CrownPeak
WEB EXPERIENCE MANAGEMENT

Branching Best Practices
End users, Administrators, and Developers

Table of Contents

Overview

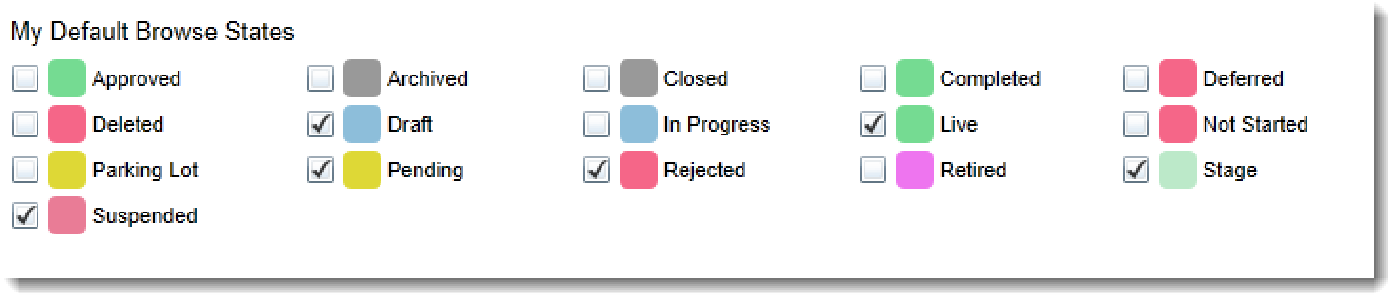
Branching in the CMS allows for making modifications to existing assets within the CMS and previewing the effects they have upon the site, without affecting the Live and Stage versions published to the server. One of the primary benefits of branching over cloning is the automatic resolution of links from other assets without re-pointing. In addition the branching process in conjunction with workflow can control the presence of duplicate assets within the same path to avoid overwriting files on the web server.

Upon branching an asset in the CMS, an exact copy of the data is made from the Original Asset. The new asset by default starts its new life in the CMS in the Draft state. The content in the Branched Draft is stored separately from the content in the original asset, so changes made to one will not affect the other.

CMS Behavior

To prevent unwanted/unapproved content from appearing on the Live and Stage versions of the site, the CMS has some automatic checks in place. This filter is applied based upon the template layout file, and the state of the asset making the API calls.

When previewing content in the CMS, by default all states that are selected under user preferences are displayed (See Figure 1). Unlike publishing, items that are for example "LIVE" can see and/or link to items that are in "DRAFT" (lower states) as long as they are checked in the My Default Browse State.



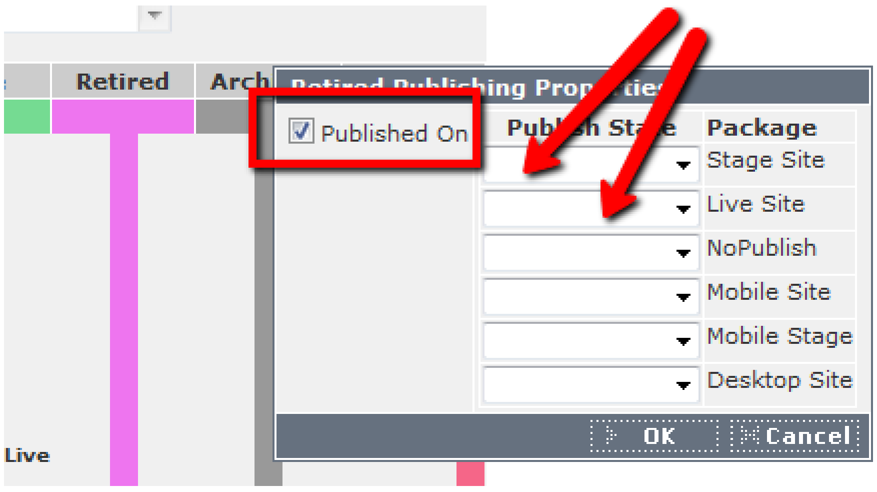
My Default Browse States

☐ 🟩 Approved	☐ ⬛ Archived	☐ ⬛ Closed	☐ 🟩 Completed	☐ 🟥 Deferred
☐ 🟥 Deleted	[✓ 🟦 Draft]	☐ 🟦 In Progress	[✓ 🟩 Live]	☐ 🟥 Not Started
☐ 🟨 Parking Lot	[✓ 🟨 Pending]	[✓ 🟥 Rejected]	☐ 🟪 Retired	[✓ 🟩 Stage]
[✓ 🟥 Suspended]				

Figure 1.

