Ektron to EPiServer Digital Experience Cloud Content Transfer Guide

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 content items, as defined herein, including:

- HTML Content
- Smart Forms
- Page Layouts
- Widgets acting as Content Items

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

- Menus
- Taxonomy
- Collections
- Folders
- Widgets acting solely as views for Content Items

These items are covered in a separate document: *Ektron to EPiServer Digital Experience Cloud Data Structure Mapping 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. **Find** - EPiServer's cloud-hosted enterprise search and content relevance application, included as part of DXC Cloud packages.
4. **"Complete" Content** - Content within the CMS that represents a self-sustaining page of information and has no dependencies on other content. For example, a Press Release might be considered complete.
5. **"Atomized" Content** - Content within the CMS that represents only a single portion or component of the page, but is not succinct in and of itself. It is intended to supplement "complete" content or to be employed in a series of other atomized items.
6. **Page** or **Page Type** - A unit of content within the DXC which represents a complete website page of content. Pages are granted unique URLs for access via web browser.
   Ektron equivalent: HTML Content, Smart Form (if representing a complete content item) or Page Layout
7. **Block** or **Block Type** - A unit of content, or atomized content, within the DXC which represents a page component or partial unit of content. Blocks are only accessible as part of a parent Page Type and are not granted unique URLs.
   Ektron equivalent: Smart Form (if representing atomized or component content), Widget
8. **Media** or **Media Type** - A unit of content within the DXC which represents a document file (any format), video, or image.
9. **Content Item** - An encompassing term representing any or all units of content within the DXC, including Page, Block, and Media Types.
10. **Sizing** - The process of estimating the number of "Complete" and "Atomized" content within Ektron CMS for the purpose of determining the most suitable Digital Experience Cloud package for the project.

Assumptions

1. This document assumes practices around an automated or scripted transfer and does not cover aspects of manual transfer either in full or in part, though nearly all such projects have a manual component.
2. This document assumes deployment into an EPiServer Cloud packaged environment. Most information will apply to either cloud or on-premises deployments equally.
3. This document assumes that the reader or implementor has received, at a minimum, EPiServer CMS Developer Fundamentals training and is therefore familiar with basic EPiServer best practices.
4. This document assumes that the reader or implementor is familiar with Ektron CMS and related code practices and therefore does not provide definitions for components of the Ektron platform.

Introduction to the Project

Upgrading a customer application from Ektron CMS to the new EPiServer Digital Experience Cloud introduces some breaking changes to the way content is modeled and created.

Modeling

Content modeling within the Digital Experience Cloud is done by a developer directly in .NET. This provides an immediate advantage, allowing the developer to work directly with content as objects rather than stepping it through any transformations or intermediary formats such as XML. In addition, the code can now request and use data directly with the platform, speeding development time and allowing the developer to spend more time in native code.

The new changes to content modeling also introduce *inheritance*, dramatically improving on the flexibility and reusability of code while ensuring greater levels of consistency for the authoring experience.

Using inheritance, a developer or content strategist can easily establish common properties for many different content types. This is especially useful for properties like SEO metadata.

Authoring

Through this advanced modeling, authors will also take advantage of more opportunities for content reuse and better content organization - for both complete and atomized content.

In addition, authors will now have the opportunity to work almost completely within the page context, rather then in distinct experiences for different types. Plus, customization can be added at the field level, giving authors even further enhanced functionality and productivity.

Implications for Upgrading

For some content models, these changes will mean breaking apart existing content and remodeling it in order to bring it in line with EPiServer DXC - and industry - best practices for content strategy. While it may be possible to replicate other models for easier transfer, we strongly recommend reviewing those models to ensure data is not being duplicated, the total number of models is minimized, and that the application is taking full advantage of the capabilities of the upgraded system.

Right-Sizing Your Project for EPiServer DXC SaaS

Unlike most of our top competitors, EPiServer Cloud (SaaS) solutions are sized based on usage. So sizing the project is critical in helping to understand into which package a project will fall. For most projects, this will come down to the total number of "content items" needed by a project.

Most marketers are at least somewhat aware of the total number of pages on their site, but that's not the same thing. A content item, in this context, is any piece of content managed (or indexed) by EPiServer's SaaS solution. This includes, but may not be limited to: documents, images, videos, pages, and blocks (atomized content). The total number of content items, then, will likely be at least 2x the total number of pages (considering that a site is likely to have a minimum of one image per page of content).

However, reuse of content will bring this count down. A single image will be counted only one time, though it may be used many times across the site or across channels. Similarly, content pages may be displayed as a block - without actually creating a new block instance - without furthering the count.

Therefore much depends on the implementation itself. When planning for the number of page types and block types, attempt to estimate an average number of items for each type. Depending on opportunities for reuse, the blocks estimate might be significantly lower or higher than the pages estimate. Also attempt to estimate an average number of images per page type or block type. Then it becomes a matter of calculation.

