Ektron to EPiServer Digital Experience Cloud: Steps for Success

This document is intended for review and use by Marketing or IT Managers and other stakeholders in the application upgrade process. The goal of this document is to outline benefits of the new Digital Experience Cloud in addition to the recommended preparations to ensure a successful upgrade, including:

- Content Inventory
- Link Management
- Keys to Success

This document does not attempt to cover all aspects of upgrading to the latest version of the EPiServer Digital Experience Cloud. For complete details and advanced topics, please see the following related documents:

- *Ektron to EPiServer Digital Experience Cloud: Data Structure Mapping Guide*
- *Ektron to EPiServer Digital Experience Cloud: Content Transfer Guide*
- *Ektron to EPiServer Digital Experience Cloud: Users and User-Generated Content*

Table of Contents

Definitions and Assumptions

Definitions

1. **Content** - Any item stored within or otherwise managed by your Content Management System which is intended to be consumed by one or more audiences, including web page content, images, documents, and other assets.
2. **Unstructured Content** - Content which does not have a predefined model or pattern which is enforced upon the author entering the information.
3. **Crawler** - A “crawler” is a software utility which will open a page on a web site, read the page for links to other pages or assets, and follow each of those in turn repeating the process. Crawlers generally keep an internal record of which pages, images, or documents have already been crawled in order to avoid repeat effort.
4. **API** - An acronym for Application Programming Interface, the term API references a software package’s own interface for working with that software. In the context of this document, an API provides one means by which content may be retrieved or added to a system directly.
5. **Orphan** - An orphan is a content item - or microsite - to which no other items are linked. For example, a campaign landing page that is not linked from other pages within the site may be considered “orphaned.”
6. **UAT** - An acronym for User Acceptance Testing, the term UAT references the process the customer, owner, or “user” of an application undertakes to assure themselves that the application delivered by the vendor or system integrator meets their acceptance criteria.
7. **Deep Link** (AKA Cross Link) - A term used to describe the commonplace scenario in which one item of content within a system provides a direct URL link to another content item within the same system. Generally applied because such links are often taking visitors “deeper” within a site or application, as opposed to a home page or top-level landing page.
8. **Redirect** - A rule or configuration which specifies that a visitor to *URL A* should be automatically taken, or “redirected,” to *URL B*. Redirects may be configured to be either *temporary* or *permanent*. In the context of this document, redirects are generally assumed to be permanent.

Assumptions

1. This document assumes an EPiServer Cloud deployment, though on-premise is available.

Introduction to the Project

Upgrading to the new EPiServer Digital Experience Cloud will introduce changes which require remodeling content as well as variable amounts of development effort to address changes in the display. These changes, as part of the upgrade, will bring your site into modern best practices and will bring into the core platform greater versatility and opportunity, better arming you and your team to tackle the challenges of the modern Marketing and IT worlds.

Benefits for Marketers

When moving your site into the Digital Experience Cloud, you nearly gain all of the benefits the cloud has to offer. This includes an industry-leading uptime guarantee not just for the infrastructure, but for your application.

With a cloud solution, global reach and scalability are readily achieved and you can roll your messaging out into new countries and regions on your own schedule. Scalability and rapid content delivery is enhanced with the inclusion of intelligent CDN capabilities that work well with EPiServer personalization and targeting.

Personalization and targeting also are now part of the core product. In the new UI, personalization is front-and-center, dramatically easing the burden of organizing campaigns created to target each of your visitor personas.

The new DXC user interface also optimizes the efficiency of your team by offering 100% of the authoring experience directly through the canvas of your own site. This converged product has eliminated slow and confusing admin interfaces. Content authors work within the environment which is most familiar to them - your own site - and can even preview their changes, as any given persona, and at any point in time.

You also will be able to organize many changes throughout the site - to content, images, and assets - then preview the site with those changes and launch all of them on the same schedule. This feature, called Projects, simplifies the rollout of campaigns and product launches without interrupting regular daily updates to the site.

