Ektron to EPiServer Digital Experience Cloud: Information Architecture

This document is intended for review and use by Sr. Developers, CMS Architects, and other senior development staff to aide in the process and considerations when upgrading from a legacy Ektron CMS to the new EPiServer Digital Experience Cloud (DXC). It covers strategies and considerations for porting information architectures, including:

- How the Digital Experience Cloud manages the Content and Site IA
- Ektron Menus
- Ektron Taxonomy
- Ektron Folders
- Ektron Collections
- Ektron Metadata (as an IA tool)
- Specialized use-cases for Ektron Smart Forms

This document does not establish best practices or considerations for mapping content or other data between the two systems. Examples of source data types not covered in this document include, but are not limited to:

- Unstructured Content
- Structured Content
- Smart Forms
- Assets such as Images and Documents

These items are covered in a separate document: *Ektron to EPiServer Digital Experience Cloud Content Transfer Guide*.

Table of Contents

Definitions and Assumptions

Definitions

1. **DXC** - **Digital Experience Cloud**. The primary EPiServer product consisting of CMS, Commerce, or both as part of a Cloud- or SaaS-based package.
2. **Cloud** - EPiServer's managed hosting packages and solutions.
3. **Site Information Architecture (or Site IA)** - Refers to the structure of the website supported by the CMS. Typically reflected in the primary navigation or menu structure of the site.
4. **Content Architecture (or Content IA)** - Refers to the structure or structures within the CMS in which content may be organized. This may or may not be a direct reflection of the Site Information Architecture.
5. **Folder Hierarchy** - A data architecture based on "folders" or similar virtual containers that do not themselves represent content. Folders imply a one-to-one relationship between the folder objects and the content they contain - one content item belongs to one folder.
6. **Content Hierarchy** - A data architecture based on a "child content" model, in which nearly every content item in the system is a direct child of another content item. Content Hierarchy implies a one-to-one relationship from child to parent. A content item may only have a single parent content item.
7. **Taxonomy Hierarchy** - Also referred to as Categories. Similar to a Folder Hierarchy in that the Taxonomy "nodes" do not typically represent content. The distinction being that Taxonomy Hierarchies can support a one-to-many relationship. A single content item may be "categorized" using multiple Taxonomy nodes.
8. **Menu Hierarchy** - Similar to Taxonomy Hierarchy in that one-to-many relationships are supported, though seldom employed. A Menu Hierarchy directly represents the navigation of the site and generally coincides with a Content Architecture, though limited to only those items which one wishes to appear in the site navigation. Menu nodes may or may not represent content.

Assumptions

This document currently lists no assumptions.

Introduction to the Project

Upgrading a customer application from Ektron CMS to the new EPiServer Digital Experience Cloud introduces changes to the internal Content Architecture which will often require the developer to consider the base structure being moved from Ektron into the EPiServer DXC.

Advantages of a Content Hierarchy

Though Ektron can be considered more flexible due to the separation of the Site IA from the Content Architecture for content storage, this presented a great number of challenges for both customers and developers.

Ektron Folder Hierarchies across the customer base have often been found to have grown in ways that were restrictive when looking to future improvements of the application. For example, a too wide-open structure making it difficult to later implement more restrictive role and permission management.

Enforcing a best practice approach in which deviations are the exception rather than the rule better equips the customer and developer to implement future improvements to the application.

Authoring

A common failing in mis-managed Ektron Folder Hierarchies is that the author entering the Workarea would get somewhat lost in a myriad of folders, most of which they didn't need to access. EPiServer's Digital Experience Cloud uses a Content Hierarchy that matches the Site IA, so as long as the author can find the content in the site, they also can find the content in the tree.

Implications for Upgrading

Because the DXC uses a Content Hierarchy whereas Ektron relies upon a Folder Hierarchy for the storage of content, the development team will likely wish to formulate a plan in which Ektron's Site IA structure (whether Menu or Taxonomy or otherwise) is moved along with the limited scope of content it represents. After this point, additional content can be loaded on a per-type or per-folder basis from Ektron into the correct locations in the Content Hierarchy in the DXC.

Overview of the Digital Experience Cloud Information Architecture

EPiServer's site IA follows a Content Hierarchy model, in which most content is a direct child of another content item. While it offers less flexibility than the variety of options available in Ektron, this style of architecture provides a direct correlation between the management of content and browsing of the site.

Content in EPiServer can be flagged for removal from the primary navigation, but by default all top-level content items are made available for navigation.