Content Item Estimate

Page Types	Count Estimate	Images / Page	Blocks / Page	
Start	1	10	8	
Press Release	2500	1	2	
Product X	10000	7	3	
	Pages Total	**Images Total**	**Blocks Total**	**Content Total**
	12501	72510	35008	120019

As you can see, the number can rise quickly, especially for commerce.

Pro Tip:

If you under-calculate, EPiServer's Cloud packages to allow for iterative increases in capacity. For example, if you estimate 400,000 items but the client actually needs 550,000, then they don't necessarily need to jump to the million-item package. EPiServer's Cloud packages allow for incremental increases in scale. The fictional customer above may be able, for example, to increment their package up to 600,000 content items. Perhaps more, depending on projected growth.

Data Transition Basics

The implementation partner or architect should understand some high-level points when planning the transfer of data between two systems.

Volume

The first step in a data transfer is to identify the various types or models required. This means that you will need to have at least some basic definition for each of the Page, Block, and Media types to be transferred and a quantity estimate for each. If you've completed the "Right Sizing" exercise above, then you should already have this information.

The volume - or quantity - of each type will determine whether the opportunity is right for scripting or manual transfer between the two environments. The tipping point may depend on multiple factors:

- Scalability of the technical team for scripted transfer development
- Scalability of less-skilled teams who may perform manual data entry
- Willingness of the client or customer to take on manual data entry
- Velocity of content generation (covered in detail below)

For example, if you have content that is reasonably static, and the customer is willing to dedicate three resources x four weeks to data entry for a total of 480 man-hours, and you can reasonably estimate that it will take 5-7 minutes to manually transfer each item, then you can rely on that team to transfer approximately 4,000 items.

3 team members x 4 weeks x 40 hours per week = 480 man hours

480 man hours x 60 minutes per hour = 28,800 man minutes

28,800 man minutes / 7 minutes per transfer = ~4,114 transfers

Compared to scripted transfers, this is extremely slow and would never keep pace with high-velocity content (content that is under a state of near-constant change). However, it may prove valuable for static or low-volume content.

In addition, manual data entry can be more reliable when transitioning from unstructured to structured content, particularly if the unstructured content does not follow a consistent paradigm for formatting.

Pro Tip:

While screen-scraping utilities can also serve as an aide in transferring unstructured information, they can suffer from a higher error rate when compared to transferring structured data. Third-party systems that specialize in such transfers also can be used to mitigate this risk and provide successful transfers from unstructured to structured. We recommend looking for applications that either work directly with the database or document object model (DOM).

Velocity

Velocity is most useful in identifying the breadth of the window of opportunity for transfer. In other words, the amount of time that you may disable editing in system A, perform the transfer, then enable editing in system B. This is also known as a “content freeze.”

A low-velocity site is one in which changes are done infrequently and with typically low levels of urgency. For example, a hospital or an informational B2B website may be considered low velocity. Such a site may be campaign-driven or have a focus on a small number of products that change infrequently, but are not generally sites with very active blogs, forums, news, or other quick-turn content. These sites typically are able to support larger windows for transfer, sometimes supporting multiple days or weeks to perform and test the transfer.

A high-velocity site is one in which changes are done frequently and with necessary immediacy. News sites, like Huffington Post, for example, will publish sometimes dozens of articles per day, have highly active blogs, as well as comments or other user-generated data.

Whether a site is considered high- or low-velocity there will be cases in which you are not able to institute a full content freeze but may allow content to be added or entered to the source system during or after the transfer into the Digital Experience Cloud. If that is the case, then there must either be a follow-up synchronization between the two systems or a temporary requirement that the authors must post changes within both systems until the upgraded application is launched.

Keep in mind that the reach of a site may also impact the transfer window. Sites that serve a global audience do not have many or lengthy windows of opportunity for downtime. If your transfer project will negatively impact production site performance or cause any downtime, such must be taken into account alongside the velocity.

Pro Tip:

Different Page, Block, or Media Types within a site may have different velocities. The advantage of this strategy is that you may have not one content freeze, but several of varying lengths as you transfer different content types or site sections and align these transfers so that the freeze may be lifted on all at once when the new site is enabled.

Transposing DXC Content

Ektron developers should appreciate the changes coming about with the transition to the EPiServer Digital Experience Cloud. While these new Page, Block and Media types appear familiar in form and function from an authoring perspective, they are, in fact, quite different and significantly more advanced under the hood. With the addition of DXC and Find APIs, Ektron developers will be able to retrieve, sort, and filter data in ways that would have never been available under Ektron's legacy data storage and management strategies. This is in addition to providing a more consistent content creation experience which places the practitioner in-context for nearly all authoring tasks, rather than a couple of select functions as is the case in Ektron CMS. This change allows structured content, e.g. Smart Forms, and page-building, e.g. PageBuilder, to operate within the same context and flow, thereby dramatically increasing the efficiency of content creation.

However, this is considered a breaking change and requires that the data be transitioned to operate in the new paradigm. While planning the data transfer, the developer or architect should objectively evaluate the usability and effectiveness of the current Ektron-based content strategy to determine what portions will fit within the modeling of the Digital Experience Cloud.