The latest release of this converged cloud platform requires changes to the core of the platform in order to bring all of these amazing enhancements to life, including changes to the way content is stored and managed, changes to the way we develop and organize visitor groups and personas, and even changes to the way the application renders the display.

This is a major upgrade to the product which brings with it equally major opportunities to improve your internal efficiency as well as take advantage of tools which can allow you to rapidly adopt and adapt to changes in the marketplace.

Benefits for IT & Developers

Adopting a cloud-based solution has obvious benefits for IT teams. Improved scalability, better uptimes, and more manageable cost structures. By allowing the infrastructure to be managed by a dedicated team, IT managers can reassign internal talent to developing enhancements or providing better support for internal systems and departments.

The DXC is XCOPY-friendly, which allows us to operate in the cloud at a scale and efficiency that, at time of writing, other platforms cannot match. For your team, this means operating in more countries and regions with greater reliability and speed, helping your application achieve target response times world-wide.

Our team proactively notifies you of any necessary adjustments to accommodate changes in traffic, so you can better deliver even when facing unforeseen spikes. We also provide state-of-the-art CDN, which can be configured to deliver not only digital assets, as most CDNs are configured to do, but also the HTML content itself. This configuration keeps all content at the edge of the network, close to the customer.

Distributed cloud computing helps protect your digital property from DDoS attacks and more. Our team uses best-of-breed tools to monitor, mitigate, and generate reports to add critical ability to handle and contain potentially devastating attacks.

Development within the application adheres very closely with Microsoft best practices. Working with MVC and EPiServer is, simply, working with MVC. Our best practices and reference architectures empower standard .NET developers and engineers to get started right away and boosts their efficiency while also boosting employee satisfaction by keeping the applications they develop in-line with the most advanced technologies coming out of Microsoft.

The advanced authoring interface within the Digital Experience Cloud is highly self-discoverable. For IT teams, this leads to fewer support calls and deviations to address minor issues or provide basic instructions. As DXC developers, your team can allot more of their time to enhancing the application through new features.

Our extensible-by-default approach to product development allows IT teams to readily tie into existing systems, including CRM, PIM, ERP, and more. Many of which have pre-built connectors available as add-ons to the core platform.

The upgrade to the latest converged Digital Experience Cloud platform, and an optional subsequent deployment into the EPiServer cloud, is intended to dramatically improve the developer experience and boost employee satisfaction by allowing them to work with state-of-the-art software and industry-leading practices, while at the same time giving your organization the lift it needs to deliver personalized content with amazingly low latency to every region in which you do business.

This is a major upgrade which introduces a number of breaking software changes necessary for us to continue to deliver software that meets the high demand of all of our customers. We recommend that you perform your own cost/benefit analysis to demonstrate the long-term value of this essential upgrade to your web applications.

Establishing a Content Inventory

It is said that failing to prepare is preparing to fail. Holding to that precept, it is recommended, if not essential, that prior to starting the upgrade to the Digital Experience Cloud, you should establish a working content inventory. This being the system of record designed to establish which content items will be pushed into the new application.

Because upgrades to the DXC are often predicated by the need for a redesign, this also is an opportune time to perform a content audit as part of developing and refining the inventory.

Crawlers Versus API

There are two primary methods by which this content inventory may be created: using a crawler or accessing the content directly using the API. Both have their advantages and disadvantages.

Crawlers will accept a specified starting point - often the Homepage URL - and begin working through the entire site, following link after link in the start page and each subsequent page. They can require less development than using an API, but these platforms may make assumptions or lack access to data that is not rendered by the page being loaded. While crawlers have no built-in method for inventorying orphan content, they may work well during a content audit for identifying which items you do or do not use on your site.

In addition, crawlers may rely on each page's URL as a unique identifier, which is not guaranteed to be accurate in configurations which allow multiple rendering URLs per content item - particularly cases in which URL canonicalization is not employed.