EPiServer also supports a commonly employed practice of using "containers" – specialized content without any public renderings and without representation in the site IA. These container objects operate similarly to folders, and can have properties of their own, but are not typically displayed in the menu, though their child items may be represented with specialized navigation separate from the primary site architecture.

Pro Tip:

Containers are one way of "deviating" from the best practice of maintaining a single content tree per site within EPiServer DXC. This can be useful for segregating content with special use-cases from the main application. While you may have containers in the middle of the primary content tree, this is considered bad practice as the expectation is that each node of the tree will have an associated rendering.

Site IA Management in Ektron

While there are several options, the site IA for an Ektron-powered site is primarily derived from one of two architectures: Menus or Taxonomy, and rarely derived from the internal content architecture directly.

Menus

Menus are the primary approach to providing a static site navigation, as they natively support options for internal links, external links, and links to digital assets. Each of these also supports other properties, such as a link image, description, and target frame. Adding items to the menu creates new “Menu Item” instances in the DB that contain references to the content in question, but are not a part of the content object.

All *Menu Items* follow the same structure, with properties like Title, URL, Image, Target, and Description. By default, a new internal link to content inherits the Content Title as the link text and will use the Default Alias as the hyperlink, or the non-aliased QuickLink (direct template link) if no aliases are configured. Other properties are not pre-configured and may be changed only after the *Menu Item* is created.

The second menu object type is a *Submenu*. A submenu inherits the same base properties as a Menu Item, but adds some options and functionality to enhance management of the navigation. Though a submenu can be linked to content, either internal or external, one point of distinction is that submenus are permitted to have child items in the form of menu items or additional layers of submenus.

In addition, Submenus may have “associations” with templates or folders. A template association might be considered similar in use-case to partial routing in EPiServer. The intent is to better manage the “selected” state of the item within the menu rendering. If, for example, you have a template that lists content and may allow filtering through additional querystring values, a *Template Association* will ensure that the menu item remains in a selected state even as the querystring is modified.

Under normal circumstances, /Template.aspx?id=3000 might be the linked item in the navigation. Because /Template.aspx?id=3000&filter=products isn’t an *exact string match* to the

top-level item, a *Template Association* of /Template.aspx will ensure that the item remain selected as long as that template is in use.

The second form of association is a *Folder Association*. This type will set a *selected* state on the given item as long as content within one of the associated folders is being displayed to the visitor. A classic example here would be Press Releases. You may have a menu link that takes the visitor to the list page for releases, but you wouldn't want the overhead of creating menu links for each release published.

Remember that there is no parent/child relationship between content within Ektron. A *Folder Association* allows you to specify that the Press Releases menu link remains in a selected state as long as any content in the Press folder is being rendered to the site.

One readily identifiable downside to this separated architecture is that the URL paths do not necessarily mirror the navigational path within the content or menu hierarchy. Because of this and other issues, such as ease of management, some Ektron customers turned to Taxonomy as an alternative solution to the site IA.

Taxonomy

Born out of extreme uses of menu systems to create a solution for categorization of content, Taxonomy is one of the most used – and most abused – features within Ektron. The fact that it is hierarchical and that content can be readily assigned to it without permissions elevated far above that of standard authoring have lead Taxonomy to be a contender for managing a site IA. Pair that with Taxonomy-based automatic aliasing and it seems to be a fair match.

Taxonomy structures must be created and managed by someone with *Administrator* or *Taxonomy Administrator* permissions. The root node then can be made available to content on a folder-by-folder basis (with inheritance options for child folders). When editing a content piece, the author then has the option of using a new *Category* tab in the UI to assign the current content a position within any Taxonomy tree assigned to the parent folder. Similar to menus, this association is not tied directly to the content, but rather via a new *Taxonomy Item*, which represents the relationship between node and content.

Also like menus, Taxonomy does not offer a true content hierarchy. In fact, because Taxonomy does not offer the same configurable linking options that are find in Submenu nodes, it can become more difficult to establish a link-to-node relationship. Developers often have resorted to hijacking properties of the node (such as the description field), relying on convention (node name must match content name), or extending the node, via a supported customization feature, with a new property in which do store a reference to the content to be associated with that node. None of these solutions are particularly elegant and each of them adds to the decision-making requirements and labor of the administrators and authors.

Taxonomy also proves challenging when a customer desires links to custom templates (pages that are not represented by managed content) and external domains. Designed and intended for hierarchical categorization of internal resources, the Taxonomy feature has no out-of-the-box way to manage category associations for unmanaged or external URLs. A variety of workarounds have been created. Some clients will use the Ektron Library *Hyperlink* object, since Library resources also may be categorized, or a custom *Smart Form* (content type) to represent custom or external resources. While these can and do work, workarounds such as this tend to make the navigation more difficult to manage, particularly if such unmanaged or external resources are commonplace.