Pro Tip:

While many Ektron content types will likely transfer with little or no modification, this also should be seen as an opportunity to put that imported data into models that reflect best practices for storage, management, and reuse in the new system. That includes taking advantage of inheritance, the ability to render Page Types as Blocks and more.

DXC Pages and Blocks

Most of the Ektron content, particularly Smart Form-based content, will be readily moved into either DXC Pages or Blocks, depending on the usage for the content (Complete versus Atomized).

Despite the name, Page Types are not tightly tied to any particular rendering, whether web, app, or other channel. However, Page Types are often served independently or as the main body of content alongside a number of atomized content items to any selected channel.

Blocks, on the other hand, are typically not rendered or delivered to any channel independently of some parent content item, e.g., a Page. Blocks will generally be functional (e.g. Carousel), promotional (e.g. CTAs or Ads), or supplemental (related or associated content).

In contrast, Ektron handles all structured content, whether complete or atomized, with Smart Forms. In some advanced Smart Form configurations, Group Boxes can be used to essentially build a repeatable, nested structure. When transposing this structure into the DXC paradigm, such a Smart Form may actually need to be split into more than one model. This will allow for

greater permission management as well as increased opportunities for content reuse for the "child" model.

For example, an Ektron-based Press Release built using a Smart Form may have top-level fields for *Date* and *Body* and also include a repeatable Group Box for *Press Contact* details. When transposed to the DXC, the Page would consist of the actual content fields: *Date* and *Body*. The *Press Contact* fields, however, should be considered separate, atomized content even if it's not put into a separate structure within Ektron.

Here is an example of how such Ektron Smart Form XML content may be stored. The Group Box field is bolded for emphasis.

```
<root>
  <Date>2015-06-27</Date>
  <Body>NEW YORK, NY - EPiServer announced today…</Body>
  <PressContact>
    <Name>Sarah Jane Smith</Name>
    <Title>PR Manager</Title>
    <Email>sjsmith4@tardis.tl</Email>
  </PressContact>
  <PressContact>
    <Name>Amelia Pond</Name>
    <Title>PR Coordinator</Title>
    <Email>apond11@tardis.tl</Email>
  </PressContact>
</root>
```

In order to replicate this inside the DXC, the models (overly simplified in this example) may appear similar to:

Page:

```
[ContentType(
    DisplayName = "Press Release Page",
    GUID = "c76db04c-8bde-47a0-b9e2-37e8b694725a")]
public class PressReleasePage : PageData
{
    public virtual DateTime Date { get; set; }

    public virtual XhtmlString Body { get; set; }

    [AvailableContentTypes(
        Availability = Availability.Specific,
        Include = new[] { typeof(PressContactBlock) })]
    public virtual ContentArea PressContacts { get; set; }
}
```

Block:

```
[ContentType(
    DisplayName = "PressContactBlock",
    GUID = "6f907e65-e6f5-4a39-b6e0-b1a34eee0d00")]
public class PressContactBlock : BlockData
{
    public virtual string Name { get; set; }
    public virtual string Title { get; set; }

    [EmailAddress]
    public virtual string Email { get; set; }
}
```

In this example, the Group Box in Ektron's Smart Form is separated from the main content model and reinterpreted as a Block Type. In the Page Type, note the ContentArea included to attach Press Contact information to the main content. Separating these models makes single-entry, multi-use content out of the *Press Contact* information and provides a single point to make changes to this information.

Pro Tip:

Content may be transposed as a single Page or Block Type, or to any combination of the two. Take this as an opportunity to better prepare the content for an improved authoring experience as well as cross-channel distribution. Don't just make it again, make it better.

This also illustrates one of several ways to create content relationships within Ektron: nesting. In this method, the two objects are part of the same model. Relationships between two disparate content items may also be established through collections, taxonomy, or even metadata. This topic is covered in more depth in the *Ektron to EPiServer Digital Experience Cloud Data Structure Mapping Guide.*

Pro Tip:

Ektron's metadata, including that for SEO, is defined and stored separately from the content item itself, though the two are edited in the same interface. Within the new Digital Experience Cloud, the content and metadata are all part of the same model. When transforming Ektron CMS content, make sure to include this information as part of the imported data.

Ektron Widgets as Content

In a typical Ektron implementation, Widgets are used primarily as *views* for Smart Form or other content, whether to provide the primary or an alternate view, such as a list or promotion. Widgets also may be used to bring in content or functionality not at all associated with the CMS. In some cases, however, the widget can contain actual content in the form of headings, labels, button or link text, or even complete sentences and paragraphs.

This data may be more challenging to associate with items being transferred from Ektron CMS to the Digital Experience Cloud and may be better put into consideration for manual transfer. Regardless, because it is a view that may also contain configurable fields for all or part of the rendering, it must be accounted for in the new content modeling strategy.

Pro Tip:

When a widget is used to render Smart Form or other CMS content, it usually also contains a property for the source Content ID. This is the key for tracing a widget back to the source material. Accessing these properties is done by retrieving the parent Page Layout and parsing the XML using development tools such as Linq-to-XML.