APIs provide direct and complete access to the content by giving a developer a way to "loop through" the entire structure of the CMS. While this works well for getting all content, it may get content that would fall on the wrong side of the line during a content audit. It also may be more challenging to map the content into parent/child relationships using the API, though those relationships are necessary for moving the content into the DXC.

Following the benefits and detriments of each, those who opt for a strategy which depends on a crawler will often find themselves using an API approach as a supplement to cover those areas in which a crawler falls short, such as completeness of data. Meanwhile, those who take an API approach tend to rely solely on that approach and their own ability to develop whatever logic necessary to accommodate its shortcomings.

Crawler vs. API Comparison

	Crawler	API
Completeness of Inventory	**Mixed** A crawler will be better equipped to retrieve linked pages, files, images, or other assets which are not managed by the parent CMS. However, crawlers will miss items such as orphaned pages and files. If not everything is CMS-managed, this can yield better results.	**Mixed** An API will be guaranteed to find all content managed within the CMS, but may miss non-CMS pages and other assets.
Completeness of Data	**No** A crawler relies on the content being served to the browser, and so cannot reliably acquire all data to be transferred. This often results in the need for an approach which combines the crawler results with an API to complete the models.	**Yes** The API will work directly with the platform and direct object references to retrieve and inventory complete content models. This may take multiple calls, but can yield more reliable results.
Orphans	**No** Orphan content to be preserved in the new system would have to be added to the inventory manually.	**Yes** All content can be found and retrieved, even if orphaned. However, this may yield more results than desired putting greater emphasis on the need for an audit.
Content Relationships	**Possible** Some crawlers ignore this, some make an automated attempt. If attempted, the structure should be scrutinized with great care to ensure accuracy.	**Possible** This would require some additional thought and effort since content will come from a Folder structure and the relationships from menus or taxonomy.
Content Transforms	**Possible** Some crawling utilities can provide assistance in transforming content from one model or pattern (or no discernable pattern) into another. When transforming unstructured content, this can require a large amount of monitoring and adjusting.	**Yes** Because one is usually working with data that is already structured, it tends to be easier to transform one model or pattern for content into another. This will suffer similar limitations to crawlers when working with unstructured content, however.

Orphans

Orphaned content can exist in three types:

- Single pages or assets with specific purposes. E.g., campaign landing pages
- Micro-sites that are kept distinct from the parent application, but fall under the same domain.
- One of the above types, but which is no longer active or viable and should be removed.

The first challenge, as described above, is in identifying all of this content. From there, the content owners should ask:

- Is this content still valuable?
- Should this content remain orphaned?
- If it's a micro-site, should this become a completely separate site? Alternatively, is this micro-site part of the upgrade or should it be addressed separately?

These content items and micro-sites can often have strategies that are distinct from their parent application - else they would likely not be orphaned. As such, the go-forward strategy for each of these types should be reevaluated as part of developing and auditing the content inventory.

For the third type above, this serves to highlight the importance of performing a content audit. There are occasions in which content may be, for all intents and purposes, removed from the site, yet remain dormant within the CMS. This content hoarding can be detrimental to the upgrade project and increase associated costs.

Assets such as images, PDFs, and other files also may fall into these categories and be moved erroneously.

Images and Other Assets

When planning a migration, customers often make the mistake of assuming - with or without forethought - that images or PDFs or other content linked from within web pages is simply part of the page to be migrated. This, however, is not the case.

Images and other assets represent distinct content items within the CMS and therefore must have an associated strategy for transfer and preservation. Especially within Ektron's DMS (Document Management System), these items may have their own metadata, taxonomy, and other properties which need to be ported into the Digital Experience Cloud.

Link Management and Maintenance

Link preservation should be a top priority for any project involving a shift in the underlying architecture of the data. Even if a site is not focused on marketing, for which SEO is always a primary concern, high-utility applications, such as those common in B2B and portals, often have pages that are bookmarked by the end user. This, however, does not imply that the resource URLs should remain unchanged, particularly when there's an opportunity for them to be more human-readable, more consistently formed, and applied with better adherence to current standards.