When publishing content, the CMS will filter content by STATE. Items in a lower state than the published state will not be seen and/or linked by the publishing asset. For example if there is an index page that is being published to the LIVE server, items that are DRAFT or STAGE would be excluded from those links. Conversely if that same index was being published to STAGE it would include links to be STAGE and LIVE assets. Assets can see and/or link to content that is higher or the same when publishing but not lower.

The exception to this rule specific to publishing involves workflow configuration and/or published content. Any state that an asset has not been published to will not be seen or linked to by any asset that has not been published to that server, even if that state is higher. This is specifically how Retired and Archived assets are excluded. Under workflow configuration the publishing state is configured on, but the packages are blank (see figure 2). This actually forces the asset to be removed from the server instead, thus excluded from being seen or linked. This can also occur if that package has never been published to prior (such as a new publishing package/destination).



Retired Publishing Properties

Published On	Publish State	Package
✓		Stage Site
		Live Site
		NoPublish
		Mobile Site
		Mobile Stage
		Desktop Site

Figure 2

The simple rule is that when involving published content, assets can see or link to only assets in an equal or higher state that currently is published to that package/server destination. Conversely content in a lower state or that does not exist on that publishing package/destination are excluded.

Workflow/Life Cycle

The typical life cycle of an asset that incorporates branching would be as follows:

1. Draft of new asset is created.
2. Content is submitted for approval.
3. Asset is published to stage and reviewed.
4. Final changes are applied and Asset is published to live.
5. Asset is branched with new version created as Draft status.
6. Branched asset is edited and changes submitted for approval.
7. Branched asset is published to stage and reviewed.
8. Final changes are applied and the Branched Asset is published to live.
9. When the Branched version is published to Live the previous version is automatically transitioned to Retired and that version is removed from the web server.
10. Previous asset is automatically or manually transitioned to "Archived" which hides the asset from view within the CMS.

Note: Workflow/Lifecycle is based on a standard workflow of Draft, Pending, Stage, Live, Retired, Archived. Your workflow may vary but the same fundamental life cycle is applicable.

Digital Assets

Digital assets are defined as assets that do not have a template and are added to the CMS through the CMS upload. Common digital assets are images (jpg,png,gif) and documents (doc, pdf, xls). For CMS instances created prior to June, 2014, digital assets could not be branched in the same manner as they were stored as standard files, so no two could co-exist with the same name in the same folder. As of June, 2014 all new CMS instances now support fully branchable digital assets. For instances created prior to June, 2014, your CMS instance can be converted to support branchable digital assets. This does require a conversion of existing digital assets. The conversion typically takes 1-2 hours at which time the CMS is inaccessible. This conversion process is typically scheduled as a maintenance process. If you would like to enable this feature, please contact support, who will coordinate the maintenance to convert your CMS instance.

For any instance not yet converted, when one of these assets is branched, it will rename the new branch, therefore it is not recommended to use branch on digital assets. If a particular digital asset may require regular updates (such as a policy document), it is recommended that an asset wrapper template be used such that the asset can be branched. Contact CrownPeak for more details on an asset wrapper template, though it is not recommended that instead you convert your CMS instance to support the new file storage and only use an asset wrapper when additional meta information is required for your digital asset.

