelines for areas to improve and optimize.

Optimizing Asset.Load

Even though you are not creating queries, in many cases the API methods you use trigger database connections and SQL queries. Designing your templates to minimize the number of trips to the database can help improve performance.

One way to improve this is to avoid loading the same assets repeatedly. `Asset.Load()` makes a database request each time, even if it is requesting the same asset. When using the same asset, and therefore same content, it is best to store the content in a dictionary. If you are looking up a list of fields, it is best to execute this in one call rather than loading them each individually.

See this article for additional details - <https://connect.crownpeak.com/blogs/developers/assetload-best-practices>

Using Filters and Lists

- When using `GetFileList`, `GetFolderList`, or `GetFilterList`, you can and should limit the result set size with the "Limit" property on the param object that is passed to these.
- You should also consider what will be returned when setting up parameters. For example, if you create a filter list and the only parameter is the template name, this list will increase in size as more content is created with that template. If it is used frequently, this list could quickly grow in size and slow down the filter. In this case, you should use additional parameters to limit the result set by location or shared content, for example.
- You can also use `FieldNames` property to restrict the result set to specific content fields. More information is available here - <https://connect.crownpeak.com/blogs/developers/filterparams-fieldnames-property-best-practices>

Disabling Dependencies

Dependencies help manage publishing so linked assets are queued and published together. However, there are certain situations when it makes sense to disable them. For example, if you have a template that automatically creates a site map for you, the asset will have dependencies to every single site page. In this case, you should turn them off. Not only will they add to the number of assets in your publishing session, but they can slow down rendering. You can modify template dependencies by adding this code to your `output.aspx` template file

```
context.IsGeneratingDependencies = false;
```

Reducing Dependencies

Any asset that links to another generates dependencies, so when you publish one asset, tens or hundreds of additional assets will also be queued. If you consistently see thousands of assets queued, contacting support is recommended. They can adjust the dependency configuration for the entire instance, reduce the queue, and speed up publishing.

Disabling Preview Cache

```
context.DisablePreviewCache = true;
```

This property overrides default behavior for caching previews. This method should be limited to use cases only where there is a specific reason to disable caching such as a "report" type preview that should be checked and refreshed each time it is displayed. In other words, overriding caching via template code should be used sparingly.

Setting this property can impact performance because this forces re-rendering of all assets. If this is enabled in a template, you may encounter

- UI performance issues
- Degraded performance and timeouts

Editing Publishing Properties

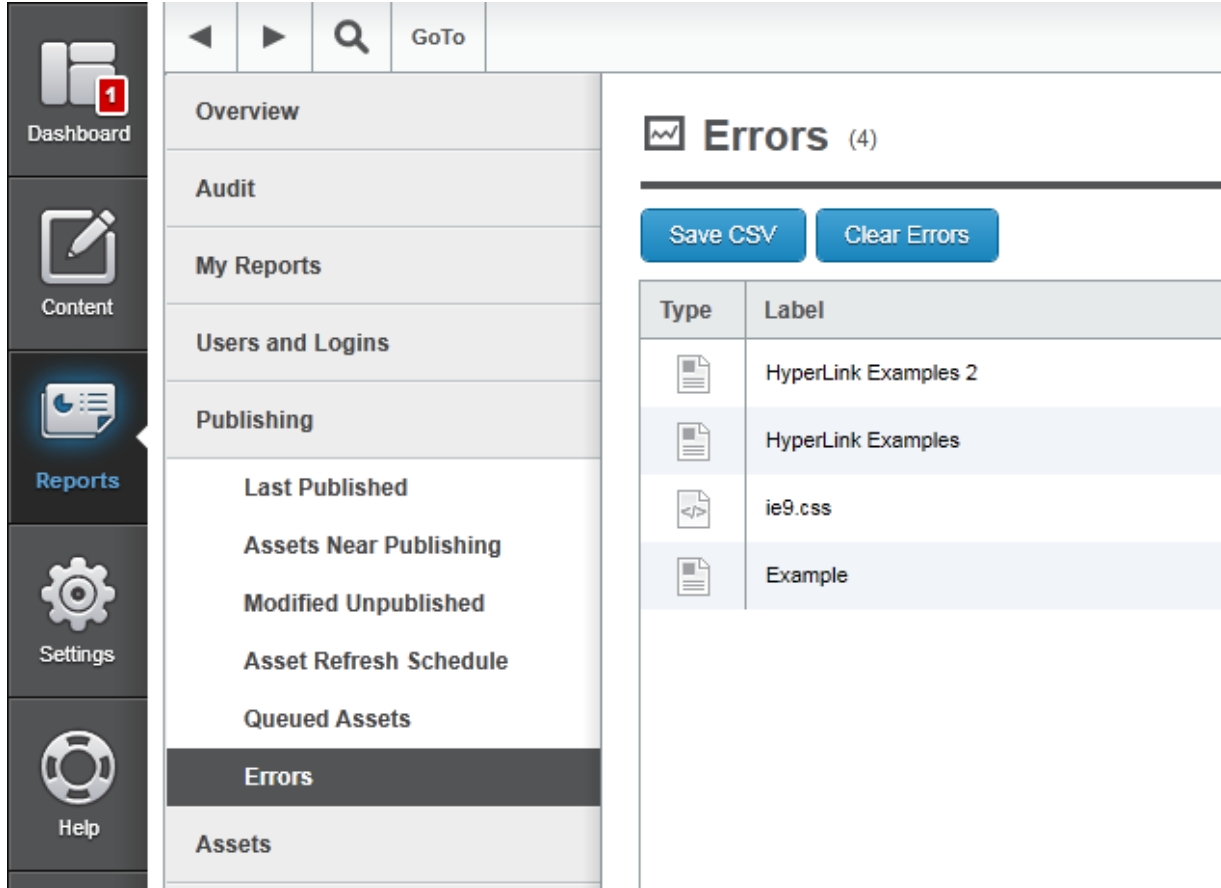
Some factors to know when modifying Properties > Publishing in the right rail.