Moving the Site IA to EPiServer DXC

When planning the migration of the site IA, it's important to note that the EPiServer IA is dependent upon the content of the site. Because of this more direct relationship, one cannot import the structure independently of the content, though it may be possible to import the content without the structure.

Regardless of which architecture is used to drive the site IA in Ektron, it will be possible to maintain a similar structure in EPiServer and this can be done via automation. For Menus and Taxonomy, the two most common structures for site IA in Ektron, Ektron's APIs provide access to the data in its original hierarchical format. As content is imported, this original structure may be used as a reference guide for the placement of content within EPiServer's architecture.

This import strategy implies that content modeling exercises should be completed and that content coming from Ektron should be successfully mapped to the destination content structures first. Using the exported IA as a reference will allow the destination IA to be built automatically as content is added.

Other Ektron Data Structures

Folders

Ektron users and developers will be accustomed to an array of options and a variety of best practices for defining architectures for both the content and the site experience. However, at its core, Ektron uses a folder-based hierarchy for storage of content.

In addition, Ektron favored distinct, pre-defined content types for various functions within the platform: one object type for content, inclusive of Smart Forms, and one for Page Layouts, i.e. PageBuilder, one for Events, etc. This paired with the need to index content properly lead to best practices such as the separation of Content and Pages folders.

For more information on recommended practices in folder structure, see: https://developer.ektron.com/Experts/James-Stout/The-CMS-Architecture-Theory/

Note that though there may be recommended practices, they are certainly not going to be the case across all Ektron customers – many of whom have a single, mostly flat structure in which to house all of their content. Customers with such a structure often face difficulties in managing search indexing, roles & permissions, content workflow and approvals, and more.

Folders are often the basis for lists of like content, such as Press Releases. Though they may reflect the site IA in some cases, folders are rarely used to drive navigation directly.

Content URL aliasing, on the other hand, is often automatic and based on folder path paired with the content title.

Collections

Collections are similar to LinkCollection properties within EPiServer – a flat, ordered list of content references. They often are used to drive flat navigation elements (utility or footer navigation, for example), but rarely the primary menu of a site. In addition, you'll find collections used for manually controlled lists such as featured items on the home page.

Collections often can be restricted to a specific folder of content – unable to reference content outside of that section of the internal architecture. When best practices are followed, this also implies a restriction on the Content Type(s) permitted within the Collection since one of Ektron's best practice recommendations is to restrict any given folder to a single Smart Form.

Metadata

The primary response when considering the term "metadata" typically centers on HTML metadata – or SEO-related markup of the page. In Ektron, metadata is a way of extending any given content model that can, yes, include title, keywords, description, and more.

Metadata is not part of the content model, however, but rather assigned to the content as an artifact of the metadata options being made available to, and in some cases required by, the content's parent folder configuration. This allows content models in separate folders to inherit different metadata options and requirements based on their respective location in the internal content architecture. Though it's uncommon to have such separation, one cannot rely on a given content model to always be limited to the same finite set of metadata definitions.

In the context of Information Architecture, Metadata can be used to reference and create implicit relationships between items of content within Ektron CMS. Moreover, these references are not limited to content-to-content. They also can support content-to-folder, content-to-image, content-to-collection, content-to-hyperlink (as a Library item type), content-to-menu, and content-to-user.

For more information on these types, please read the documentation. http://documentation.ektron.com/cms400/v9.10/Reference/Web/EktronReferenceWeb.html#Content/Metadata.htm#Creating4

This flexibility in generating content relationships that are not directly seen through any other architecture, such as Folders, Menus, or Taxonomy, serves to emphasize the need for in-depth discovery and research focusing on the current definitions of content and current practices in content entry and management.

Specialized Smart Forms

In recent years, Ektron's Smart Forms have also been used for administrative control over the site as well as management of single strings of content on the site that might otherwise have been hard-coded or placed in an XML file.

While some EPiServer partners and customers are accustomed to string translations being managed via XML files that reside as part of the application project, this may not be the case for many Ektron customers who have been introduced to translatable content models specifically designed to manage individual, and often re-usable, strings on the site. This would include content such as the site logo, the placeholder text in the search box, the copyright statement, and even button or link text within repeated areas.

In cases such as this, one best practice recommendation is to separate these fields into a non-display entity, such as a Block, that is registered as a field in the Start Page, which is maintained in cache as long as the application is active. One or more of these administrative blocks would provide the end user with a familiar method for entering and translating these micro-instances of text and other properties on the site.