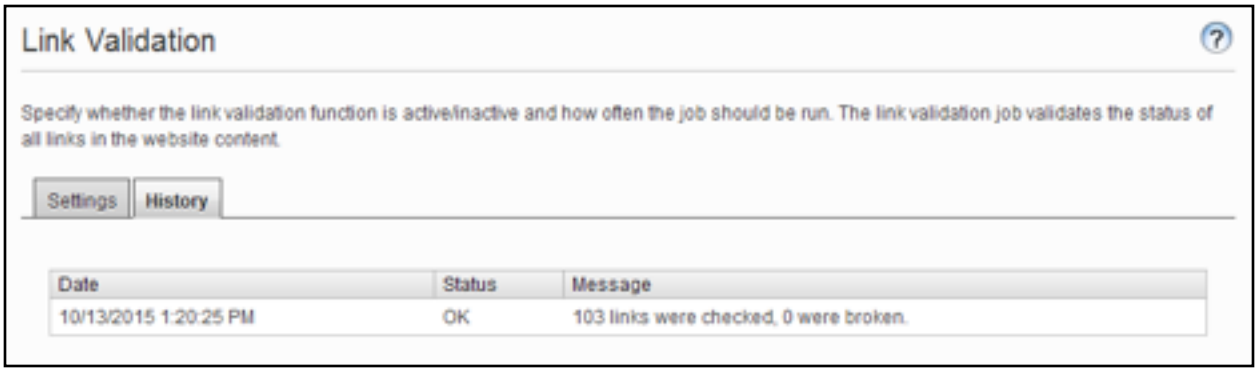
All of these aide in the predictability and discoverability of any given resource as well as provide a boost to SEO by having the "content" of the URL match that of the resource to which it points.

Deep Linking

So-called "deep linking," which is also referred to as "cross linking" as well as other terms, is a term used to describe instances in which a content item contains a URL, or equivalent reference, pointing to another resource or content item within the same system. E.g., you may add a link to an article which refers the visitor to another article, job posting, document, or other resource also managed by the CMS.

Assuming that the source system uses actual URLs in its provided "deep linking" functionality, this can cause problems in cases in which the content is transformed and thereby granted a different URL.

Popular platforms, including EPiServer, contain Link Validation utilities that can comb through content within the platform to verify the in-content links are valid.



While tools such as this can help identify problems, they can't automatically address them. For this reason, developed or scripted tools to aid in transform and transfer of content between the two systems will often keep a record of transferred items and perform multiple passes. During each pass, the software can evaluate the links within content, check whether the linked item has been moved and a new URL assigned and, if so, provide an update to the embedded URL.

These tools, whether custom or pre-built, 3rd-party applications, can provide a tremendous amount of comfort in knowing that prior efforts around valuable cross linking of content is not lost and does not need to be performed manually.

Updating deep links, however, fails to address another problem: SEO.

Redirects

Accounting for and correcting deep links within content is valuable and necessary to the success of a project. Essentially, this could be described as updating content due to updates in the content architecture. You're informing content in the system of changes brought about within the same system. This does not, however, address concerns which arise from outside the system. Apps may retrieve content for display by URL, and are less readily informed of the change, for example. Along similar lines, this does not inform search engines such as Google, Bing, or Yahoo of the new URLs for the content. For sake of brevity, the remainder of this section will refer to the search engine as "Google," though similar rules apply to all search engines.

An XML Sitemap - a file which follows a specific convention in order to inform Google or other search engines of the content on your site - is a great way to ensure your content is crawled and discovered. It does not, however, indicate to Google that the content at these new URLs is the same as that from the old URLs. In short, it doesn't provide Google with the information it needs to preserve the results of prior efforts around SEO for the given content.