End User Best Practices

- Whenever making substantial changes to an asset, that asset should be branched. This provides the user the ability to review the revised content prior to that content being published to the live server. This also provides a means to rollback changes.
- Never make changes to a LIVE asset that cannot be published to live. A common mistake is users will make updates to live assets without branching with the mindset that they haven't published the asset to live it should be ok. However that content may still get published to live either through dependency publishing, content used by other assets getting published or simple human error. If changes are made to a LIVE asset, assume that those changes can go to live at any point.
- Branching an asset should be used instead of cloning and/or creating a duplicate asset for the purpose of maintaining revisions of assets. This provides inherent controls within the CMS to handle linking of assets as well as handling conflicts that arise by having two assets with the same location and name within the CMS. This prevents unwanted file overwrites when publishing.
- Avoid using un-branch as a matter of practice. Un-branching an asset will break the cms resolution of links that are stored in various assets. The un-branch option is for special cases where it is recognized that the new asset is not a branch of the existing asset.
- Avoid multiple branches of the same asset in the same state. Typically only one LIVE branched version can exist of the same asset. This is handled within the workflow as a "conflict". However other states, such as STAGE may not have the same conflict rules applied. Retaining multiple versions in the same state may result in unexpected results. Use discretion when maintaining multiple branched copies with the same state.
- Avoid outdated branched copies. Following the typical workflow an asset is branched with the intent of creating a new updated version of an asset. There are times when that version will not be promoted to live. For those versions they should be retired if they are not intended for live to eliminate any confusion when creating new branches.
- Move RETIRED assets to ARCHIVED status. This is often done with a workflow transition. This prevents the working folder from being cluttered with branched versions. Typically ARCHIVED status is hidden from the user within the workflow and keeps the workspace clean. Some workflows maybe configured to hide RETIRED assets as well.
- Moving a STAGE asset back to DRAFT requires the LIVE version to be republished. If both a STAGE and LIVE version of an asset exists and the STAGE version is moved back to DRAFT and/or to a non-published state, it will remove the asset from the server and there will be no published record for that state. If this occurs always republish the LIVE asset to re-establish a STAGE version of the asset (or any lower publishing state).

Administrator Best Practices

- Disable editing for LIVE assets. The CMS allows setting permissions (ACL) on asset STATE. By disabling editing of live assets it will force the user to branch any live asset to make new changes. This will prevent non-approved changes from getting published as the result of dependency publishing. This also will force the changes to process through workflow and receive the proper review and approval as defined by your company's workflow configuration.
- When copying/pasting a folder or cloning a folder, use Clone Special or Paste Special. These alternate UI functions allow the user to select which state(s) to paste or clone. When copying or cloning a folder with multiple assets when the new folder is created, all items will be immediately set to "DRAFT" (or the lowest configured state). The normal paste/clone utilities will copy all items regardless of state, therefore you can end up with multiple duplicates all in the same DRAFT state with no way to determine which is the right one. **Note**: If these menu options are not available contact CrownPeak to enable this feature.
- Use the route option for moving multiple assets from state to state. When multiple assets need to be moved from state to state, such as moving an entire new site from Draft to Stage, use the Admin Route tool. This tool will allow the user to use "Also Applies To" to select multiple assets. The most common is using siblings (same folder) or neighborhood (multiple folders down). When this runs it will provide a list of assets for final approval or removal from the list. When routing multiple assets from state to state, select the current status that you wish to route from. This ensures no unwanted assets are provided in the list. This feature is only available in Volte, the classic UI currently does not support this feature. Note: When routing multiple assets some items may publish out of order i.e. the index before the children pages. As a result some items may not link from this process. To resolve use the site refresh option to refresh the same folder again to that state. This will clean up any issues with links due to order of publishing.
- Disable un-branch from normal user groups i.e. non-admin and non-developer roles. The un-branch option is best used for very special use cases and should not be used as a matter of process. Most if not all intended behavior can be handled without un-branch and it can cause issues with links not resolving when assets are un-branched.

Developer Best Practices

- Do not override the default CMS behavior with filter status. Avoid using Asset.SetParam "filter_status","*" (VB) or ExcludeFilterStatus / FilterStatus Property (C#) in output. This disables the CMS behavior for controlling which asset to use at publish time and can cause unexpected results. These are provided as tools for special cases, not as standard use cases.
- Always use models for creating new assets. Models ensure that the asset workflow is set at initial creation. Creating after the fact can result in some assets not having workflow or having the wrong workflow.
- Avoid assets without workflow. The CMS will allow assets to not have workflow, but if the CMS instance is specifically using workflow, it is best to have all assets under workflow to ensure proper approval for content.
- Avoid storing content in the folder for items that need to maintain alternate versions. The CMS will allow content to be stored in the folder object instead of the asset. However you cannot branch folders in this manner, even if the content is set with an asset that is. This can result in unexpected content going to the LIVE server that is not approved.

- For workflow steps that are between published states, make sure publishing is disabled. For many workflow configurations there are often workflow states between publishing i.e. STAGE, READY FOR LIVE, LIVE where that state does not publish. Make sure that state has publishing turned off to avoid removing the asset from the previous state, in this case STAGE. Only states in which content should be removed but not published should have publishing turned on with the package state blank.