XSLT Strategies and Best Practices for Your CMOD Migrations

Our series on Content Management on Demand migrations continues with a look at best practices for migrating from an on-premises CMOD system to a cloud-based CMOD. We'll share the XSLT methodologies that have worked well for our existing customers during very large moves to a hosted CMOD environment. We'll also discuss common situations you may experience and best practices to solve any issues.

DAS has worked with content management since 1994. As we're seeing many of our public sector customers move to the cloud, we're excited to bring our experience with file management, automation, migration, and more to the table. Learn the best practices for your move to a cloud-based CMOD.

When we're moving from an existing, established CMOD system to a cloud-based CMOD, we need to be able to export all of the objects from the on-premise system over to the destination. Obviously, it's vital that no data is lost in this process. In this situation, we've found it's best practice to use the arsxml process to make this transfer.

The arsxml 'export' function can be used to export a list of objects from your original CMOD source system. These objects can include everything that's currently stored in your CMOD system, from users, groups, applications, storage sets, and any other object type you've created. The resulting list is formatted as an XML file which includes all the settings related to these objects.

You can make any changes that are necessary to the XML file before moving it to the destination CMOD using the arsxml 'add' function.

During the process of making the necessary changes, you may run into certain problems with the export. These problems may keep you from adding it to the destination system, so it's important that we solve these errors now.

First, when you produce the XML file, you want to validate that it's in the correct format with proper parameters. When you're running files on the same system, this file should always check out. However, it's best practice to confirm before moving forward.

Next, transfer the exported XML file over to the destination system. We can do an arsxml file validate at the destination system. This helps ensure that files are properly formatted even when moving between different systems.

After validation is when we may see issues with the file. For example, we may find some deprecated attributes that are no longer in use in current systems. There can be old user input errors, incorrect date/time default values, or other incorrect attribute settings.

Transferring your CMOD system provides a valuable opportunity to find outdated information and bring files up to date. It's best practice to remove any deprecated attributes before we can validate the file in the new system.

We could manually edit XML files to remove these deprecated attributes, but that's not efficient or realistic in systems with hundreds or even thousands of application groups and storage sets.

Instead, write XSLT to make changes to the XML file in bulk. This allows you to make changes to all of your files at once. You can remove all outdated or unsupported attributes from your system. Simply define all of the attributes to be removed in the XSLT file. When you execute the file against the XML file, a new XML file will be produced that does not include the unsupported attributes.

Once the new XML file is produced, you're part of the way through the file transfer. The new file can be validated by arsxml 'validate' with no errors by the destination CMOD system. Next, we'll use arsxml 'add' to add it to the new system.

However, we're not finished yet. Files that are properly formatted can still produce errors during the 'add' process.

For example, we may find values or storage sets that are not valid for the new operating system. These values need to be changed before the transfer is complete. To solve this issue, it's still best practice to use XSLT to find and modify invalid values or attributes.

Once you've created the XSLT to fix and remove deprecated attributes, run it against the XML file to create another new file. The new file will include the updated attributes that are compliant with your new CMOD system.

The benefit of using XSLT is that this strategy allows you to easily update many different application groups, applications, users, groups, storage sets, etc. If you're not using XSLT to update in bulk, your transfer process will take much longer than it needs to. Ultimately a migration should be efficient and economical, so we encourage you to use an XSLT methodology for a smoother migration.

At DAS, we're familiar with making CMOD migrations to many different clouds. We can leverage a hybrid cloud strategy to make your experience as smooth and easy as possible. Please reach out to DAS to discuss the best strategy and build a plan for your migration needs.

SOURCES USED:

- https://drive.google.com/file/d/1TAhN_3hoa7J6gAcl1O1c86_QTxZ-q-RT/view?usp=sharing
- https://drive.google.com/file/d/1_O-neVsdDWqn4CkTqqjwlxAzETyrpY6_/view?usp=sharing