Let's assume that content is moved from Ektron into EPiServer and, as such, is granted a new URL. Without additional effort, when the site launches, the old URL will simply cease to function while the new URL is "starting fresh," from an SEO perspective. What you'll see is the immediate decline of the old URL and the cessation of organic traffic to that content (which wouldn't do any good, as it was just assumed that the URL ceases to function). Meanwhile, the new content has to await the next crawl to be found at all, at which point it will gradually work its way back up in the Google ranks to even match the keyword/results location of its predecessor.

Redirects help address this problem by informing an application that the requested content or resource has been moved to a new location - a bit like redirecting your mail at the post office.

Most popularly, redirects are available in two forms: permanent and temporary. Temporary redirects are useful for campaigns or other site changes which would cause an org to only temporarily want to send traffic from page A to page B.

In the context of new content architecture and associated new URLs, permanent redirects are desired. Effectively, this immediately informs Google of the change upon the next attempted crawl, allowing Google to update its own records and apply any theretofore earned rankings to the same content under the new URL, instead of treating it like brand new content.

Also consider that Google does not instantly index content as soon as it's updated. The crawl frequency for any given page can vary greatly, even within the same site, and depends on the page-rank or popularity of the page in question. This may mean several hours, days, or even weeks for Google to pick up the changes. During that time, Google may be serving up your content under the now-antiquated URL. Having a redirect strategy in place ensures that, though

Google or other sites may present a visitor with the old URL, the visitor always lands on an active content page rather than receiving a less-useful error page.

Note that strategies around ensuring successful crawls and positive momentum in page-rank are many and, at times, complex. Each of these strategies should be taken into consideration when planning the site architecture, content structure and requirements, the site design, performance requirements, and more. Redirects are only one strategy among many for promoting search engine success. Discuss and work with your implementation partner and/or marketing agency to ensure that you are doing all that you should in regards to boosting search and other organic traffic.

Keys to Success

Success in any project relies on the same things: knowledge in the complexities or “forces” involved in the project and diligence in planning. Below are some additional points of consideration for your upgrade project.

Multiple Passes

Providing updates to content transferred throughout the transition can be massively beneficial for multiple reasons.

1. Multiple passes provide better opportunity to address necessary changes, such as updates to deep links, as described above.
2. Multiple, continuous passes can keep high-volume content refreshed, minimizing the “content freeze” window. This means that your content authors can keep working with the reasonable expectation that the majority of their changes will also be reflected in the new platform without the overhead of dual-entry.
3. More opportunity to test, and assure the success of, the new content architecture. By supporting multiple passes, you can begin the transfer earlier and begin testing the new environment for accuracy and performance.

Whether using a custom or packaged solution, look for its ability to successfully run multiple or continuous passes.

Error Threshold

Understand that variances in the source data can, and almost certainly do, introduce the opportunity for failure in any scripted transfer. This does not necessarily indicate a failure in the script as a whole.

Scripted or programmatic efforts are valuable because of the savings they provide. Assume, for the moment, that you have 150,000 content items (recall that this includes content, images, documents, and other managed items) to transfer into the new architecture. If you were to hire a team of 25 interns at minimum wage ($7.25/hr) to move these items at an average rate of one item every four minutes, then you would spend over $70,000 over a period of 10 weeks to perform a single pass of the content items. This also assumes that the items are moved with 100% accuracy, which would almost certainly not be the case.

Project Metrics

Interns	Minutes/Item	Items	Rate
25	4	150,000	$7.25

Calculations

Total Minutes	(Minutes * Items)	600,000
Total Hours	(Total Minutes / 60)	10,000
Total Dollars	(Total Hours * Rate)	**$72,500.00**
Hours / Intern	(Total Hours / Interns)	400
Total Days (at 8hrs / day)	(Total Hours / 8)	50
Total Weeks (at 5 days/week)	(Total Days / 5)	**10**

In addition, a human-powered effort introduces the risk for human error, which is unpredictable in nature. You may look at 10 items entered by humans and note 10 distinctly different errors.