When you edit publishing properties on multiple assets using the *Also apply properties to* feature, the entire LinkCache is cleared. It is wise to avoid these edits while publishing is going on, especially if you need to make more than one. Over time, the cache refills, but publishing will slow down as links are generated from scratch.





Clearing Publishing Errors

If there are errors, prioritize to fix them.

Errors get re-tried. There are safeguards to limit the number of tries, but it still adds to the publishing load. They should not be ignored and left to burden the system.



The screenshot shows a web application interface with a sidebar on the left and a main content area. The sidebar contains navigation options: Dashboard (with a red notification badge '1'), Content, Reports, Settings, and Help. The main content area shows a navigation menu with 'Errors' selected. The 'Errors' panel displays a table with 4 errors and buttons for 'Save CSV' and 'Clear Errors'.

Type	Label
	HyperLink Examples 2
	HyperLink Examples
	ie9.css
	Example

Notes and Additional Information

More information can be found in the Solutions section on Connect

https://connect.crownpeak.com/developers_and_partners/solutions

Notes

<https://connect.crownpeak.com>

Appendix: Input Field Types

Panels

Panel types

Tabs: Tabbed Panel, Horizontal Panel, List Panel Collapse

Tabbed Panel Horizontal Panel List Panel

Starts the tabbed panel. Similar to a windows panel, a tabbed panel (unlike an expand panel) displays a list of tabs that when selected shows the hidden panel.

List Panel Collapse

Gets the next available panel from the named list. Always returns at least one entry. Uses data already stored in the asset.

List Panel: List Panel Collapse

Name

Telephone

Horizontal Panel Collapse

Creates a container that arranges the contained controls horizontally. If the controls can't fit in one row, they wrap to a new row below the first.

First name **Last name** **Telephone**

Basic form fields

Basic Form Fields

Text Box Collapse

Adds a textbox control to the input screen. If the Optional: height, is set to a value greater than 0, ShowTextBox will render a multiline textarea.

Text box

WCO: No Snippet Attached Collapse

CrownPeak WCO Create snippet ...or select a snippet ▼

Text box

Check Box Collapse

Adds a checkbox control to the input screen.

Check if necessary

Yes

Radio Button Collapse

Adds a group of radio buttons to the input screen. If the Optional: defaultValue, is provided the radio button will be initialized to the to the defaultValue. Using the Optional: helpMessage, will render help text.

Select State
 California Nevada Mississippi
This is help text

List Collapse

Shows a list selector control, which allows the user to select values from a dictionary.

Select from list

California	>	
Mississippi	<	
Nevada		

Autocomplete Collapse

Adds an autocompletebox control to the input screen. (Try typing C, M, or N to see some options)

Autocomplete

WCO and WYSIWYG Collapse

Adds Wco controls that will bind a field on the asset with a WCO Snippet. Should be paired with an editable text fields of the same name, like a TextBox or Wysiwyg

Shows the WYSIWYG control.

WCO: teest Collapse

CrownPeak.WCO Detach Snippet ✕

Add Testing Variant Add Targeting Variant 📄 ✍ 🔗

Content

Password Collapse

Adds a password textbox control to the input screen. This features is provided as a convenience for developers. CrownPeak will not be responsible for the security of this password. Template developers will be responsible for handling the security of passwords submitted with this type of text box. Passwords can be handled in the post input template. They can be encrypted before storing or used and cleared depending on the usage. If they are not handled in the post_input, they will be stored in clear text in the asset content properties.

Password

Selector and Upload Fields

Fields to select other assets or upload content.

Color Select

Collapse

Shows a color selector control.

Select color

Date Select

Collapse

Shows a date selector control. Dates are stored in the format "MM/dd/yyyy".

Select Date

Select Document

Collapse

Displays a "select" and "clear" button that is used to upload a document into the current template.

Select Document

No item selected.

Select Image

Collapse

Displays a "select" and "clear" button that is used to upload an image into the current template.

Select Image

No item selected.

Select Folder

Collapse

Displays a "select" and "clear" button that is used to select a folder.

Select Folder

No item selected.

Drop-down Menus

Dropdown types

Dropdown Collapse

Shows a drop down menu selector control.

Select option

Dropdown Container Collapse

Creates a dropdown that will show the controls associated with the currently selected value, while hiding those that are not. All controls that fall under the StartDropDownContainer will correspond to the first item in the rows dictionary. Supports nesting, so calling StartDropDownContainer within another DropDownContainer will work as expected.

Drop Down: Links Collapse

Links

Display Only

Popup message Collapse

The popup help message, to the right of the text box, is available for most Basic and Selector input fields

Text box

?

Link Collapse

Creates a link to another asset from the input form

[Homepage](#)

Header Collapse

Adds a section header to the input screen. Useful for highlighting groups of controls.

Message Collapse

Adds a section message to the input screen. Useful for adding a message to groups of controls.

This is a message

Control Panel. Click here to open. Collapse

Creates an control panel. It is used to group a set of controls. You must close the control panel with an EndControlPanel.

Expand Panel. Click here to open. Collapse

Creates an expand panel. The expand panel presents itself as a single gray line that when clicked reveals the set of input controls that it wraps. You must close the expand panel with an EndExpandPanel.