A programmatic effort, though it takes time to configure or build, can perform the same task in a fraction of the time, but with a more predictable nature to the errors. In essence, if you discover an error in one item transferred via script, then you can reasonably assume that the error is consistent in other items and address it by altering the configuration or code.

Because a programmatic solution takes time to build and configure, it's reasonable to assess how many items of content to be moved are worth one hour of development time. Meaning that you may determine that an hour of development time has a similar value to 100 manually transferred, or 200 manually corrected items (this is hypothetical, please do your own assessment). By this measure, you can determine whether additional hours of development to address an issue can be considered worthwhile.

In the case of 150,000 items above, assume that the program accurately moves 120,000 items, leaving 20,000 items moved with defects and 10,000 items not moved at all. Using the value-equivalency numbers above, addressing either case would be considered worthwhile if it can be done in less than 100 development hours. If the 20,000 items with defects can be corrected for 40 hours of development, but the 10,000 unmoved items can't be addressed for less than 150 hours, then you may have a clear direction to revisit development for the former and hire interns for the latter.

Note that this decision should be based on a number of factors, and not just cost of development versus interns. For example, the 10,000 items may require specific knowledge of complex product information and require 100% accuracy. In which case, the number of minutes per item moved may increase dramatically, increasing the value attained of each development hour, or the content itself may be of such high value that it cannot be risked to untrained hands.

Prior to executing the script, you should work with the team performing the task to establish acceptance criteria with as much cost/benefit analysis as can reasonably done. The acceptance criteria may be a % of the total items either unmoved by the script or moved with defects, requiring manual effort to address in either case. You also may establish acceptance criteria based on the content type since, as stated earlier, content can vary in both complexity and value.

Project User Acceptance Testing

Nearly all projects of this nature will have some ratio of both programmatic and manual effort. Customers often will assume the latter portion of this, though unless they have prior experience they may not know precisely what that entails or which internal staff will be responsible.

Nonetheless, the outset of the project should include some division of responsibilities between the customer and system integrator. This may be that the customer is completely responsible or that the agency is completely responsible, or anywhere in between. The precise division is less important than having it established in advance.

Once the project is delivered and considered complete by the agency, this is when the customer needs to review the delivered project for accuracy and completeness of the information. Unfortunately, other than to say that the customer should review a sample of the content (not each and every one of 150,000) there are no hard and fast rules to guide the customer through this process. However, there are some recommendations.

1. Ensure that you have the proper staff on hand and committed to the effort for reviewing whichever sections of the delivered application are their specialty. For example, it's not a best practice to have a marketer review items that are under the purview of a merchandiser. The persons' involvement also should not be assumed. Rather, expectations should be set and agreement to meet those expectations should be sought.
2. Select samples that are statistically representative of the size and breadth of the population. For example, a news station may sample for a survey only persons in New York. That sample, however relevant for that region, cannot be said to reflect the views of the entire USA. Similarly, a random sample of 10 people, no matter how distributed, cannot be said to represent the general viewpoint of millions. This may require a basic understanding of statistical research and analysis. Even if that knowledge is sought outside the team directly involved in the project, it may well prove worthwhile.
3. Allow the value of the content to drive the sample selection. You may establish that product content is inherently more valuable than blog content. Therefore it will behoove your organization to select a larger sample and spend more effort reviewing the products, even if you have a larger set of blog posts.
4. Review agreements made at the beginning of the project before proceeding. You should know where the onus of responsibility for any given defect lies - with the customer or with the agency. A frequent knee-jerk reaction when finding an error is to report it to the development team immediately. However, customers should be aware whether their agreement with the agency stipulates any customer responsibility and exactly what those are. Having a prior understanding for the division of responsibility can boost your position when presenting a defect (or being presented with one) as well as save both time and frustration that can arise from misunderstanding.